A novel hybrid-based approach of snort automatic rule generator and security event correlation (SARG-SEC)

Ebrima Jaw [1,2], Xueming Wang [Corresp. 1]

[1] College of Computer Science and Technology, Guizhou University, Guiyang, Guizhou, China

[2] School of Information Technology and Communication, University of The Gambia, Banjul, Greater Banjul, Gambia

Corresponding Author: Xueming Wang
Email address: xmwang3@gzu.edu.cn

The rapid advanced technological development alongside the Internet with its cutting-edge applications has positively impacted human society in many aspects. Nevertheless, it equally comes with the escalating privacy and critical cybersecurity concerns that can lead to catastrophic consequences, such as overwhelming the current network security frameworks. Consequently, both the industry and academia have been tirelessly harnessing various approaches to design, implement and deploy intrusion detection systems (IDSs) with event correlation frameworks to help mitigate some of these contemporary challenges. There are two common types of IDS: signature and anomaly-based IDS. Signature-based IDS, specifically, Snort works on the concepts of rules. However, the conventional way of creating Snort rules can be very costly and error-prone. Also, the massively generated alerts from heterogeneous anomaly-based IDSs is a significant research challenge yet to be addressed. Therefore, this paper proposed a novel Snort Automatic Rule Generator (SARG) that exploits the network packet contents to automatically generate efficient and reliable Snort rules with less human intervention. Furthermore, we evaluated the effectiveness and reliability of the generated Snort rules, which produced promising results. In addition, this paper proposed a novel Security Event Correlator (SEC) that effectively accepts raw events (alerts) without prior knowledge and produces a much more manageable set of alerts for easy analysis and interpretation. As a result, alleviating the massive false alarm rate (FAR) challenges of existing IDSs. Lastly, we have performed a series of experiments to test the proposed systems. It is evident from the experimental results that SARG-SEC has demonstrated impressive performance and could significantly mitigate the existing challenges of dealing with the vast generated alerts and the labor-intensive creation of Snort rules.

1 **A novel hybrid-based approach of Snort Automatic Rule Generator and Security**
2 **Event Correlation (SARG-SEC)**

3   Ebrima Jaw[1,2] Wang Xueming[1]

4   [1]College of Computer Science and Technology, Guizhou University (GZU), Guiyang, Guizhou,
5   China

6   [2]School of Information Technology and Communication, University of the Gambia (UTG),
7   Banjul, Greater Banjul, The Gambia

9   Corresponding Author:

10   Wang Xueming[1]

11   Guizhou University, East Campus, Bo Xue Building, Office 915, Guiyang, Guizhou, 550025,
12   China

13   Email address: xmwang3@gzu.edu.cn

ABSTRACT

The rapid advanced technological development alongside the Internet with its cutting-edge applications has positively impacted human society in many aspects. Nevertheless, it equally comes with the escalating privacy and critical cybersecurity concerns that can lead to catastrophic consequences, such as overwhelming the current network security frameworks. Consequently, both the industry and academia have been tirelessly harnessing various approaches to design, implement and deploy intrusion detection systems (IDSs) with event correlation frameworks to help mitigate some of these contemporary challenges. There are two common types of IDS: signature and anomaly-based IDS. Signature-based IDS, specifically, Snort works on the concepts of rules. However, the conventional way of creating Snort rules can be very costly and error-prone. Also, the massively generated alerts from heterogeneous anomaly-based IDSs is a significant research challenge yet to be addressed. Therefore, this paper proposed a novel Snort Automatic Rule Generator (SARG) that exploits the network packet contents to automatically generate efficient and reliable Snort rules with less human intervention. Furthermore, we evaluated the effectiveness and reliability of the generated Snort rules, which produced promising results. In addition, this paper proposed a novel Security Event Correlator (SEC) that effectively accepts raw events (alerts) without prior knowledge and produces a much more manageable set of alerts for easy analysis and interpretation. As a result, alleviating the massive false alarm rate (FAR) challenges of existing IDSs. Lastly, we have performed a series of experiments to test the proposed systems. It is evident from the experimental results that SARG-SEC has demonstrated impressive performance and could significantly mitigate the existing challenges of dealing with the vast generated alerts and the labor-intensive creation of Snort rules.

