Received June 23, 2020, accepted July 7, 2020, date of publication July 21, 2020, date of current version July 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010789

Hyperion: A Visual Analytics Tool for an Intrusion Detection and Prevention System

SEUNGHOON YOO[1,2], (Member, IEEE), JAEMIN JO[2], (Member, IEEE), BOHYOUNG KIM[3], (Member, IEEE), AND JINWOOK SEO[2], (Member, IEEE)

[1]Department of Computer Science, Republic of Korea Air Force Academy, Cheongju 28187, South Korea
[2]Department of Computer Science and Engineering, Seoul National University, Seoul 08826, South Korea
[3]Division of Biomedical Engineering, Hankuk University of Foreign Studies, Yongin 17035, South Korea

Corresponding authors: Bohyoung Kim (bkim@hufs.ac.kr) and Jinwook Seo (jseo@snu.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (MSIP), Korean Government under Grant NRF-2019R1A2C2089062 and Grant NRF-2019R1A2C1088900, and in part by the Hankuk University of Foreign Studies Research Fund.

ABSTRACT Intrusion detection and prevention systems (IDPSs) are at the core of protecting an enterprise's network. In general, IDPSs use pre-defined rules to detect potential attacks. As the size of an organization grows and new types of intrusions appear, the quantity and complexity of the rules also increase. Moreover, IDPSs generate an overwhelming number of logs that are challenging to handle and analyze. For a more effective and integrative analysis and management of the rules and logs, we propose a novel visual analytics tool, Hyperion. Hyperion interactively visualizes rules to help users understand how the IDPS rules are managed and applied to the enterprise's network entities. Hyperion also provides effective visualizations to enable users to visually analyze the type, period, traffic, and frequency of attacks in addition to a traditional count-based timeline visualization. Finally, Hyperion enables users to interactively simulate the effect of a change in parameters of a detection rule. These features can help streamline the security control cycle consisting of rule application, information collection, log analysis, and rule revision.

INDEX TERMS Cybersecurity, intrusion detection, visual analytics.

I. INTRODUCTION

Network attacks have been increasing since the inception of the Internet. Thus most organizations employ cyber security systems to defend their networks against such attacks. The security systems of these organizations usually consist of various infrastructures, such as intrusion detection and prevention systems (IDPSs), firewalls, virtual private networks, and enterprise security management solutions. An intrusion detection system (IDS) allows security experts to inspect raw network packet data and detect intrusive activities, such as malicious codes, vulnerability attacks, and abnormal access attempts. In addition to an IDS, an IDPS enables a more active approach, blocking threats based on various criteria specified as IDPS rules. With this ability, the IDPS became an essential component for maintaining enterprise security.

The IDPS mainly uses pre-defined rules to detect network attacks. When a network packet arrives, the IDPS examines whether the packet violates any of the rules. If the packet is determined to be an attack, the packet is logged as an alert and is blocked based on its severity.

As network attacks evolve and grow, understanding and managing IDPS rules have become more important. Through a collaboration with security experts in a major information company for more than one year, we identified the following challenges in operating an IDPS.

- **Managing the sheer number of rules**: As network attacks have evolved over the past few decades, IDPSs have added a number of rules to defend networks against the attacks. For example, their IDPS has about 4,000 rules. Moreover, some rules must have variants depending on the context. For example, suppose that there is a network management system that periodically pings an application server for a health check. To distinguish this behavior from a *ping flooding attack*, the administrators should change the rule for the corresponding attack so that the normal behavior is not recognized as an attack (i.e., *false positive*). Such a change results in the creation of a new rule based on the previous one. Although these cases should be done

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

very conservatively and carefully, we found they were not very rare. Finally, as the size and complexity of the network grow, the number of rules is likely to increase.

- **Balancing the trade-offs in the parameters of rules**: IDPS rules have several parameters; for example, rules have *acceptance count* and *acceptance time* to determine which packet is regarded as an attack. If packets with a particular attack pattern continuously arrive, and their number surpasses the *acceptance count* within the *acceptance time*, the packets are regarded as an attack, and an alert log is generated. These parameters can have a significant impact on the IDPS. If an expert increases the *acceptance count* of a specific rule, the number of attacks to be captured decreases; that is, fewer logs are collected at the cost of missing real attacks. However, when the expert sets the *acceptance count* to a smaller value, more logs are generated; thus, log analysis can become difficult especially due to the number of false-positive alerts. Therefore, experts should carefully tune the parameters, and predict the impacts of parameter change on network security.
- **Dealing with volume and variety in log analysis**: In general, an IDPS generates an enormous number of logs, because the system inspects attacks at a very low level (i.e., at the packet level). For example, their IDPS detects about 200,000 logs as attacks within one hour. It is also necessary to take into account the numerical information contained in the alert (e.g., the *attack count*). Therefore, IDPS logs should be handled differently from typical time-series data. We believe a visual analytics tool with effective visual encodings and interactions can play an essential role in the analysis of these large-scale logs.

To address these challenges, we present Hyperion, an interactive visual analytics tool for an IDPS. Through a close collaboration with security experts with more than a decade of experience in an information security company, we designed a system that supports the whole cycle of the security control process with the IDPS. Hyperion provides visualizations that help analysts understand the policies (i.e., sets of rules) currently applied to the networks. Log visualization components enable them to efficiently investigate the target's large-scale IDPS logs. Experts can simulate and compare how different parameter values affect a rule through interactions and visualizations with analytic components.

This paper is organized as follows. We first cover related work in Section 2. After discussing the background of IDPSs and the security domain in Section 3, we elaborate on the data in the target domain and justify major tasks in Section 4. We then describe visualizations and interactions in our design in Section 5. In Section 6, for the evaluation, we report on case studies based on data collected in a real internet environment. Finally, we discuss limitations and future work.

II. RELATED WORK

A significant amount of research on visualization in the security domain has been conducted [13], [23], [33], [38], [39]. In this section, we describe two topics that are most related to our tool: security event visualization and security policy visualization.

A. SECURITY EVENT VISUALIZATION

Among the several types of data sources related to the network security domain, numerous visual analysis studies have been conducted on logs related to security events. Security events generated by security infrastructures, such as an IDPS or a firewall contain labeled data (e.g., allowed or blocked) in addition to the endpoint information (e.g., the source and destination IPs). There are cases (e.g., *NFlowVis* [11], [20]) where only the occurrence of an event was visualized, but most studies actively used the following additional data.

Attack type and **attack name** are the most common types of data that accompany the endpoint information. *Snort* [30], one of the most widely used open-source IDPSs, can categorize events such as *"Denial of Service"* or *"Information Leak"*. This categorical information about a security event gives analysts more detailed information about the event. In *Visual Firewall* [18], Lee *et al.* proposed a visualization showing events on a quad-axis diagram consisting of time, IP information, and corresponding snort rule categories. Koike *et al.* [17] displayed events with pixels on two matrices related to the source IP address information, reflecting the attack type data in the color of the pixel. Alsaleh *et al.* [3] suggested using various forms of visualizations for analyzing PHP intrusion detection system logs, many of which employed attack type data. Zhao *et al.* [40] proposed a tool, *MVSec*, which uses various visualizations together for analyzing heterogeneous logs such as the IDS log along with the event type. *VisAlert* [19], *Avisa* [32], *AlertWheel* [8], *IDSRadar* [41], and *IDSPlanet* [31] uses information about the type of attack based on radial visualization, though their specific shapes and purposes are different.

The security event may contain the **severity** of the identified attack. *NIVA* [27] represents IP addresses as nodes and the relation between the addresses as lines with color encoding the severity of attack in the 3D space. *IDS RainStorm* [2] displayed the events detected from Class B IP addresses during the day as dots which represent their severity.

There are security event visualization systems that mainly utilize **service** (e.g., web, mail) or **protocol** information (e.g., TCP and UDP). *SnortView* [16] displays security events with symbols that represent protocol and service information in a matrix-based visualization. At the same time, the severity of the event is encoded in the color of the symbol. *SpiralView* [5] uses a view composed of a spiral chart and bar charts, displaying alarm type, alarm code, severity, and app category.

The logs generated by the target IDPS in this paper have many more attributes (e.g., attack period, traffic amount, attack count) than the data attributes utilized in existing visualization systems. Therefore, it is necessary to design new visualizations and interactions to help analysts get in-depth understanding of security events by employing the new additional attributes in the analysis process. In this work, we also support rules management because it is essential for operating an IDPS and it significantly affects the quantity and quality of logs an IDPS generates. We designed Hyperion for an integrative analysis of rules and logs, which has been studied in existing work.

B. SECURITY POLICY VISUALIZATION

Besides IDPSs, firewalls and SELinux [1] are also widely used security infrastructures. These systems operate on the basis of predefined rules or policies like IDPSs. Several tools have been developed to visualize rules or policies to help security experts to manage the infrastructures.

PolicyVis [34] presented a visualization tool for showing firewall rules and policies. It uses binary decision diagrams to represent Boolean expressions for segmenting firewall rules and a configurable matrix visualization to analyze data such as source/destination IP addresses and ports. Morrissey and Grinstein [25] introduced an interactive visualization method for showing relations between different rules. This tool includes a rule-set editor allowing users to see the impact of rule changes. Mansmann *et al.* [21] used a hierarchical sunburst visualization and interactive tree views to understand firewall rule sets and object groups. This tool also has an embedded editor to enable simulation of firewall configuration changes. Kim *et al.* [15] proposed a tool for analyzing the firewall rule-set of a six-dimensional space. This system provides an interactive 3D view for checking the overall condition of the rules. Aupetit *et al.* [4] presented a tool that allows Internet service providers to determine the impact of Distributed Reflective Denial of Service attacks on traffic, create rules to mitigate them, and simulate the effects.

SEGrapher [22], a visualization tool for SELinux policy analysis, modeled a policy as a directed graph and visualized it as a kind of node-link diagram called focus-graph. This tool helped users analyze the relationship between objects (e.g., files) and subjects (e.g., processes) of policies using its own clustering algorithm. SPTrack [7] is based on the node-link diagram, and visualizes the criticality level as colored edges according to interactions (e.g., write, signal) allowed by policies. Xu *et al.* [35], [36] suggested using semantic substrates to visualize the key categories (i.e., user, role, domain, type) of the policy in separate spaces. The components of each policy and their relationships were visualized with nodes and edges. They also used an adjacency matrix to solve the visual clutter that could occur in the node-link diagram, and added visualizations of directed paths to the matrix to aid analysis. V3SPA [12] proposed a method to help analyze the differences between the two policies based on the node-link diagram.

Besides, several techniques for visualizing other security policies using tree maps [28] or grids [29] have been studied.

As can be seen from the above, most of the studies related to the visualization of security policy has been conducted mainly on systems for access control. However, since our target IDPS rule includes attack pattern data (e.g., attack count or period) in addition to access control, new methods are required. Therefore, we propose techniques that fit IDPS rules and meet practitioners' goals. To the best of our knowledge, Hyperion is the first visual analytic system that can help users analyze and manage IDPS rules and logs together. It also targets all attacks in an enterprise scale that the IDPS responds to, not just predefined specific attacks. Through Hyperion, the analyst can grasp the impact of the rule modification on a detailed level, such as a change in IP distribution as well as the amount of traffic.

III. TASKS AND DATA

As mentioned earlier, the IDPS has rules for responding to a large number of known attacks, and the rules have to be managed efficiently to maintain network security of an enterprise. In addition, our target IDPS uses a more sophisticated structure suitable for use in a real large-scale enterprise environment. We explain the tasks and data for IDPS operations.

A. TASKS

When constructing an IDPS as one of the security infrastructures, security experts set up rules and parameters. The experts then collect and analyze logs, and finally, revise the rules as needed. This process is iterative and is the main task of a security control department. We define them as the *security control cycle*. It consists of the following four phases:

1) **Application of rules**: The rules for already known attacks are set for the IDPS. The parameters are tuned according to the structure of the target network and the experts' previous knowledge and experience.

2) **Information collection**: In this phase, the IDPS collects logs according to the rule set in the first phase. Security experts monitor the collected logs. If the experts decide that the current situation is abnormal (e.g., a specific attack is detected or the volume of the logs increases abruptly), they must proceed to the next phase.

3) **Log analysis**: Through analyzing the collected logs, the experts can confirm if they should modify a countermeasure (i.e., *block* or *ignore*) and parameters (e.g., *acceptance count*) of rules for specific attacks. For example, if too many alerts (i.e., logs) are generated for a particular attack and are judged to be excessive or to be false-positive alerts, then the experts may consider updating the rule for that attack.

4) **Rule revision**: The experts revise the corresponding rule, for example, by decreasing its *acceptance count*. After revising the rule, the experts activate the rule (i.e., return to the first phase).

B. RULE AND TEMPLATE

An IDPS inspects packets using their own *signatures*, and processes the classified packets according to the corresponding rules. An IDPS rule of the target IDPS has key-value pairs related to each network threat. Although the rule has many fields, we summarize its most important fields as follows:

- **Attack ID**: Define the unique ID of an attack. The ID consists of the *attack type* and the *code* (i.e., *serial number*). The target IDPS categorizes the attacks into the following types: *Denial of Service, Information Gathering, Pattern Block, Protocol Vulnerability, RegEx, Service Attack, and Web CGI Attack.*
- **Countermeasure**: Specify the method against the attack: do nothing (*ignore*), log the attack (*detect*), and log and block the attack (*block*).
- **Thresholds**: Set the criteria for judging whether an intrusion attempt is malicious. There are three fields corresponding to this category: *acceptance (attack) count, acceptance (attack) time, and block count*. The *acceptance count and time* are the criteria for the minimum number and time to respond to an attack detected by IDPS. The *block count* means a threshold value for IDPS to block the source of the attack.

Table 1 shows an IDPS rule for *TCP SYN flooding attack* and the descriptions of the fields. For example, if the IDPS recognizes an intrusion attempt as having 180 attack counts and 10 seconds, this attempt, although not exceeding the *acceptance count* (200), is recorded in the IDPS log (i.e., detected) because the attempt exceeded the *acceptance time* (4 s), but is not blocked because it does not exceed the *block count* threshold (200). If the attack count exceeds 200, the source of the attack is blocked and log related to the block is recorded.

TABLE 1. A sample IDPS rule.

Key	Value	Description
Template name	DEFAULT	Name of the template to which the current rule belongs
Attack type	Denial of Service	Type of attack that the current rule will detect
Attack name	TCP SYN Flooding	Name of attack that the current rule will detect
Attack code	0	Unique number of the current attack name in the attack type set
Action	Detect	Countermeasure if the attack is detected
Acceptance count	200	Threshold count against which this attack is logged
Block count	200	Threshold count against which this attack is blocked
Acceptance time	4	Threshold time (sec.) against which this attack is logged

As mentioned in Introduction, depending on the network environment and the role of the network entity, the parameters of the rule may be set differently. To this end, the target IDPS groups rules into a *template*, and that is then applied to the network entity. The *DEFAULT* template contains all the rules with the parameters taking into account the general situation. If customization is needed for a particular situation, experts create a new template for it and include only the relevant rules with the customized parameters. From now on, for the network entity to which the new template is applied, the rules in that template take precedence. This method can help experts use multiple rules simultaneously for specific situations. In addition, experts can set different templates for inbound and outbound traffics for more sophisticated monitoring.

C. FIELDS IN THE LOG

According to the rule definitions mentioned above, alerts (i.e., detected or blocked attacks) are recorded as IDPS logs. The fields of the log data are similar to those in the corresponding rule. Table 2 shows an IDPS alert (log) for *UDP flooding attack*. Our IDPS alert also includes additional information such as severity or protocol, which is commonly used by other IDPSs. We interviewed two domain experts in the IDPS development team and found that the following information needs to be analyzed:

TABLE 2. A sample IDPS alert (log).

Key	Value
Attack Period	14:26:15-14:26:15
Attack type	Denial of Service
Attack name	UDP Flooding
Source Country	South Korea
Source IP	116.x.y.z
Destination Country	South Korea
Destination IP	49.x.y.z
Attack Count	140
Traffic Amount	206KB

- **Attack period**: Unlike typical time-series data, the target IDPS log is recorded as an interval rather than as a point. When the IDPS detects an attack, it records the start and end times to provide the interval information. This time-related information tends to be very different depending on the attack type. For example, the *Information Gathering* network attacks tend to continue for a longer period, while the *Denial of Service* attacks are usually logged within a very short period.
- **Endpoint information**: This field covers the information about the source and destination of an attack. The endpoint information includes the IP address, port, and country of the attack. The country data of the source can help select IP ranges to block when attacks such as *Denial of Service* come from a specific country. The information of the destination can be used to identify the target of the current attack. Especially, the destination port can be a crucial clue for analyzing the logs.
- **Aspect of attack**: The target IDPS records the aspect of network attacks in two features: *traffic amount* and *attack count*. The *traffic amount* is used to measure the load that the attack places on the network. The *attack count* shows the continuity and strength of an attack and is directly related to the *acceptance count* of the rule. Even if the attacks are of the same type, the aspects vary depending on the specific attack method (i.e., attack name) or the attacker.

IV. DOMAIN REQUIREMENTS

The goal of Hyperion is to facilitate rule management and log analysis to help security experts operate the IDPS. This is also closely related to the security control cycle mentioned earlier. We collaborated with two security experts at the IDPS development company for about half a year. About once a month we had meetings, including initial semi-structured interviews, to discover their problems encountered in the real field [10], and received feedbacks about our prototypes. Considering the IDPS's operational challenges mentioned in Introduction and through the meetings with domain experts, we extracted the specific requirements for a visual analytics system for the IDPS.

R1. Recognize the relationship between templates and rules. As mentioned previously, the target IDPS handles rules through templates for flexible application and management of large numbers of rules. For efficient operation of templates, it is necessary to be able to easily understand what rules each template has and what parameters of the rules are changed.

R2. Identify templates that apply to a network entity. In the real enterprise environments, the structure of the network that the IDPS is monitoring can be complex and have many entities. Considering this situation, users need to efficiently comprehend which templates are applied to each network entity.

R3. Support comprehensive log analysis. Because the number of logs (i.e., alert count) is a fundamental factor in security log analytics, it is important to help experts perceive the changes in the number of logs in their analytics process. In addition, the target IDPS's logs contain fields about attack patterns: *attack count, traffic amount*, and *attack period* (i.e., duration). Our experts suggested we should develop methods to collectively observe and understand all the fields. Comprehensive understanding of the attack patterns can help experts recognize attacks with similar patterns and discriminate between false and true positives. They also wanted to use intuitive statistical data; for example, statistics on categorical data, such as log count for each attack name, can be very helpful for early analysis.

R4. Detect outlier logs using data analysis techniques. For most types of attacks, logs with a different pattern from usual can be of interest to analysts. To find an outlier in the log, the experts wanted to exploit statistical methods or data analysis algorithms, such as clustering. Obtaining clues through such analytic techniques enables more efficient analysis when analyzing a large volume of logs, and helps identify true positive attacks [41]. The experts also hoped to use their experience and knowledge in outlier decisions. In other words, they needed an interface that could not only find out outliers determined according to predefined parameters, but also change the parameters for the outlier detection.

R5. Simulate the effects of rule parameter changes. As mentioned above, modifying the rule parameters is an important task for domain experts and should be done carefully. To this end, the effects of parameter changes should be intuitively perceived. This should include not only numerical changes such as the number of logs, but also the loss of information such as IP data.

V. DESIGN OF HYPERION

This section ties Hyperion's features to domain requirements by referencing them in parentheses. To satisfy the domain requirements and support the security control cycle, we propose a three-stage analysis model (Rule & Network Exploration, Summary Log Analysis, and then Detailed Log Analysis). Based on the three-stage analysis model, we designed Hyperion so that users can intuitively use it for effective rule management and log analysis.

A. STAGE 1: RULE & NETWORK EXPLORATION

The first stage starts with the **Rule & Network** tab (Figure 1) consisting of the following three views: a template-rule relation matrix, a rule table, and a network-policy map.

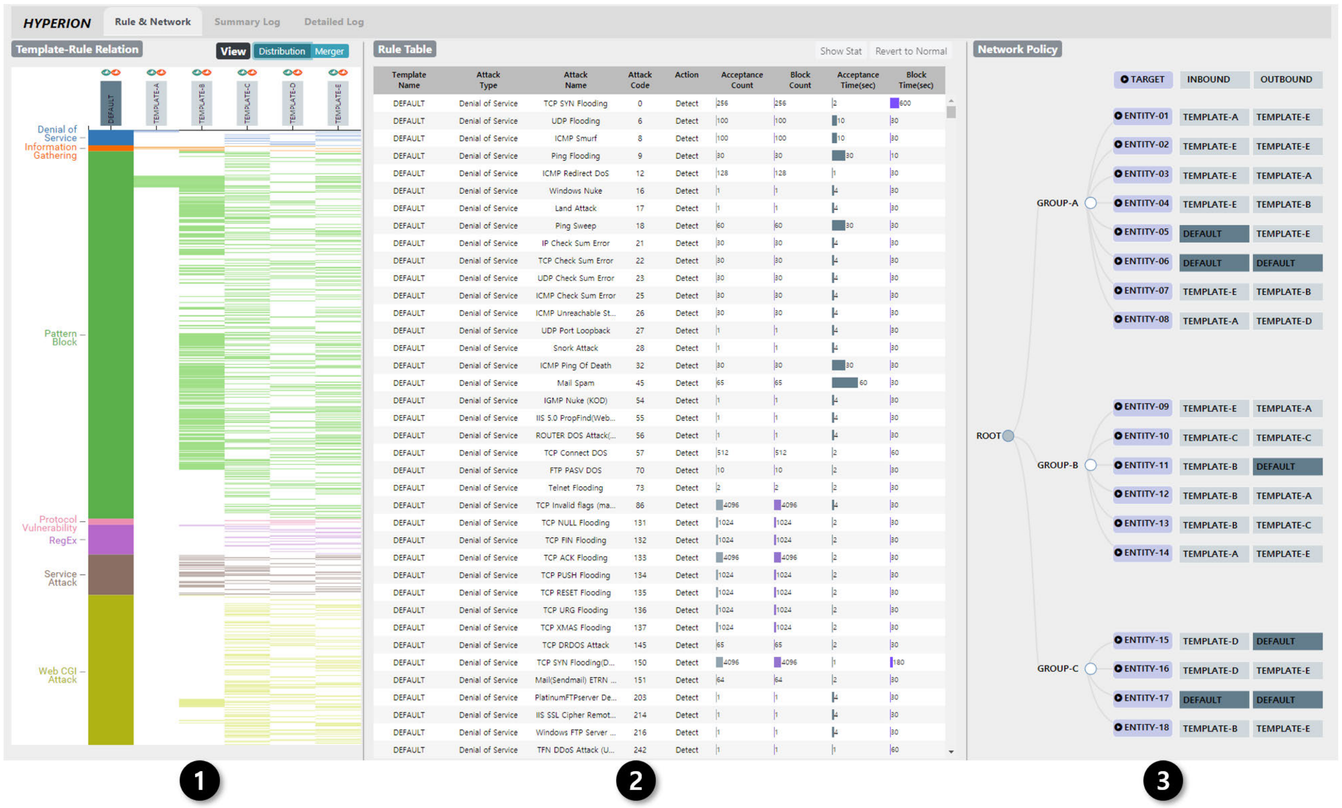
1) TEMPLATE-RULE RELATION MATRIX

Our IDPS can assign a name to a template, but because it can contain rules for several attack types, it may not be enough to figure out the state of the template. Considering that the relationship between template and rule is similar to the relationship between set and element, we explored various views such as a node-link diagram or a tree-map. Finally, we designed a template-rule relation matrix (Figure 1-1) to understand the status of the rules in the template **(R1)** considering that the attack type, which is an important attribute for understanding the state of the template, is limited to within a few, and that it cannot allocate a lot of space due to the large number of rules to grasp. First, the columns in this matrix represent the templates currently held by the IDPS. Considering the customized template inherits the rules from the **DEFAULT** template, we plotted the **DEFAULT** template on the left and then enumerated the customized templates to its right.

Next, the rows in the matrix represent the rules held by the IDPS. Rows are arranged based on the attack type and the attack code field of each rule, and different colors are assigned to help distinguish the attack type. When the user hovers the mouse over each cell of the template, a tooltip tells which rule corresponds to the cell. According to this encoding, the **DEFAULT** template that holds all rules is shown as boxes in which all the rows are filled, and the custom templates that hold only the changed rules are colored only in some rows. For example, in the top of Figure 2, we can see that **TEMPLATE-A** customizes some rules for *Denial of Service* and *Information Gathering*.