INTRODUCTION

The advent of the Internet has come with the cost of wide-scale adoption of innovative technologies such as cloud computing (Al-Issa et al., 2019), artificial intelligence (Miller, 2019), the Internet of Things (IoT) (J. H. Park, 2019), and vast ranges of web-based applications. Therefore, leading to considerable security and privacy challenges of managing these cutting-edge applications using traditional security and privacy protection mechanisms such as firewall, anti-virus, virtual private networks (VPNs), and anti-spyware (Meryem & Ouahidi, 2020). However, due to the vast range of competitive solutions such as higher efficiencies, scalability, reduced costs, computing power, and most importantly, the delivery of services, these technologies continue to revolutionize various aspects of our daily lives drastically, for instance, in the health care systems, research industry, the global business landscape, government, and private sectors (Xue & Xin, 2016).

Moreover, countless types of cyber-attacks have evolved dramatically since the inception of the Internet and the swift growth of ground-breaking technologies. For example, social engineering or phishing (Kushwaha et al., 2017), zero-day attack (Jyothsna & Prasad, 2019), malware attack (McIntosh et al., 2019), denial of service (DoS)(Verma & Ranga, 2020),

68 unauthorized access of confidential and valuable resources (Saleh et al., 2019). Additionally,
69 according to the authors of (Papastergiou et al., 2020), a nation's competitive edge in the global
70 market and national security is currently driven by harnessing these efficient, productive, and
71 highly secure leading-edge technologies with intelligent and dynamic means of timely detection
72 and prevention of cyberattacks. Nevertheless, irrespective of the tireless efforts of security
73 experts in defense mechanisms, hackers have always found ways to get away with targeted
74 resources from valuable and most trusted sources worldwide by launching versatile,
75 sophisticated, and automated cyber-attacks. As a result, causing tremendous havoc to
76 governments, businesses, and even individuals (Sarker et al., 2020).
77   For instance, the authors of (Damaševičius et al., 2021) intriguingly review various cyber-
78 attacks and their consequences. Firstly, the paper highlights the estimated 6 trillion USD of
79 cyber-crimes by 2021 and the diverse global ground-breaking cyber-crimes that could lead to the
80 worldwide loss of 1 billion USD. Finally, it highlights a whopping 1.5 trillion USD of
81 cyberattack revenues resulting from two to five million computers compromised daily.
82 Furthermore, according to published statistics of AV-TEST Institute in Germany, during the year
83 2019, there were more than 900 million malicious executables identified among the security
84 community, and data breach costing 8.19 million USD for the United States, predicted to grow in
85 subsequent years. Moreover, the Congressional Research Service of the USA has highlighted
86 that cybercrime-related incidents have cost the global economy an annual loss of 400 billion
87 USD.
88   Similarly, 2016 alone recorded more than a whopping 3 billion zero-day attacks and
89 approximately 9 billion stolen data records since 2013 (Khraisat et al., 2019). In addition, the
90 energy sector in Ukraine suffered catastrophic coordinated cyberattacks (APT) that led to a
91 significant blackout affecting more than 225,000 people. They also highlighted similar alarming
92 APT threats, such as DragonFly, TRITON, and Crashoverride, that could cause devasting
93 consequences to individual lives and the global economy, thereby leading to national security
94 threats (Grammatikis et al., 2021). Accordingly, it is essential for security experts to design,
95 implement and deploy robust and efficient cybersecurity frameworks to alleviate the current and
96 subsequent alarming losses for the government and private sectors. Additionally, it is an urgent
97 and crucial challenge to effectively identify the increasing cyber incidents and cautiously protect
98 these relevant applications from such cybercrimes.
99   Therefore, the last few decades have witnessed Intrusion Detection Systems (IDSs)
100 increasing in popularity due to their inherent ability to detect an intrusion or malicious activities
101 in real-time. Consequently, making IDSs critical applications to safeguard numerous networks
102 from malicious activities (Dang, 2019; Meryem & Ouahidi, 2020). Finally, James P. Anderson
103 claimed credit for the inception of the IDS concept in his paper written in 1980 (Anderson,
104 1980), highlighting various methods of enhancing computer security threat monitoring and
105 surveillance.
106   Intrusion Detection is the procedure of monitoring the events occurring in a computer system
107 or network and analyzing them for signs of intrusion," similarly, an intrusion is an attempt to