2) RULE TABLE

Although the template-rule relation matrix is useful for browsing the rule hold status of a template, there is a need for a way to directly look at the fields of the rule. At first, we tried



Rule Table — Show Stat — Revert to Normal

Template Name	Attack Type	Attack Name	Attack Code	Action	Acceptance Count	Block Count	Acceptance Time(sec)	Block Time(sec)
DEFAULT	Denial of Service	TCP SYN Flooding	0	Detect	256	256	2	600
DEFAULT	Denial of Service	UDP Flooding	6	Detect	100	100	10	30
DEFAULT	Denial of Service	ICMP Smurf	8	Detect	100	100	10	30
DEFAULT	Denial of Service	Ping Flooding	9	Detect	30	30	30	10
DEFAULT	Denial of Service	ICMP Redirect DoS	12	Detect	128	128	1	30
DEFAULT	Denial of Service	Windows Nuke	16	Detect	1	1	4	30
DEFAULT	Denial of Service	Land Attack	17	Detect	1	1	4	30
DEFAULT	Denial of Service	Ping Sweep	18	Detect	60	60	30	30
DEFAULT	Denial of Service	IP Check Sum Error	21	Detect	30	30	4	30
DEFAULT	Denial of Service	TCP Check Sum Error	22	Detect	30	30	4	30
DEFAULT	Denial of Service	UDP Check Sum Error	23	Detect	30	30	4	30
DEFAULT	Denial of Service	ICMP Check Sum Error	25	Detect	30	30	4	30
DEFAULT	Denial of Service	ICMP Unreachable St...	26	Detect	30	30	4	30
DEFAULT	Denial of Service	UDP Port Loopback	27	Detect	1	1	4	30
DEFAULT	Denial of Service	Snork Attack	28	Detect	1	1	4	30
DEFAULT	Denial of Service	ICMP Ping Of Death	32	Detect	30	30	30	30
DEFAULT	Denial of Service	Mail Spam	45	Detect	65	65	60	30
DEFAULT	Denial of Service	IGMP Nuke (KOD)	54	Detect	1	1	4	30
DEFAULT	Denial of Service	IIS 5.0 PropFind(Web...	55	Detect	1	1	4	30
DEFAULT	Denial of Service	ROUTER DOS Attack(...	56	Detect	1	1	4	30
DEFAULT	Denial of Service	TCP Connect DOS	57	Detect	512	512	2	60
DEFAULT	Denial of Service	FTP PASV DOS	70	Detect	10	10	2	30
DEFAULT	Denial of Service	Telnet Flooding	73	Detect	2	2	2	30
DEFAULT	Denial of Service	TCP Invalid flags (ma...	86	Detect	4096	4096	4	30
DEFAULT	Denial of Service	TCP NULL Flooding	131	Detect	1024	1024	2	30
DEFAULT	Denial of Service	TCP FIN Flooding	132	Detect	1024	1024	2	30
DEFAULT	Denial of Service	TCP ACK Flooding	133	Detect	4096	4096	2	30
DEFAULT	Denial of Service	TCP PUSH Flooding	134	Detect	1024	1024	2	30
DEFAULT	Denial of Service	TCP RESET Flooding	135	Detect	1024	1024	2	30
DEFAULT	Denial of Service	TCP URG Flooding	136	Detect	1024	1024	2	30
DEFAULT	Denial of Service	TCP XMAS Flooding	137	Detect	1024	1024	2	30
DEFAULT	Denial of Service	TCP DRDOS Attack	145	Detect	65	65	2	30
DEFAULT	Denial of Service	TCP SYN Flooding(D...	150	Detect	4096	4096	1	180
DEFAULT	Denial of Service	Mail(Sendmail) ETRN ...	151	Detect	64	64	2	30
DEFAULT	Denial of Service	PlatinumFTPserver De...	203	Detect	1	1	4	30
DEFAULT	Denial of Service	IIS SSL Cipher Remot...	214	Detect	1	1	4	30
DEFAULT	Denial of Service	Windows FTP Server ...	216	Detect	1	1	4	30
DEFAULT	Denial of Service	TFN DDoS Attack (U...	242	Detect	1	1	1	60

TARGET	INBOUND	OUTBOUND
ENTITY-01	TEMPLATE-A	TEMPLATE-E
ENTITY-02	TEMPLATE-E	TEMPLATE-E
ENTITY-03	TEMPLATE-E	TEMPLATE-A
ENTITY-04	TEMPLATE-E	TEMPLATE-B
ENTITY-05	DEFAULT	TEMPLATE-E
ENTITY-06	DEFAULT	DEFAULT
ENTITY-07	TEMPLATE-E	TEMPLATE-B
ENTITY-08	TEMPLATE-A	TEMPLATE-D
ENTITY-09	TEMPLATE-E	TEMPLATE-A
ENTITY-10	TEMPLATE-C	TEMPLATE-C
ENTITY-11	TEMPLATE-B	DEFAULT
ENTITY-12	TEMPLATE-B	TEMPLATE-A
ENTITY-13	TEMPLATE-B	TEMPLATE-C
ENTITY-14	TEMPLATE-A	TEMPLATE-E
ENTITY-15	TEMPLATE-D	DEFAULT
ENTITY-16	TEMPLATE-D	TEMPLATE-E
ENTITY-17	DEFAULT	DEFAULT
ENTITY-18	TEMPLATE-B	TEMPLATE-E

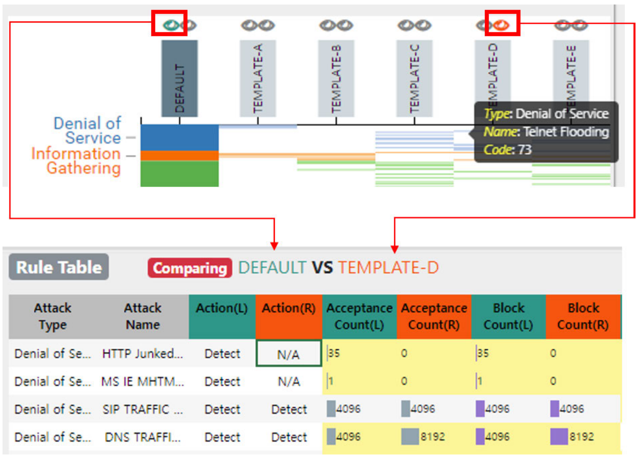
FIGURE 1. **Stage 1: Rule & Network exploration. ① Template-rule relation matrix helps to understand the status of the rules in the template considering that the attack type. ② Rule table allows users to directly look at the fields of the rule using tabular forms and a bars. ③ Network-policy map helps analysts identify the templates currently applied to network entities with a tree diagram.**

to express numeric data using a heat-map, as there are about 4,000 rules and a number of significant fields, we determined that it would be appropriate to show the differences in numerical data as shown in Figure 1-2, using tabular forms and bars rather than other forms of visualization. In the numerical data field, a bar is visualized so that relative differences can be compared. The length of the bar is relative to each column. For example, in the acceptance time field, as shown in the bottom of Figure 2, the value of the **acceptance count** and **block count** fields of *DNS TRAFFIC Flooding* are larger than those of the other attacks.

When the user clicks a row in the matrix, the corresponding rule appears directly in the rule table. Using this, the user can check exactly which rule is customized. To allow more direct template comparisons, the user can switch to the template comparison mode. When the user clicks on the eye icon above each template in the template-rule relation matrix, as shown in Figure 2, the table switches to the comparison mode. In this mode, the fields of the rule with a difference between the two templates have a yellow background, and the other fields have a white background.

3) NETWORK-POLICY MAP

We designed a network-policy map to help analysts identify the templates currently applied to network entities, along

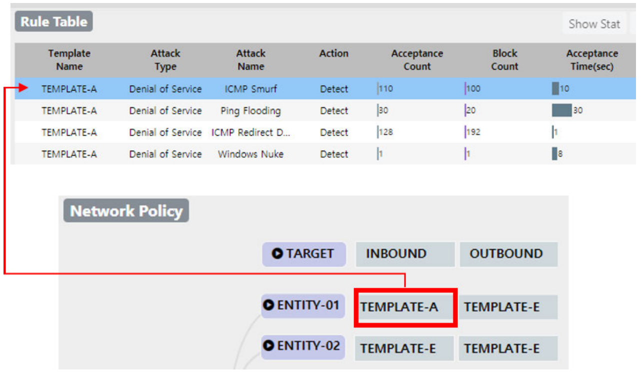


Rule Table — Comparing DEFAULT VS TEMPLATE-D

Attack Type	Attack Name	Action(L)	Action(R)	Acceptance Count(L)	Acceptance Count(R)	Block Count(L)	Block Count(R)
Denial of Se...	HTTP Junked...	Detect	N/A	35	0	35	0
Denial of Se...	MS IE MHTM...	Detect	N/A	1	0	1	0
Denial of Se...	SIP TRAFFIC ...	Detect	Detect	4096	4096	4096	4096
Denial of Se...	DNS TRAFFI...	Detect	Detect	4096	8192	4096	8192

FIGURE 2. **Template-rule relation matrix (top) and rule table (bottom) in the comparison mode. When the analyst clicks each of the eye icons on the template, the rule table switches to the comparison mode.**

with the structure of the network that the IDPS is monitoring (Figure 1-3) **(R2)**. Our experts demanded that the structure of the network be consistently observed and the application status of the templates comparable. One "group" means an organization such as a company or a school, and may have multiple sub-network. Our IDPS utilizes a range for the network address, not the individual terminal, to which a rule applies. This address range is treated as a network "entity". For example, a network entity named **ENTITY-X** could have an address range of 10.x.y.z/24 or 10.x.y.z/16. At the beginning of the project, we proposed a force-directed graph as a prototype. However, we discovered difficulties in understanding the network structure when the number of network entities was larger and in comparing the template set in the entity with other entities. Therefore, we chose a tree diagram to visualize the network policy. The tree shape of our target network structure makes understanding the structure intuitive. In addition, the tree diagram helps experts figure out the status of the templates. This view is also linked to the rule table. When the user clicks on the template in the network-policy map, the rules of the corresponding template can be viewed directly in the rule table (Figure 3).


Rule Table

Show Stat

Template Name	Attack Type	Attack Name	Action	Acceptance Count	Block Count	Acceptance Time(sec)
TEMPLATE-A	Denial of Service	ICMP Smurf	Detect	110	100	10
TEMPLATE-A	Denial of Service	Ping Flooding	Detect	30	20	30
TEMPLATE-A	Denial of Service	ICMP Redirect D...	Detect	128	192	1
TEMPLATE-A	Denial of Service	Windows Nuke	Detect	1	1	8

FIGURE 3. Rule table (top) and network-policy map (bottom). When the analyst clicks a template assigned to each network entity in the network-policy map, he or she can immediately browse the template in the rule table.

B. STAGE 2: SUMMARY LOG ANALYSIS

The user can check rules, templates, and network structure through the three views in the **Rule & Network** tab, and then start the log analysis. When the user selects the network entity from the network-policy map, the screen is switched to the second stage, the **Summary Log** tab (Figure 4). This tab contains three views: a top-10 data view, a source country map, and a timeline view.

1) TOP-10 DATA VIEW

The top-10 data view deals with ranking data for four items that our experts are interested in, as mentioned in sections 3.2 and 3.3: the **attack name, source IP, destination IP,** and **destination port**. As the ranking of items can be based on each of the four important fields of numeric log data (i.e., **alert count**, **attack count**, **traffic amount**, and **attack duration**), it is necessary to be able to change the sorting criteria field according to the user's needs and observe all the field values at the same time (**R3**).

In our first version, we used a grouped bar chart consisting of four bars for each element, but we found the disadvantage that data interpretation may be confusing and the comparison is not easy because the four fields have different units. Therefore, we suggested a form of radar chart where the four fields can be constructed as independent axes, and the visualization results appear as glyphs, making it easier to compare (Figure 4-1). The user can change the sorting criteria by clicking on the field name in the legend at the top right of the view, and click on a radar to see the exact numeric data as a tooltip (Figure 5).

2) SOURCE COUNTRY VIEW

The source country view, located at the bottom left of the screen, shows the country data statistics from the source information of the logs as a world map and a barchart (Figure 4-2). As mentioned in section 3.3, source country information is important data that can identify some types of attack characteristics. We used a world map to design the source country view because a map-based view can help find attack patterns [24] and our domain experts also strongly advised the need for visualization in the form of a world map. The statistics are displayed in gray and black heatmap colors to aid relative comparisons. When the user clicks the bar, the country is selected as the filtering item.

3) TIMELINE VIEW

The last view is the timeline view that allows analysts to observe the four numeric log data according to the timeline (Figure 4-3). The additional data covered in this view are the **attack type**. As the impact on the four numeric log data is generally different for each attack type, understanding the timeline of the data for the attack type can help to select a candidate of the detailed analysis step. For example, if a part of the timeline shows a similar flow between different attack types, it may be the result of the same attack being detected simultaneously by different rules.