108   bypass the security mechanisms of a network or a computer system, thereby compromising the
109   Confidentiality, Integrity, and Availability (CIA) (Kagara & Md Siraj, 2020). Moreover, an IDS
110   is any piece of hardware or software program that monitors diverse malicious activities within
111   computer systems and networks based on network packets, network flow, system logs, and
112   rootkit analysis (Bhosale et al., 2020; H. Liu & Lang, 2019). Misused detection (knowledge or
113   signature-based) and anomaly-based methods are the two main approaches to detecting
114   intrusions within computer systems or networks. Nevertheless, the past decade has witnessed the
115   rapid rise of the hybrid-based technique, which typically exploits the advantages of the two
116   methods mentioned above to yield a more robust and effective system (Saleh et al., 2019).
117   Misused IDS (MIDS) is a technique where specific signatures of well-known attacks are
118   stored and eventually mapped with real-time network events to detect an intrusion or intrusive
119   activities. The MIDS technique is reliable and effective and usually gives excellent detection
120   accuracy, particularly for previously known intrusions. Nevertheless, this approach is
121   questionable due to its inability to detect novel attacks. Also, it requires more time to analyze and
122   process the massive volume of data in the signature databases (Khraisat et al., 2020; Lyu et al.,
123   2021). The authors of (Jabbar & Aluvalu, 2018) presented an exceptional high-level SIDS
124   architecture, which includes both distributed and centralized modules that effectively enhanced
125   the protection of IoT networks against internal and external threats. Furthermore, the authors
126   exploit the Cooja simulator to implement a DoS attack scenario on IoT devices that rely on
127   version number modification and "Hello" flooding. Finally, the authors claimed that these attacks
128   might influence specific IoT devices' reachability and power consumption.
129   In contrast, the anomaly-based detection method relies on a predefined network behavior as
130   the crucial parameter for identifying anomalies and commonly operates on statistically
131   substantial network packets. For instance, incoming network packets or transactions are accepted
132   within the predefined network behavior. Otherwise, the anomaly detection system triggers an
133   alert of anomaly (Kagara & Md Siraj, 2020). It is essential to note that the main design idea of
134   the anomaly detection method is to outline and represent the usual and expected standard
135   behavior profile through observing activities and then defining anomalous activities by their
136   degree of deviation from the expected behavior profile using statistical-based, knowledge-based,
137   and machine learning-based methods (Jyothsna & Prasad, 2019; Khraisat et al., 2020). The
138   acceptable network behavior can be learned using the predefined network conditions, more like
139   blocklists or allowlists that determine the network behavior outside a predefined acceptable
140   range. For instance, "detect or trigger an alert if ICMP traffic becomes greater than 10% of
141   network traffic" when it is regularly only 8%.
142   Finally, the anomaly-based approach provides a broader range of advantages such as solid
143   generalizability, the ability to determine internal malicious activities, and a higher detection rate
144   of new attacks such as the zero-day attack. Nevertheless, the most profound challenge is the need
145   for these predefined baselines and the substantial number of false alarm rates resulting from the
146   fluctuating cyber-attack landscape (Einy et al., 2021). For instance, (Fitni & Ramli, 2020)
147   intelligently used logistic regression, decision tree, and gradient boosting to propose an

148 optimized and effective anomaly-based ensemble classifier. The authors claimed impressive
149 findings such as 98.8% performance accuracy, 98.8%, 97.1%, and 97.9% precision, recall, and
150 F1-score.
151   The hybrid-based intrusion detection systems (HBIDS) exploit the functionality of MIDS to
152 detect well-known attacks and flag novel attacks using the anomaly method. High detection rate,
153 accuracy, and fewer false alarm rates are some of the main advantages of this approach (Khraisat
154 et al., 2020). The authors of (Khraisat et al., 2020) suggested an efficient and lightweight hybrid-
155 based IDS that mitigates the security vulnerabilities of the Internet of Energy (IoE) within an
156 acceptable time frame. The authors intelligently exploit the combined strengths of K-means and
157 SVM and utilize the centroids of K-means to enhance the process of training and testing the
158 SVM model. Moreover, they selected the best value of "k" and fine-tuned the SVM for best
159 anomaly detection and claimed to have drastically reduced the overall detection time and
160 impressive performance accuracy of 99.9% compared to current cutting-edge approaches.
161   Additionally, the two classical IDS implementation methods are Network Intrusion Detection
162 Systems (NIDS) (Mirsky et al., 2018) and Host-based IDS (HIDS) (Aung & Min, 2018). A
163 HIDS detection method monitors and detects internal attacks using the data from audit sources
164 and host systems like firewall logs, database logs, application system audits, window server logs,
165 and operating systems (Khraisat et al., 2019). In contrast, NIDS is an intrusion detection
166 approach that analyses and monitors the entire traffic of computer systems or networks based on
167 flow or packet-based and tries to detect and report anomalies. For example, the distributed denial
168 of service (DDoS), denial of service (DoS), and other suspicious activities like internal illegal
169 access or external attacks (Niyaz et al., 2015). Unlike HIDS, NIDS usually protects an entire
170 network from internal or external intrusions. However, such a process can be very time-
171 consuming, high computational cost, and very inefficient, especially in most current cutting-edge
172 technologies with high-speed communication systems.
173   Nevertheless, this approach still has numerous advantages (Bhuyan et al., 2014). For
174 instance, it is more resistant to attacks compared to HIDS. Furthermore, it monitors and analyses
175 the complete network's traffic if appropriately located in a network, leading to a high probability
176 of detection rate. Also, NIDS is platform-independent, thus enabling them to work on any
177 platform without requiring much modification. Finally, NIDS does not add any overhead to the
178 network traffic (Othman et al., 2018).
179   Snort is a classic example of NIDS (Sagala, 2015). Irrespective of the availability of other
180 signature-based NIDS, such as Suricata and Zeek (Bro) (Ali, Shah, & Issac, 2018), this work
181 adopted Snort because it is the leading open-source NIDS with active and excellent community
182 support. Likewise, it is easy to install and run with readily available online resources.
183 Furthermore, because Suricata can use Snort's rulesets but is prone to false positives with the
184 need for an intensive system and network resource, we concluded to use Snort for the proposed
185 solutions.
186   Snort is a classical open-source NIDS that has the unique competence of performing packet
187 logging and real-time network traffic analysis within computer systems and networks using

188 content searching, matching, and protocol analysis (Aickelin et al., 2007; Tasneem et al., 2018).
189 Therefore, significantly contributing to the protection of some major commercial networks. Snort
190 is a signature-based IDS that detects malicious live Internet or network traffic utilizing the
191 predefined Snort rules, commonly applied in units of packets' header, statistical information
192 (packet size), and payload information. Thus, it has a unique feature of high detection rate and
193 accuracy (Tasneem et al., 2018). However, it cannot detect novel attacks and, at the same time, it
194 requires expert knowledge to create and update rules frequently, which is both costly and faulty.
195   Similarly, IDS have emerged as popular security frameworks that significantly minimized
196 various cutting-edge cyber-attacks over the past decade. Anomaly-based IDS has gained quite a
197 buzz among network and system administrators for monitoring and protecting their networks
198 against malicious attempts, which has achieved phenomenal success, especially in detecting and
199 protecting systems and networks against novel or zero-day attacks (Tama et al., 2019). However,
200 it comes with costly negative consequences of generating thousands or millions of false-positive
201 alerts or colossal amounts of complex data for humans to process and make timely decisions
202 (Sekharan & Kandasamy, 2018). As a result, administrators ignore these massive alerts, which
203 creates room for potential malicious attacks against highly valued and sensitive information
204 within a given system or network.
205   Accordingly, the past years have seen a growing interest in designing and developing
206 network security management frameworks from academia and industry, which involves
207 analyzing and managing the vast amount of data from heterogeneous devices, commonly
208 referred to as event correlation. Event correlation has significantly mitigated modern cyber-
209 attacks challenges using its unique functionality of efficiently and effectively analyzing and
210 making timely decisions from massive heterogeneous data (G. Suarez-Tangil et al., 2009).
211 Consequently, the past years have seen many researchers and professionals exploit the efficiency
212 of event correlation techniques to address the problems mentioned earlier (Dwivedi & Tripathi,
213 2015; Ferebee et al., 2011; Guillermo Suarez-Tangil et al., 2015).
214   Nevertheless, this field of research is at its infant stage as minimal work is done to address
215 these issues. Likewise, according to the authors' knowledge, none of the above solutions
216 provides comprehensive solutions that address the manual creation of Snort rules and the event
217 correlation as a single solution. Based on the above challenges, this paper proposed two effective
218 and efficient approaches to address the problems associated with the manual creation of Snort
219 rules and mitigating the excessive false alarm rates generated by current IDSs. First, we present
220 an automatic rule creation technique that focuses on packet header and payload information.
221 Generally, we need to find standard features by examining all the network traffic to create a rule.
222 Nonetheless, this method is inefficient and requires much time to complete the rule, and the
223 accuracy of the rules made is variable according to the interpreters' ability. Therefore, various
224 automatic rule (signature) generation methods have been proposed (Sagala, 2015).
225   However, most of these methods are used between two specific strings, which is still
226 challenging for creating reliable and effective Snort rules. Therefore, we present a promising
227 algorithm based on the content rules, enabling the automatic and easy creation of Snort rules