This view shows the timeline of each numeric log data in two ways: a stacked area chart and a juxtaposed filled area chart (Figure 11a). This is because the purpose of this view is to allow the user to search for ranges that have a distinct slope or characteristics in each data timeline [14], [26]. In addition, for a more accurate analysis, the user can choose between the linear scale and the log scale in the stacked area chart, and select whether to use the range of data values identically or individually for each attack type in the juxtaposed filled area chart.

C. STAGE 3: DETAILED LOG ANALYSIS

The user switches to the **Detailed Log** tab, the third stage, by brushing the time area to be analyzed in detail in the timeline view and clicking the "Detail" button located at the top of the view. At this time, the contents of the work (i.e., filtering) performed by the user at the previous stage are maintained at the top of the screen, thus helping the detailed log analysis [6]. This tab provides the user with three views: a raw data table, a related rule table, and a detailed timeline view.

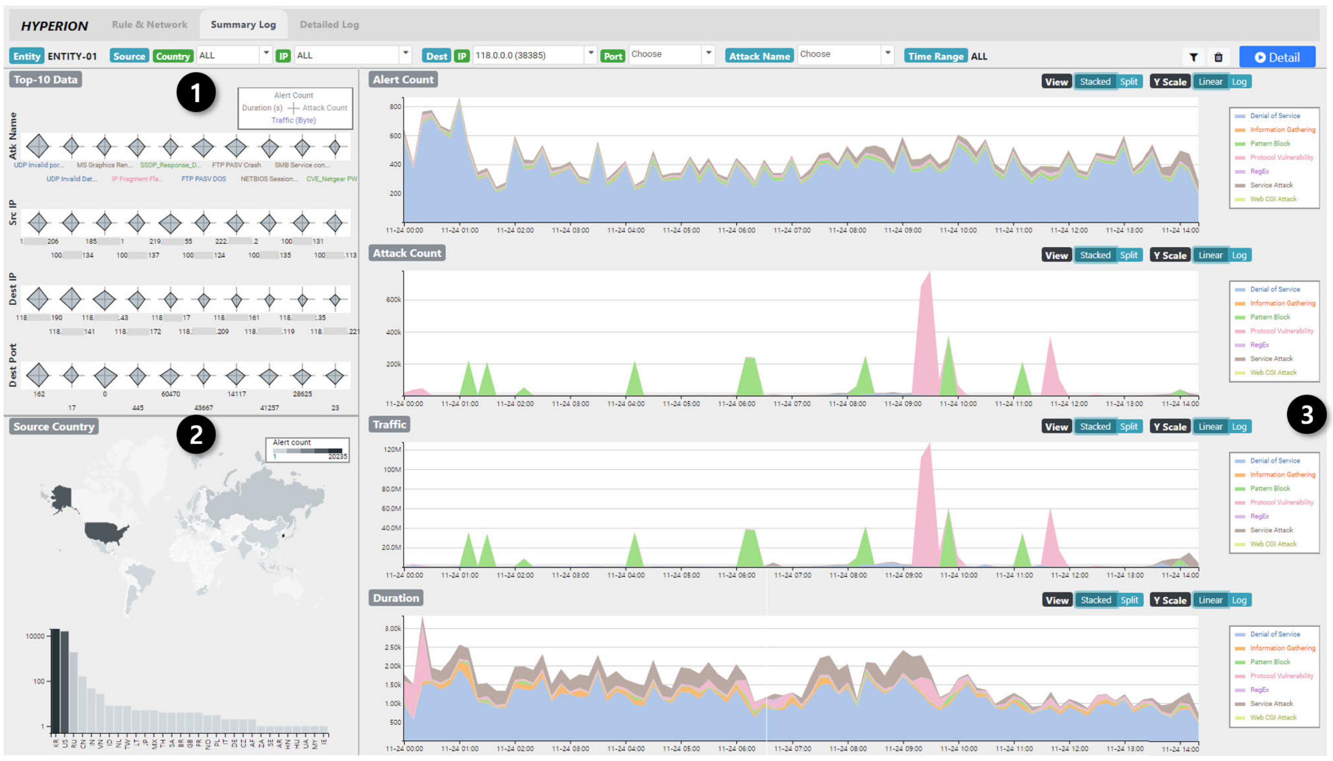


FIGURE 4. Stage 2: Summary Log analysis. ① Top-10 data view shows ranking data for four items (attack name, source IP, destination IP, and destination port) with a form of radar chart. ② Source country view shows the country data statistics from the source information of the logs as a world map and a barchart. ③ Timeline view allows analysts to observe the four numeric log data (alert count, attack count, traffic amount, and attack duration) using a stacked area chart and a juxtaposed filled area chart.

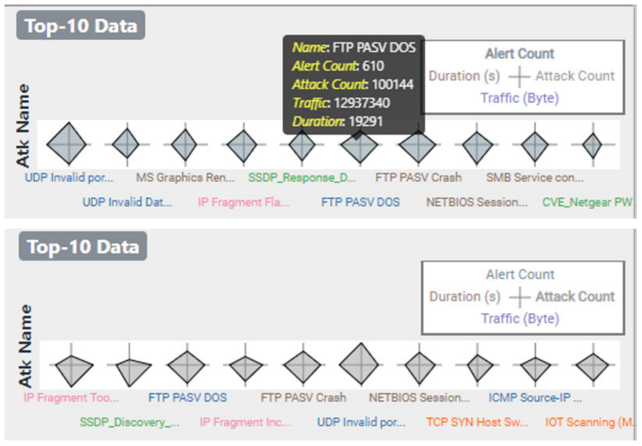


FIGURE 5. Top-10 data view. The top is sorted by the alert count, and the bottom is sorted by the attack count.

1) RAW DATA TABLE

The raw data table (Figure 6-1) provides two functions. The first shows raw data of selected logs in a detailed timeline view to be explained later. These logs are visualized in the form of radar charts, as in the top-10 data view, and can be viewed by clicking the Top-10 button at the top left of the view.

The other function is to extract outlier logs based on attack pattern data (**attack count, traffic amount, and attack duration**) in each log (**R4**). Through discussions with experts, we decided to use our own metric to determine the outlier considering several factors:

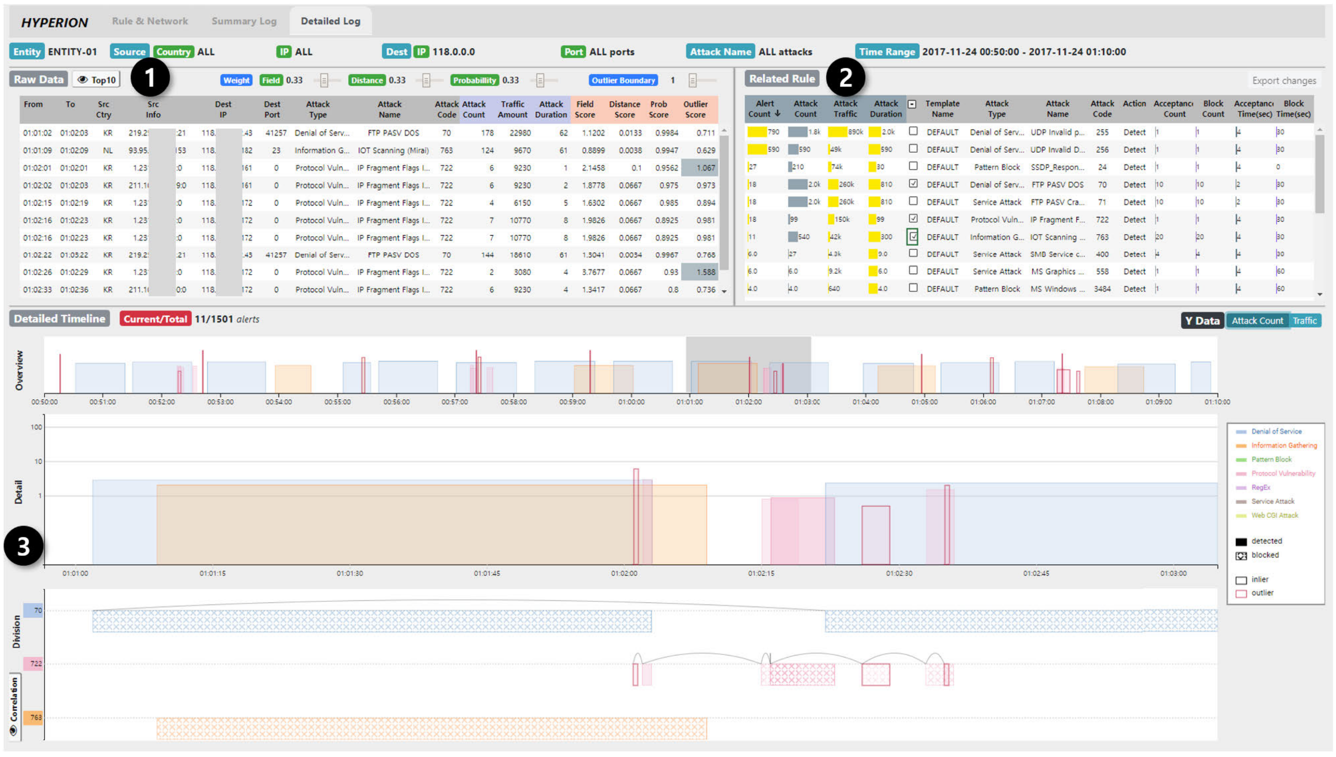
(1) Out target IDPS's logs are all considered malicious by the rules. In other words, the logs are difficult to see as labeled data for supervised analysis. Therefore, we decided to use an unsupervised analysis method [9].

(2) A log with an unusual attack pattern is likely to be an outlier. This unusual pattern can be determined by two aspects: the first case is when the field value itself is very large or small compared to the log of the same attack; and the second case is when the field values are very different from those of other logs.

(3) Due to the nature of the IDPS operating on a pre-defined rule base, the possibility of false positives should be considered. For example, repeatedly found logs that have the same attack patterns are likely to be false positives [16], [41]. As a result, the frequency of the specific patterns can be used as a metric.

(4) It is better to indicate the outlierness with a score rather than a binary label (inlier or outlier). This is because it can show a clue with which the user can judge the outlier detection result.

(5) The analyst should be able to intervene in the outlier detection. That is, adjustable parameters that can reflect the knowledge and experience of the analyst should be



Entity ENTITY-01 Source Country ALL IP ALL Dest IP 118.0.0.0 Port ALL ports Attack Name ALL attacks Time Range 2017-11-24 00:50:00 - 2017-11-24 01:10:00

Weight Field 0.33 Distance 0.33 Probabillity 0.33 Outlier Boundary 1

From	To	Src Ctry	Src Info	Dest IP	Dest Port	Attack Type	Attack Name	Attack Code	Attack Count	Traffic Amount	Attack Duration	Field Score	Distance Score	Prob Score	Outlier Score
01:01:02	01:02:03	KR	219.2[redacted]:21	118.[redacted].43	41257	Denial of Serv...	FTP PASV DOS	70	178	22980	62	1.1202	0.0133	0.9984	0.711
01:01:09	01:02:09	NL	93.95.[redacted]153	118.[redacted]182	23	Information G...	IOT Scanning (Mirai)	763	124	9670	61	0.8899	0.0038	0.9947	0.629
01:02:01	01:02:01	KR	1.23[redacted]:0	118.[redacted]161	0	Protocol Vuln...	IP Fragment Flags I...	722	6	9230	1	2.1458	0.1	0.9562	1.067
01:02:02	01:02:03	KR	211.1[redacted]9:0	118.[redacted]161	0	Protocol Vuln...	IP Fragment Flags I...	722	6	9230	2	1.8778	0.0667	0.975	0.973
01:02:15	01:02:19	KR	1.23[redacted]:0	118.[redacted]172	0	Protocol Vuln...	IP Fragment Flags I...	722	4	6150	5	1.6302	0.0667	0.985	0.894
01:02:16	01:02:23	KR	1.23[redacted]:0	118.[redacted]172	0	Protocol Vuln...	IP Fragment Flags I...	722	7	10770	8	1.9826	0.0667	0.8925	0.981
01:02:16	01:02:23	KR	1.23[redacted]:0	118.[redacted]172	0	Protocol Vuln...	IP Fragment Flags I...	722	7	10770	8	1.9826	0.0667	0.8925	0.981
01:02:22	01:03:22	KR	219.2[redacted]:21	118.[redacted].45	41257	Denial of Serv...	FTP PASV DOS	70	144	18610	61	1.3041	0.0034	0.9967	0.768
01:02:26	01:02:29	KR	1.23[redacted]:0	118.[redacted]172	0	Protocol Vuln...	IP Fragment Flags I...	722	2	3080	4	3.7677	0.0667	0.93	1.588
01:02:33	01:02:36	KR	211.1[redacted]0:0	118.[redacted]172	0	Protocol Vuln...	IP Fragment Flags I...	722	6	9230	4	1.3417	0.0667	0.8	0.736