228 using packet contents. Secondly, we equally proposed a novel model that efficiently and
229 effectively correlates and prioritizes IDS alerts based on the severity using various features of a
230 network packet. Moreover, the proposed system does not need prior knowledge while comparing
231 two different alerts to measure the similarity in diverse attacks. The following are an overview of
232 the main contributions of this research work:

233   o The authors proposed an optimized and efficient Snort Automatic Rule Generator
234   (SARG) that automatically generates reliable Snort rules based on content features.
235   o Similarly, we present a novel Security Event (alert) Correlator (SEC) that drastically
236   and effectively minimized the number of alerts received for convenient interpretation.
237   o This paper also provides solid theoretical background knowledge for the readership of
238   the journal to clearly understand the fundamental functions and capabilities of Snort
239   and various correlation methods.
240   o Finally, the proposed approach has recorded an acceptable number of alerts, which
241   directly correlates with significantly mitigating the challenges of false alarm rates.

242   The rest of the paper is organized as follows: Section 2 discusses important background
243 concepts. Similarly, Section 3 explains the materials and methods of the proposed system. Next,
244 section 4 highlights the results and discussions of the proposed approach. Finally, the paper
245 concludes in Section 5.

ESSENTIAL CONCEPTS

247   This section presents brief essential concepts that support the work in this research paper,
248 which will provide readers with the necessary knowledge to appreciate this research and similar
249 results better.

Synopsis of HIDS and NIDS functionalities

251   The crucial advantages of a host-based IDS are; the ability to detect internal malicious
252 attempts that might elude a NIDS, the freedom to access already decrypted data compared to
253 NIDS, and the ability to monitor detect advanced persistent threats (APT). In contrast, some of
254 its disadvantages are; firstly, they are expensive as it requires lots of management efforts to
255 mount, configure and manage. It is also vulnerable to specific DoS attacks and uses many
256 storage resources to retain audit records to function correctly (M. Liu et al., 2019; Saxena et al.,
257 2017). The authors of (Arrington et al., 2016) use the innovative strength of machine learning
258 such as artificial immune systems to present an interesting host-based IDS. Finally, they claimed
259 to have achieved a reasonable detection rate through rescinding out the noise within the
260 environment.
261   The central idea of any classical NIDS is using rulesets to identify and alert malicious
262 attempts. The majority of the NIDS comes with pre-installed rules that can be modified to target
263 specific attacks (Ojugo et al., 2012). For example, creating a rule for a possible probing attack
264 and saving it in the *local.rules* of Snort IDS will ensure an alert is raised whenever an intrusion is
265 initiated that matches the rule. Furthermore, the essential functionalities of a classical NIDS are:

266 practical identification and alerting of policy violations, suspicious unknown sources, destination
267 network traffics, port scanning, and other common malicious attempts. However, the bulk of
268 NIDS requires costly hardware with expensive enterprise solutions, making it hard to acquire
269 (Elrawy et al., 2018). The authors of (Nyasore et al., 2020) presented an intriguing challenge of
270 evaluating the overlap among various rules in two rulesets of the Snort NIDS. However, the
271 work failed to assess the distinction between diverse rulesets explicitly.