Alert Count ↓	Attack Count	Attack Traffic	Attack Duration	☐	Template Name	Attack Type	Attack Name	Attack Code	Action	Acceptance Count	Block Count	Acceptance Time(sec)	Block Time(sec)
790	1.8k	890k	2.0k	☐	DEFAULT	Denial of Serv...	UDP Invalid p...	255	Detect	1	1	4	30
590	590	49k	590	☐	DEFAULT	Denial of Serv...	UDP Invalid D...	256	Detect	1	1	4	30
27	210	74k	30	☐	DEFAULT	Pattern Block	SSDP_Respon...	24	Detect	1	1	4	0
18	2.0k	260k	810	✓	DEFAULT	Denial of Serv...	FTP PASV DOS	70	Detect	10	10	2	30
18	2.0k	260k	810	☐	DEFAULT	Service Attack	FTP PASV Cra...	71	Detect	10	10	2	30
18	99	150k	99	✓	DEFAULT	Protocol Vuln...	IP Fragment F...	722	Detect	1	1	4	30
11	540	42k	300	✓	DEFAULT	Information G...	IOT Scanning ...	763	Detect	20	20	4	30
6.0	27	4.3k	9.0	☐	DEFAULT	Service Attack	SMB Service c...	400	Detect	4	4	4	30
6.0	6.0	9.2k	6.0	☐	DEFAULT	Service Attack	MS Graphics ...	558	Detect	1	1	4	60
4.0	4.0	640	4.0	☐	DEFAULT	Pattern Block	MS Windows ...	3484	Detect	1	1	4	60

Current/Total 11/1501 *alerts*

FIGURE 6. Stage 3: Detailed Log analysis. ① Raw data table shows raw data of selected logs in the detailed timeline view and enables users to extract outlier logs based on attack pattern data (attack count, traffic amount, and attack duration) in each log. ② Related rule table displays only the rules related to the logs in the time range selected in the second stage and allows analysts to edit the parameters of the rule for the rule revision task. ③ Detailed timeline view helps users drill down into the logs with several charts (a building-bar chart, a data correlation chart, and a sub-Gantt chart).

introduced. For example, as mentioned above, some logs that are often detected in the same pattern may be false positives, but they may not be if the logs are about *Denial of Service* attacks. Therefore, the influence of the metrics should be appropriately alterable by analysts.

Considering all the factors above, we created a metric that can calculate the **Outlier Score** for each log. Formally, an outlier score *S(x)* is defined as follows:

$$S(x) = w_0 \cdot Z(x) + w_1 \cdot D(x) + w_2 \cdot P(x)$$

$$Z(x) = \left|\frac{x_{ac} - \mu_{ac}}{\sigma_{ac}}\right| + \left|\frac{x_{ta} - \mu_{ta}}{\sigma_{ta}}\right| + \left|\frac{x_{ad} - \mu_{ad}}{\sigma_{ad}}\right|$$

$$D(x) = \sum_{kNN}^{k} distance(X_k, x)$$

$$P(x) = 1 - \frac{count((x_{ac}, x_{ta}, x_{ad}))}{count(X)},$$

where X is a set of logs for the same attack as specified by the attack type and the attack name, x is an instance (i.e., a log) of X, x_{ac} is x's **attack count**, x_{ta} is x's **traffic amount**, and x_{ad} is x's **attack duration**.

In the equation of the Outlier Score S, the weights $w_0, w_1, w_2 \in [0, 1] (w_0 + w_1 + w_2 = 1)$ balance three terms. The first term (Z), the **Field Score**, indicates the outlierness of the pattern data field's value itself with mean μ and standard deviation σ. The second term (D), the **Distance Score**, estimates how the pattern data field values in a log x differ from those of other logs in the Euclidean distance between the log and K-nearest neighbors. We set K to the number corresponding to 5% of the size of X, taking into account the size distribution of X. This 5% range was determined by expert advice. The third term (P), the **Probability Score**, approximates the occurrence of the log with the identical combination of field values. As mentioned above, the higher the value of all three terms, the higher the outlierness, which leads to an increase in the score variable S.

As shown in Figure 6-1, the terms and scores calculated according to the definition above are presented in the columns on the right side of the log data. The user can recalculate the outlier score by adjusting the slider of the three weights located at the top of the table and change the outlier criterion by adjusting the outlier boundary slider. Logs that are determined to be outliers according to the criterion are grayed in the background color of the outlier score column and displayed with a red border in the detailed timeline view below.

2) RELATED RULE TABLE

The related rule table (Figure 6-2) provides additional functions that display only the rules related to the logs in the time range selected in the second stage. The first function is to show the statistics for the logs detected by each rule in the leftmost column. This can be used as a measure of the usefulness of the rule. The second is a filtering function that allows the user to select the attack type to display in the detailed timeline view with the check-box. The final function is to edit the parameters of the rule for the **rule revision** task **(R5)**. If the user changes the value of the parameter columns, such as the *acceptance count* or *block count*, the relevant results are visualized in the detailed time view.

3) DETAILED TIMELINE VIEW

The detailed timeline view (Figure 6-3) allows the user to drill down into the logs with several charts: a building-bar chart, a data correlation chart, and a sub-Gantt chart. In terms of visual space, this view is composed of three sub-areas: an overview area, a detail area, and a division area from top to bottom.

The building-bar chart, as the main chart, fully utilizes the position, width, and height of the bar to provide a comprehensive analysis of the log's numerical data **(R3)**. Specifically, the width of a bar represents the **attack duration** of the log, and the height expresses the *average* **attack count** or **traffic amount** per second according to the user's choice. Thus, the area of the bar represents the whole attack count or the traffic amount of the log. In addition, because the position of the start and end of the base of the bar indicates the beginning and end of the attack and the bar color is semitransparent, the user can see when the attacks overlap. The hue of the bar represents the attack type, and the border color of the bar denotes whether the corresponding log is an outlier that is determined in the raw data table **(R4)**. With this visual encoding, the chart makes it possible to understand the attack patterns with all numerical data of logs.

If there are too many logs, visual clutter in the building-bar chart may make it difficult to recognize them. To mitigate this problem, the chart provides an overview + detail interface. The user can select the log to be displayed in the detail area in the middle of the detailed timeline view by brushing the overview area at the top of this view, and the raw data of the logs visible in the detail area can be explored in the raw data table. In the division area at the bottom of the view, the user can observe the logs separately for each attack name. Although the attack count and traffic amount information of the attack represented by the height is omitted, it is possible to understand the occurrence and duration for each alert. Because this sub-area eliminates the visual-clutter problem that occurs when the attacks overlap, texture information for each bar is added to indicate whether the attack is only detected or is blocked. In addition, an arc is connected to each alert's starting point to help identify the alert period.

As another way to solve the building-bar chart's visual-clutter problem, the detailed timeline view has the alert-timeline chart of the Gantt chart form, as shown at the top of Figure 7a. When the user double-clicks a bar (i.e., a log) in the detail area, a pop-up window appears that allows the user to view logs in the range of about 5 seconds before and after the corresponding bar. A red line is placed at the start of the selected bar so that the reference can be comprehended.

The data correlation chart (Figure 7b), which is displayed by clicking the button at the lower left of the view, is a visualization that represents the correlation of all visualized logs in the detail area. This chart shows the correlation of the **attack count**, **attack duration**, and **traffic amount** in three scatter plots, considering the domain requirements. In addition, the color of the circles represents the outlier score of the log, which can help to understand the relationship between three numerical data and the degree of the outlier.

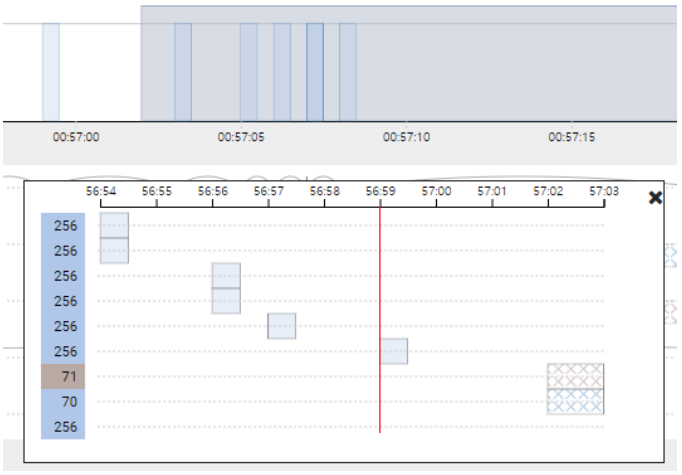
To support the **rule revision** task, this view provides additional interaction and visualization. When parameters are changed in the related rule table, the logs affected by the parameters are highlighted using animation in the building-bar chart. The user can view all original logs, unaffected logs, and affected logs separately with a button set. It also provides the revision result chart as in the form of a bullet chart to recognize statistical data of the effects of parameter changes (Figure 7c). This chart allows the user to identify all the entities in the source IP, source country, destination IP, destination port, the number of the original logs, and the number of logs affected by the parameter. At the top of the chart, statistical values indicating the loss ratio of information are shown. In particular, for IP data, an alignment method is provided that can be grouped together with the similar IP band (e.g., the same C-class). In addition, when the cursor is placed on each IP, the background of the IPs of the same C-class band is highlighted in yellow, thus helping to understand the information loss effectively.

All views in this tab are tightly linked. When a user selects a circle in the data correlation chart, the log is highlighted in the raw data table, the rule that detected the alert is emphasized in the rule table, and a tooltip is displayed on the bar in the detail area.

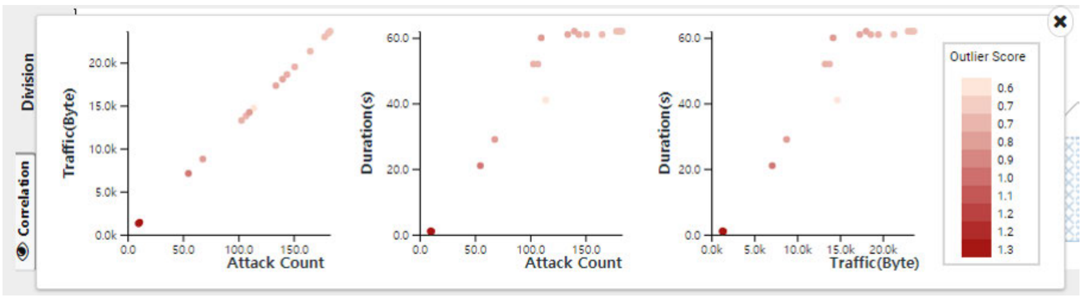
VI. EVALUATION

We evaluated the proposed system via a case study with two security experts who had more than ten years of experience in operating and developing security systems. We used the logs collected for about half a day from an Internet service provider. The number of logs was about 3 million, and the preprocessing took about 20 minutes. The preprocessing includes the computation of the score of the logs and the data aggregation for the timeline view in the summary log analysis stage. After discussions with our experts about the time range of the data and their analysis process, the logs were aggregated into 10-minute-long bins. For anonymity, all IP information and group names (i.e., companies or organizations) were masked. The client of Hyperion was implemented in Typescript with D3.js, Angular, and Bootstrap. The web server was written in Python using the Flask web application framework.

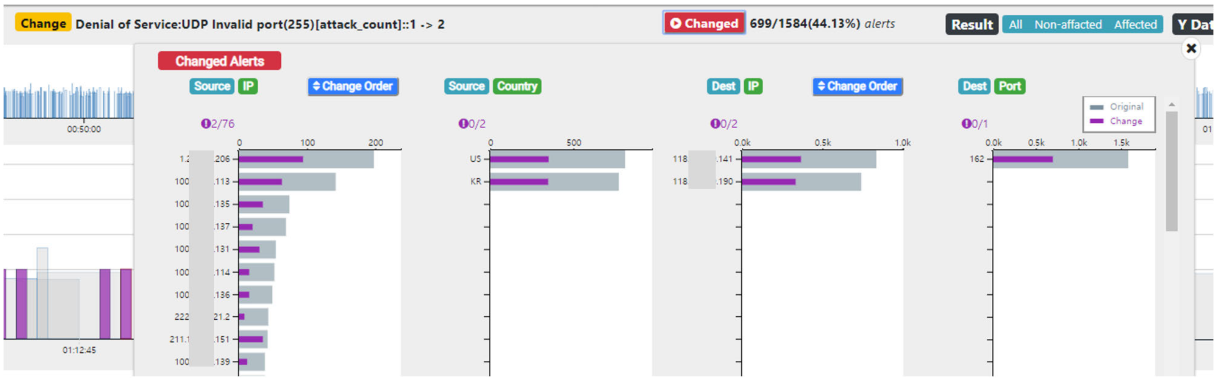
The domain experts used the Hyperion system to analyze the logs together in the same place for about an hour. Before conducting the case study, the authors provided the domain experts with tutorials and demonstrations of how to use the tools. After this, the domain experts took the wheel



(a) Alert-timeline chart. This shows the logs in the form of the Gantt chart that do not overlap.



(b) Data correlation chart. It can be seen that logs with small values in the attack count, attack duration, and traffic amount have high outlier scores (darker red).



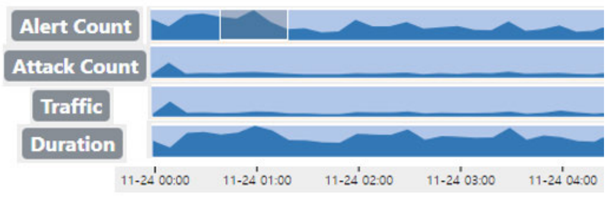
(c) Rule revision result chart. For example, the source country (second), the destination IP (third), and the destination port (fourth) originally had two and one entities, respectively, and no entities disappear due to the revision.

FIGURE 7. **Sub-charts of the detailed timeline view.**

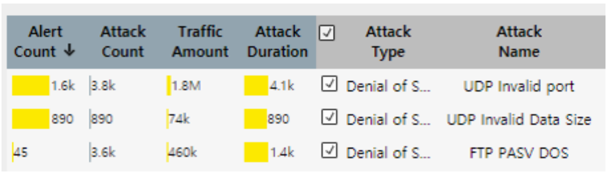
and analyzed the logs using Hyperion. The authors provided guidance on how to use the system when the experts need help using the system. We describe two most meaningful analysis findings that the experts made with the log data.

A. FINDING 1: LEADING TO REVISING RULES FOR ENTITY-01

In the overview of the ENTITY-01 logs, they found that most logs consist of alerts for *Denial of Service* attacks. Exploring the rank of **Attack Name** based on the alert count (i.e., number of logs) in the Top-10 data view, they could see that the first and second attacks were *Denial of Service* (the top of Figure 5). In particular, the number of *UDP Invalid Port* attack logs was about 30,000, which was about 17 times more than that of *UDP Invalid Data Size* attack. Even in the trend of alert count data in the timeline view of the summary logs, most logs were *Denial of Service* attacks. When inspecting the timeline for each attack type in the juxtaposed filled area

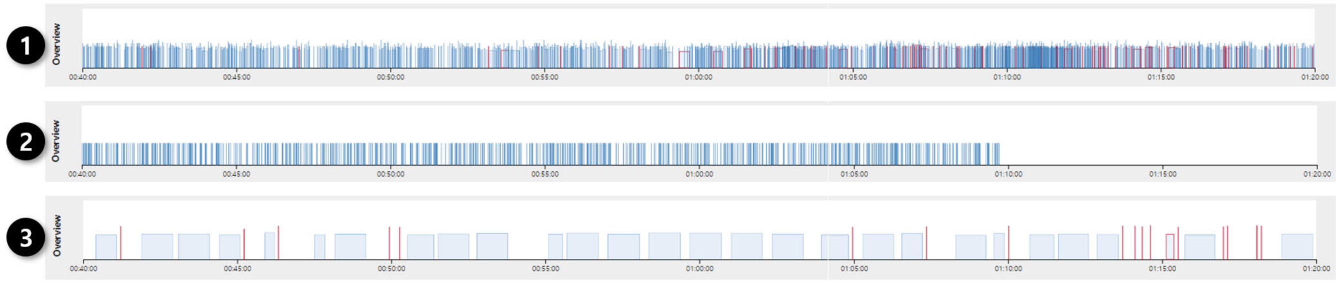


(a) Timelines of each data for *Denial of Service* type. Including the brushed area, there are differences in trends between alert count and duration, attack count and traffic amount.



Alert Count ↓	Attack Count	Traffic Amount	Attack Duration	☑	Attack Type	Attack Name
1.6k	3.8k	1.8M	4.1k	☑	Denial of S...	UDP Invalid port
890	890	74k	890	☑	Denial of S...	UDP Invalid Data Size
45	3.6k	460k	1.4k	☑	Denial of S...	FTP PASV DOS

(b) Statistics of *Denial of Service* attacks from 00:40 to 01:20. Considering the alert count, the *UDP Invalid Data Size* attack has little impact on the attack count and traffic amount.



(c) Building-bar charts for three *Denial of Service* attacks (①-*UDP Invalid port*, ②-*UDP Invalid Data Size*, and ③-*FTP PASV DOS*). The *UDP Invalid Data Size* attack was not caught after 01:10, but the other two attacks were continuously detected after that time.

FIGURE 8. Analysis of ***Denial of Service*** attacks on ENTITY-01.

mode, they found a noticeable slope in the range between 00:40 and 01:20, as shown in Figure 8a. In this time range, the alert count and attack duration data were similar, but the attack count and traffic amount did not change much.

1) FINDING THE CAUSE OF INCONSISTENT SLOPES

When they looked at the statistics of this time range in the related rule table, the top three attacks with the highest number of logs were all *Denial of Service* attacks. At this time, the *UDP Invalid Data Size* attack, which was ranked second, had a relatively small impact on the attack count and traffic amount. In addition, this attack was no longer collected after 01:10, in contrast to other attacks that occurred evenly within the time span. Based on this observation, it was determined that this attack made the inconsistent slope in the timeline view observed in the previous step.

2) RULE REVISION 1: CASE OF UDP INVALID PORT ATTACK

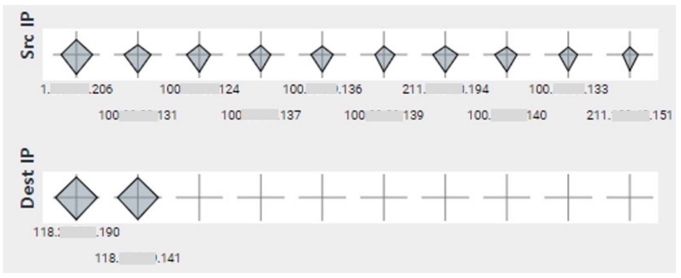
In the Top-10 chart of the raw data table, the *UDP Invalid Port* attack with the largest number of logs shows a typical aspect of *Denial of Service* attack in which multiple sources attack two destinations. Observing this attack on the building-bar chart of the detailed timeline view, they could identify an uneven attack pattern (Figure 8c-①). Looking into the outlier analysis results, they found that the logs with a very short duration, small amount of traffic, and a low attack count had relatively high scores. The logs of such a pattern have a high field score indicating the outlierness of the field itself, but have a small distance score representing the uniqueness of the field combination. Inspecting the probability scores, they concluded that this score was not a useful reference in this attack because the logs have high scores and they have almost the same endpoint information.

Based on these observations, the domain experts posited the following hypothesis: *Even if we increase the acceptance count threshold for the rule of the attack to ignore the logs with extremely low attack counts, it is still possible to deal with the attack while reducing the number of corresponding logs to take care of.*

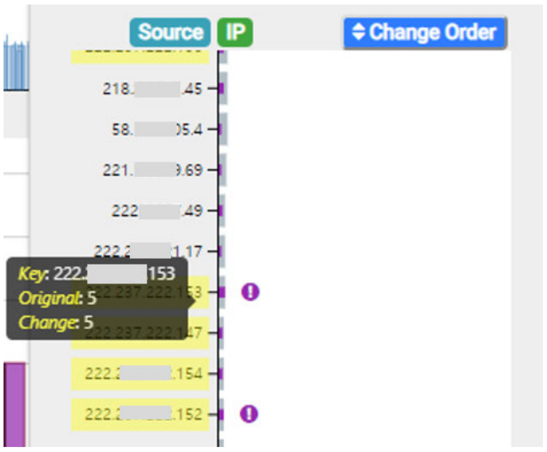
Considering the form of the rule and the current attack pattern with an extreme criterion (i.e., a low attack count), the effect on logging can be expected by simply altering the acceptance count. Changing the rules in this way will obviously reduce the number of logs recorded, which could reduce the amount of resources required for logging and analysis. Prior to making this rule change, it was necessary to confirm that the loss of information due to the change would not affect the overall performance of the security system.

To verify this hypothesis, they used the related rule table to increase the **acceptance count** of the *UDP Invalid Port* attacks from 1 to 2. Due to the change in the rule, 699 logs of the 1,584 logs (about 44%) were eliminated. Through the bullet chart (Figure 7c and Figure 9b), they confirmed that only two of the 76 source IPs disappeared (about 3%), but there was no change in the source or destination country information. Although some source IP information disappeared, it was possible to detect the attack and cope with the attack traffic because all the IP information at the C-class level (i.e., A.B.C.x) still remained.

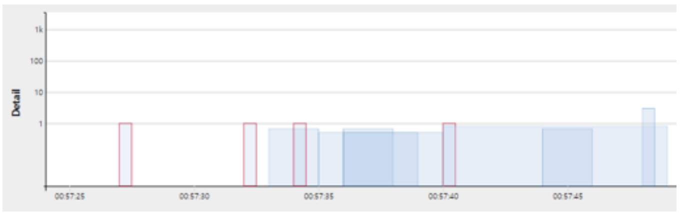
As a result of applying this rule change to the whole log, the number of logs was reduced by about 45% (i.e., 13,625 of the total 30,173 logs), and 9 IPs among 88 source IPs (about 10%), 1 country among 3 source



(a) Top-10 data chart for *UDP Invalid Port* attack. There were multiple source IPs and two destination IPs.



(b) The rule revision caused two IPs of the Source IP to disappear, but this was an acceptable level because the same IP information (yellow background) at the C-Class level remained.



(c) Building-bar chart for *UDP Invalid port*. There were various patterns, but logs with short attack duration and few attack counts appeared as outliers (red border).

FIGURE 9. **Rule revision for *UDP invalid Port Attack* for ENTITY-01.**

countries (about 30%), and 9 IPs among 15 destination IPs (about 60%) were excluded. The domain experts concluded that this rule revision of increasing the **acceptance attack count** against the *UDP Invalid Port* attack was reasonable.

3) RULE REVISION 2: CASE OF FTP PASV DOS AND FTP PASV CRASH ATTACKS

The log of the *FTP PASV DOS* attack, which was ranked the third highest in terms of the number of logs, showed a contrasting pattern. As shown in the first row of the division area in Figure 10, logs with a very low attack count and short attack duration and vice versa are clearly distinguished. However, the *FTP PASV Crash* attack, which was ranked the fourth highest in terms of the number of logs at the same time range as in the previous section, showed a very similar pattern to the *FTP PASV DOS* attack (the second row of the division area in Figure 10). In practice, two attacks work very similarly. In addition, there is no significant difference in the statistical data or the visualization results of the Top-10 chart. Such similarity in log patterns of the two attacks was maintained even when the search range was enlarged to the whole time range. However, the *FTP PASV DOS* rule was found to detect relatively more logs. Considering these points, they suggested checking the signatures of both attacks and using only the rule of the *FTP PASV DOS* attack if there is little difference between them.

B. FINDING 2: CATEGORIZING AND GROUPING ATTACKS FOR ENTITY-06

In ENTITY-06, *Denial of Service, Pattern Block, and Service Attack* types were mainly detected. In the timeline view, the experts found some noteworthy clues: The *Denial of*

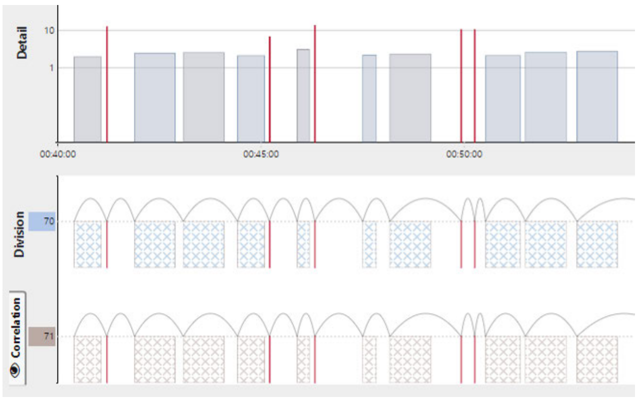
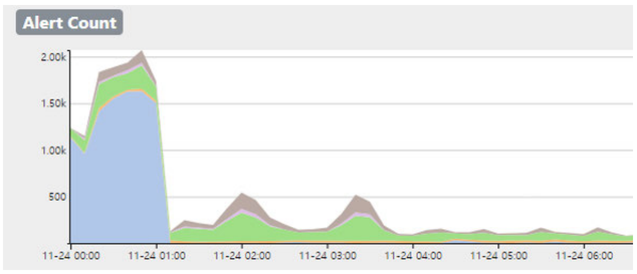


FIGURE 10. ***FTP PASV DOS* attacks (blue, the first row of the division area) and *FTP PASV Crash* attacks (brown, the second row of the division area) had almost the same pattern.**

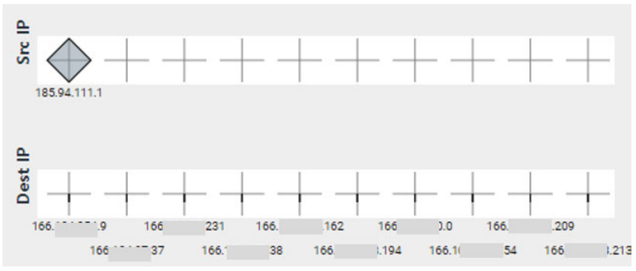
Service attacks were drastically reduced from a certain time point and the other attack types showed broadly similar patterns in the Timeline view.

1) INVESTIGATING OF DENIAL OF SERVICE ATTACKS FROM VARIOUS PERSPECTIVES

The timeline for the alert count in the summary log analysis stage shows that the *Denial of Service* attack, which occupied large areas from the start, dropped sharply from around 01:10 (Figure 11a). At the detailed analysis stage of the range from 00:00 to 01:20, they found that four types of *Denial of Service* attacks were performed, 99% of which were identified as *UDP Invalid Data Size* attacks. The attack continued from 00:00 to 01:09 and one source IP attacked



(a) The number of *Denial of Service* attacks (blue) dropped sharply from around 01:10.



(b) For about 20 minutes, one source attempted to attack multiple destinations.

FIGURE 11. Attack log analysis for ENTITY-06.

multiple destination IPs (Figure 11b). Multiple targets were attacked simultaneously and all attacks were executed in the same pattern in terms of **attack count, traffic amount, and attack duration**.

The source IP was Russian and the destination IPs were at a Korean university. In addition, as the number of packets was very small and the duration was short, it looked suspicious. The experts finally decided that these logs are true-positive and suggested blocking the corresponding IP at a lower level of the infrastructure (e.g., at the firewall).

2) DISCOVERY OF SIMILAR FLOWS IN ATTACK PATTERNS

When they observed the timeline in the summary log analysis stage in the juxtaposed filled area mode (Figure 12a), they found that the temporal patterns of *Pattern Block (green), RegEx (purple), and Service Attack (brown)* were similar. They looked into the details by choosing the time range with with the highest number of logs (between 01:50 and 02:20). By utilizing the statistical data of the rules, the patterns in the building-bar charts, and the names of the rules, they were able to group attacks with similar patterns.

The first group consisted of *SMB Service Connect* attack of the *Service Attack* type, *MS Windows SMB RCE Scan* attack, *MS Windows SMB Doublepulsar Kernel Dll Injection* attack of Pattern block type, and five *RegEx* attacks with "SMB" in rule name. As the name implies, all these attacked are related to the SMB vulnerability. Among them, the rule of the *Service Attack* type is able to cope with various types of SMB vulnerability attacks by monitoring mainly the ports used by the SMB service (e.g., port 130 and 445), the *Pattern Block* rules corresponded to *wannacry* ransomeware, which was very popular worldwide in May 2017. Thus, as shown in Figure 12b, the detection results for the three rules are similar. Because the rule of the *Service Attack* type (brown, the first row of the division area) does not completely cover the rules of *Pattern Block* (green, the second and third row of the division area) on the detection result, and the number of logs is not very large, it seems necessary to co-use the rules of the *Pattern Block*. However, it was difficult to find large similarities with the detection results of the five RegEx rules. When they looked at the similarities within the results of *RegEX* attacks, code 1044 and 1056 show similar overall results. These two rules are candidates for integration into a single rule by improving the signature.

The second group was composed of the *Win32/Netsky.worm.J.K attack* of the *Service Attack* type and the *Win32/Netsky.worm.J.K-7 (tcp 25)* attack of the *Pattern Block* type. These two rules are related to the same worm (Figure 12c), the detection results were almost the same in the data such as the attack period and the attack count, and their outlier score showed a similar tendency. However, as the *Pattern Block* rule detected more and the information for the two endpoints was almost the same, it seems more reasonable to use only the *Pattern Block* rule.

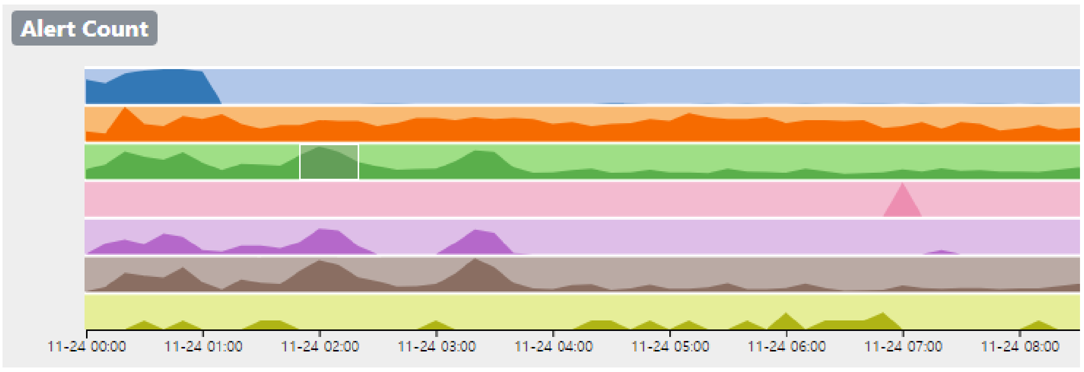
Based on the situation of these two groups, the experts concluded that logs showed similar timeline trends because multiple rules simultaneously detect attacks against the same service or vulnerability, even if the attack types are different. They were able to find this situation in a different time range other than the time range chosen above.

VII. DISCUSSION AND FUTURE WORK

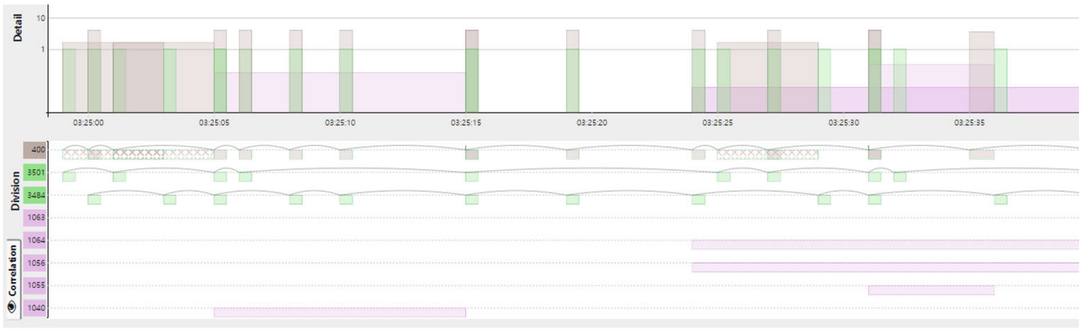
We conducted a workshop on IDPS log analysis and Hyperion with members of the IDPS development team. Thirteen researchers (including analysts) and developers, who had about ten to twenty years of experience in the security system development and operation, attended the workshop for two hours. Through feedback from workshop participants and the experience gained during the case study, we describe the limitations and the future direction of development of Hyperion.

A. SCALABILITY

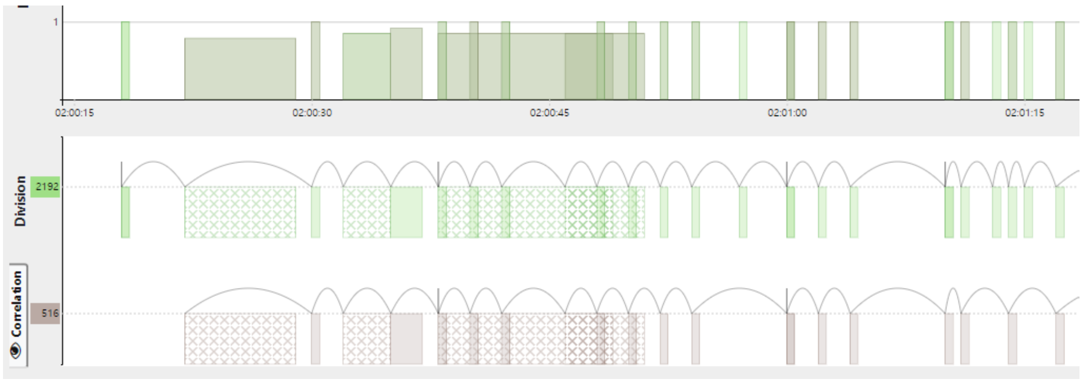
According to the security experts, when analyzing the IDPS logs, it would be useful to utilize the temporal patterns (e.g., hourly or daily). It can be an effective task in log analysis to find a day with an different pattern by comparing days over a week. In this project, as we used logs for only one day, we could not confirm whether such a task can be performed on Hyperion. Using a step-by-step approach, we expected Hyperion to be able to handle data for more than one day. However, for the data to be used in the summary log analysis stage, it may be necessary to perform preprocessing work periodically at least once a day. In addition, to perform the filtering operations (e.g., select a time range) involved in moving to the detail analysis stage, the use of large-scale



(a) Trends of *Pattern Block* (green), *RegEx* (purple), and *Service Attack* (brown) were similar in the alert count timeline.



(b) The *Service Attack* type (brown, the first row of the division area) log was similar to the set of two *Pattern Block* type (green, the second and third row of the division area) logs. However, there was no major similarity with the *RegEx* type (purple).



(c) The log of the two attacks (Win32/Netsky.worm.J.Kattack of the *Service Attack* type and the Win32/Netsky.worm.J.K-7 (tcp 25) attack of the *Pattern Block* type) showed a very similar pattern. Equally detected logs can be judged by the overlapped color at the detail area or the appearance of the division area.

FIGURE 12. **Analysis of various aspects of the attack on ENTITY-06.**

data processing engines such as Apache Spark [37] will be required.

B. ANALYSIS PROCESS AND METHOD

Currently, in Hyperion, the user selects the network entity without a hint and proceeds to the log analysis step. This takes into account the fact that analysts have to periodically perform log analysis at regular intervals (e.g., for a daily report). However, to improve the efficiency of the analysis and to support an analysis with non-periodic events such as intrusion attempts, it is necessary to introduce a method for assigning priorities to network entities. An example would be to add a sparkline visualization that shows the status of the logs collected for each network entity in the first stage.

Regarding visual analysis of logs, the experts were most interested in finding attacks with similar patterns using the building-bar chart. However, they commented that a better visualization method is required if there are numerous alerts with a small attack count, such as the *Denial of Service* attack. Their suggested improvement is to visualize the case where the destination port changes regularly on the same IP and IPs are attacked consecutively.

Due to the characteristics of monitoring the network traffic according to pre-defined rules, it is difficult to grasp the exact effects of attacks related to the alert with only IDPS logs. For example, in the case studies, we found that a *SERVER_HTTP_PORT* attack was detected, but we could not determine whether the attack succeeded or failed without investigating the actual target. In other words, to accurately comprehend the status of the network and the objects being monitored, it is necessary to collectively analyze not only the IDPS logs but also other types of logs, such as the payload of the attack or the usage history of the resources (e.g., CPU and memory).

C. INTEGRATION WITH THE IDPS

As Hyperion is a project that started with the aim of supporting the security control cycle through rule and log analysis, integrating this system with the actual IDPS was not our goal. Under this restriction, the following points remain limitations of this project.

In the first stage of Hyperion, there are visualization and interaction to browse the relationship between rules, templates, and network entities. The experts generally responded favorably to the visualization involved in this stage. As the number of IDPS rules increased, the burden of rule management also increased. Therefore, the experts commented that it can be helpful to identify the different parts of the template using Hyperion. The experts were also interested in the network-policy map. Specifically, the tree-shaped visualization would be more helpful in understanding the status of network entities and templates than the tabular view they are currently using. However, they commented that there are limitations to practical management because Hyperion cannot modify the templates and rules of the actual IDPS.

In the detailed log analysis stage, the experts stated that displaying the statistics of detected attacks is very effective, as the utility of the rule is immediately visible, and it may be possible to implement this feature in their product promptly. They also mentioned that simulating the results from parameter adjustments would be useful. However, they mentioned it was necessary to directly connect with the engine of the IDPS for reflecting the changes to fully utilize this feature.

VIII. CONCLUSION

In this paper, we presented a visual analytics system, Hyperion, for analyzing IDPS rules and logs. We addressed many challenges in a security control cycle that consists of rule application, information collection, log analysis, and rule revision. The case study showed that Hyperion was useful, and the feedback from domain experts presented a direction for improving the system. For future work, we would like to design a visual analytics system that can integrate multiple logs other than the IDPS and improve the scalability of the system.

REFERENCES

[1] (Apr. 1, 2020). *Security Enhanced Linux*. [Online]. Available: https://www.nsa.gov/What-We-Do/Research/SELinux/

[2] K. Abdullah, C. P. Lee, G. Conti, J. A. Copeland, and J. Stasko, "IDS rainStorm: Visualizing IDS alarms," in *Proc. IEEE Workshop Visualizat. Comput. Secur. (VizSEC)*, Oct. 2005, pp. 1–10, doi: 10.1109/VIZSEC.2005.1532060.

[3] M. Alsaleh, A. Alqahtani, A. Alarifi, and A. Al-Salman, "Visualizing PHPIDS log files for better understanding of Web server attacks," in *Proc. 10th Workshop Visualizat. Cyber Secur. VizSec*, 2013, pp. 1–8.

[4] M. Aupetit, Y. Zhauniarovich, G. Vasiliadis, M. Dacier, and Y. Boshmaf, "Visualization of actionable knowledge to mitigate drdos attacks," in *Proc. IEEE Symp. Visualizat. Cyber Secur. (VizSec)*, Oct. 2016, pp. 1–8, doi: 10.1109/VIZSEC.2016.7739577.

[5] E. Bertini, P. Hertzog, and D. Lalanne, "SpiralView: Towards security policies assessment through visual correlation of network resources with evolution of alarms," in *Proc. IEEE Symp. Vis. Anal. Sci. Technol.*, Oct. 2007, pp. 139–146, doi: 10.1109/VAST.2007.4389007.

[6] D. M. Best, A. Endert, and D. Kidwell, "7 key challenges for visualization in cyber network defense," in *Proc. 11th Workshop Visualizat. Cyber Secur. VizSec*, 2014, pp. 33–40, doi: 10.1145/2671491.2671497.

[7] P. Clemente, B. Kaba, J. Rouzaud-Cornabas, M. Alexandre, and G. Aujay, "Sptrack: Visual analysis of information flows within selinux policies and attack logs," in *Proc. Int. Conf. Act. Media Technol.* Berlin, Germany: Springer, 2012, pp. 596–605.

[8] 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, doi: 10.1109/MNET.2012.6375888.

[9] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, *A Geometric Framework for Unsupervised Anomaly Detection*. Boston, MA, USA: Springer, 2002, pp. 77–101, doi: 10.1007/978-1-4615-0953-0_4.

[10] G. A. Fink, C. L. North, A. Endert, and S. Rose, "Visualizing cyber security: Usable workspaces," in *Proc. 6th Int. Workshop Visualizat. Cyber Secur.*, Oct. 2009, pp. 45–56, doi: 10.1109/VIZSEC.2009.5375542.

[11] F. Fischer, F. Mansmann, D. A. Keim, S. Pietzko, and M. Waldvogel, "Large-scale network monitoring for visual analysis of attacks," in *Visualization for Computer Security*. Berlin, Germany: Springer, 2008, pp. 111–118.

[12] R. Gove, "V3SPA: A visual analysis, exploration, and diffing tool for SELinux and SEAndroid security policies," in *Proc. IEEE Symp. Visualizat. Cyber Secur. (VizSec)*, Oct. 2016, pp. 1–8.

[13] V. T. Guimaraes, C. M. D. S. Freitas, R. Sadre, L. M. R. Tarouco, and L. Z. Granville, "A survey on information visualization for network and service management," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 285–323, 1st Quart., 2016, doi: 10.1109/COMST.2015.2450538.

[14] W. Javed, B. McDonnel, and N. Elmqvist, "Graphical perception of multiple time series," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 6, pp. 927–934, Nov. 2010, doi: 10.1109/TVCG.2010.162.

[15] H. Kim, S. Ko, D. Seong Kim, and H. Kang Kim, "Firewall ruleset visualization analysis tool based on segmentation," in *Proc. IEEE Symp. Visualizat. Cyber Secur. (VizSec)*, Oct. 2017, pp. 1–8, doi: 10.1109/VIZSEC.2017.8062196.

[16] H. Koike and K. Ohno, "SnortView: Visualization system of snort logs," in *Proc. ACM Workshop Visualizat. Data Mining Comput. Secur. VizSEC/DMSEC*, 2004, pp. 143–147.

[17] H. Koike, K. Ohno, and K. Koizumi, "Visualizing cyber attacks using IP matrix," in *Proc. IEEE Workshop Visualizat. Comput. Secur. (VizSEC)*, Oct. 2005, pp. 91–98, doi: 10.1109/VIZSEC.2005.1532070.

[18] C. P. Lee, J. Trost, N. Gibbs, R. Beyah, and J. A. Copeland, "Visual firewall: Real-time network security monitor," in *Proc. IEEE Workshop Visualizat. Comput. Secur. (VizSEC)*, Oct. 2005, pp. 129–136, doi: 10.1109/VIZSEC.2005.1532075.

[19] Y. Livnat, J. Agutter, S. Moon, and S. Foresti, "Visual correlation for situational awareness," in *Proc. IEEE Symp. Inf. Visualizat. INFOVIS*, Oct. 2005, pp. 95–102, doi: 10.1109/INFVIS.2005.1532134.

[20] F. Mansmann, F. Fischer, D. A. Keim, and S. C. North, "Visual support for analyzing network traffic and intrusion detection events using TreeMap and graph representations," in *Proc. Symp. Comput. Hum. Interact. Manage. Inf. Technol. CHiMiT*, 2009, pp. 3:19–3:28.

[21] F. Mansmann, T. Göbel, and W. Cheswick, ''Visual analysis of complex firewall configurations,'' in *Proc. 9th Int. Symp. Visualizat. Cyber Secur. VizSec*, 2012, pp. 1–8, doi: 10.1145/2379690.2379691.

[22] S. Marouf and M. Shehab, ''SEGrapher: Visualization-based SELinux policy analysis,'' in *Proc. 4th Symp. Configuration Analytics Autom. (SAFECONFIG)*, Oct. 2011, pp. 1–8.

[23] R. Marty, *Applied Security Visualization*, 1st ed. Reading, MA, USA: Addison-Wesley, 2008.

[24] S. Mckenna, D. Staheli, and M. Meyer, ''Unlocking user-centered design methods for building cyber security visualizations,'' in *Proc. IEEE Symp. Visualizat. Cyber Secur. (VizSec)*, Oct. 2015, pp. 1–8, doi: 10.1109/VIZSEC.2015.7312771.

[25] S. P. Morrissey and G. Grinstein, ''Visualizing firewall configurations using created voids,'' in *Proc. 6th Int. Workshop Visualizat. Cyber Secur.*, Oct. 2009, pp. 75–79, doi: 10.1109/VIZSEC.2009.5375546.

[26] T. Munzner, ''Visualization analysis and design,'' in *AK Peters Visualization Series*. Boca Raton, FL, USA: CRC Press, 2014.

[27] K. Nyarko, T. Capers, C. Scott, and K. Ladeji-Osias, ''Network intrusion visualization with NIVA, an intrusion detection visual analyzer with haptic integration,'' in *Proc. 10th Symp. Haptic Interface Virtual Environ. Teleoperator Syst. HAPTICS*, Mar. 2002, pp. 277–284, doi: 10.1109/HAPTIC.2002.998969.

[28] L. Pan and Q. Xu, ''Visualization analysis of multi-domain access control policy integration based on tree-maps and semantic substrates,'' *Intell. Inf. Manage.*, vol. 4, no. 5, pp. 188–193, 2012.

[29] R. W. Reeder, L. Bauer, L. F. Cranor, M. K. Reiter, K. Bacon, K. How, and H. Strong, ''Expandable grids for visualizing and authoring computer security policies,'' in *Proc. 26th Annu. CHI Conf. Hum. Factors Comput. Syst. CHI*, 2008, pp. 1473–1482, doi: 10.1145/1357054.1357285.

[30] M. Roesch, ''Snort - lightweight intrusion detection for networks,'' in *Proc. 13th USENIX Conf. Syst. Admin. LISA*, Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238.

[31] Y. Shi, Y. Zhang, F. Zhou, Y. Zhao, G. Wang, R. Shi, and X. Liang, ''IDSPlanet: A novel radial visualization of intrusion detection alerts,'' in *Proc. 9th Int. Symp. Vis. Inf. Commun. Interact. VINCI*, 2016, pp. 25–29.

[32] H. Shiravi, A. Shiravi, and A. A. Ghorbani, ''IDS alert visualization and monitoring through heuristic host selection,'' in *Information and Communications Security*, M. Soriano, S. Qing, J. López, eds. Berlin, Germany: Springer, 2010, pp. 445–458.

[33] 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, doi: 10.1109/TVCG.2011.144.

[34] T. Tran, E. Al-Shaer, and R. Boutaba, ''Policyvis: Firewall security policy visualization and inspection,'' in *Proc. 21st Conf. Large Installation Syst. Admin. Conf. LISA*, Berkeley, CA, USA: USENIX Association, 2007, pp. 1:1–1:16.

[35] W. Xu, M. Shehab, and G.-J. Ahn, ''Visualization based policy analysis: Case study in SELinux,'' in *Proc. 13th ACM Symp. Access Control Models Technol. SACMAT*, 2008, pp. 165–174, doi: 10.1145/1377836.1377863.

[36] W. Xu, M. Shehab, and G.-J. Ahn, ''Visualization-based policy analysis for SELinux: Framework and user study,'' *Int. J. Inf. Secur.*, vol. 12, no. 3, pp. 155–171, Jun. 2013, doi: 10.1007/s10207-012-0180-7.

[37] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, ''Apache spark: A unified engine for big data processing,'' *Commun. ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016, doi: 10.1145/2934664.

[38] T. Zhang, X. Wang, Z. Li, F. Guo, Y. Ma, and W. Chen, ''A survey of network anomaly visualization,'' *Sci. China Inf. Sci.*, vol. 60, no. 12, Apr. 2017, Art. no. 121101.

[39] Y. Zhang, Y. Xiao, M. Chen, J. Zhang, and H. Deng, ''A survey of security visualization for computer network logs,'' *Secur. Commun. Netw.*, vol. 5, no. 4, pp. 404–421, Apr. 2012.

[40] Y. Zhao, X. Liang, X. Fan, Y. Wang, M. Yang, and F. Zhou, ''MVSec: Multi-perspective and deductive visual analytics on heterogeneous network security data,'' *J. Visualizat.*, vol. 17, no. 3, pp. 181–196, Aug. 2014, doi: 10.1007/s12650-014-0213-6.

[41] Y. Zhao, F. Zhou, X. Fan, X. Liang, and Y. Liu, ''IDSRadar: A real-time visualization framework for IDS alerts,'' *Sci. China Inf. Sci.*, vol. 56, no. 8, pp. 1–12, Aug. 2013.

SEUNGHOON YOO (Member, IEEE) received the B.S. degree in computer science from the Republic of Korea Air Force Academy, in 2006, and the M.S. degree in computer science and engineering and the Ph.D. degree in computer science from Seoul National University, in 2011 and 2019, respectively.

He is currently an Associate Professor of computer science at the Republic of Korea Air Force Academy, South Korea. His research interests include information visualization, visual analytics, and computer security.

JAEMIN JO (Member, IEEE) received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Seoul National University, Seoul, South Korea, in 2014 and 2020, respectively.

He is currently a Postdoctoral Researcher at the Department of Computer Science and Engineering, Seoul National University, South Korea. His research interests include large-scale data visualization, progressive visual analytics, and human-AI interaction.

BOHYOUNG KIM (Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in computer science and engineering from Seoul National University, Seoul, South Korea, in 2001.

She is currently an Associate Professor at the Division of Biomedical Engineering, Hankuk University of Foreign Studies, South Korea. Her research interests include computer graphics, volume visualization, medical imaging, and information visualization.

JINWOOK SEO (Member, IEEE) received the B.S. and M.S. degrees in computer science from Seoul National University, and the Ph.D. degree in computer science from the Human-Computer Interaction Lab, University of Maryland, College Park, USA, in 2005.

He is currently a Professor at the Department of Computer Science and Engineering, Seoul National University, South Korea. His research interests include human–computer interaction, interaction design, information visualization, visual analytics, and health informatics.