272   Consequently, the work presented in (Sommestad et al., 2021) provides an interesting
273 empirical analysis of the detection likelihood of 12 Snort rulesets against 1143 misuse attempts
274 to evaluate their effectiveness on a signature-based IDS. Similarly, they listed certain features as
275 the determining factor of the detection probability. Finally, they claimed impressive results such
276 as a significant 39% raise of priority-1-alerts against the misuse attempts and 69– 92%
277 performance accuracy for various rulesets.

278   Fig. 1 illustrates a standard representation of a HIDS and NIDS architecture with unique
279 functionalities in detecting malicious activities. For instance, Fig. 1 shows a malicious user
280 (attacker) who initiated a DDoS attack against one of the internal servers within the LAN.
281 However, due to the internal security mechanisms, packets are inspected by the firewall as the
282 first layer of protection. Interestingly, some malicious packets can bypass the firewall due to the
283 cutting-edge attack mechanisms, necessitating NIDS (Bul'ajoul et al., 2015). Therefore, the
284 NIDS receives the packets and does a further packet inspection. If there are any malicious
285 activities, the packets are blocked and returned to the firewall. Then, the firewall will drop the
286 packets or notify the network administrator, depending on the implemented policies. It is crucial
287 to note that the same applies to the outgoing packets from the LAN to the WAN.

288   In contrast, the scenario depicted at the top of Fig. 1 shows a regular user requesting web
289 services. Initially, the request goes via the same process as explained above. Then, if the NIDS
290 qualifies the requests, it is sent to the server through the network switch to the webserver.
291 Furthermore, the server responds with the required services, which goes through the same
292 process as the reverse. However, this process is not shown in the diagram. Lastly, Fig. 1 also
293 presents the architecture of HIDS as labeled on individual devices, which administers further
294 packet inspection within the host (Vokorokos & BaláŽ, 2010), thereby enhancing the security
295 level of a given network or computer system.

Summary of Snort and its components

297   Snort is a popular and influential cross-platform lightweight signature-based network
298 intrusion detection and prevention system with multiple packet tools. The power of Snort lies in
299 the use of rules, some of which are preloaded, but we can also design customized rules to merely
300 send alerts or block specific network traffic when they meet the specified criteria. Additionally,
301 alerts can be sent to a console or displayed on a graphical user interface, but they can also be
302 logged to a file for future or further analysis. Finally, Snort also enables the configuration
303 options of logging alerts to databases such as MSQL and MongoDB or sending an email to a

304 specified responsible person if there are alerts or suspicious attempts (Ali, Shah, Khuzdar, et al.,
305 2018).
306   Moreover, Snort has three running modes: sniffer, packet logger, and network-based IDS
307 mode (Tasneem et al., 2018). The sniffer mode is run from the command line mode, and its
308 primary function is just inspecting the header details of packets and printing it on the console.
309 For instance, *./Snort –vd* will instruct Snort to display packet data with its headers. The packer
310 logger mode interestingly inspects packets and logs them into a file in the root directory. Then, it
311 can be viewed using tcpdump, snort, or other applications for further analysis. For example,
312 *./Snort -dev -l ./directory_name*, will prompt Snort to go to packet logger mode and log the
313 packets in a given directory; if the *directory_name* did not exist, then it will exit and throw an
314 error. Finally, the network-based IDS mode utilizes the embedded rules to determine any
315 potential intrusive activities within a given network. Snort does this with the help of the network
316 interface card (NIC) running in the promiscuous mode to intercept and analyze real-time network
317 traffic. For instance, the command *./Snort -dev -l ./log -h 172.162.1.0/24 -c snort.conf* will
318 prompt Snort to log network packets that trigger the specified rules in *snort.conf*, where
319 *snort.conf* is the configuration file that applies all the rules to the incoming packets for any
320 malicious attempt (Tasneem et al., 2018).
321   It is equally important to note that Snort also has the strength of real-time packet logging,
322 content matching, and searching with protocol analysis. It also has the advantage of serving as a
323 prevention tool instead of just monitoring (Ali, Shah, Khuzdar, et al., 2018). However, it does
324 come with some notable shortcomings. For example, a very unpopular GUI makes using Snort a
325 bit difficult. Additionally, the vast number of network traffics can compromise the reliable and
326 functional operation of Snort, which inspires the use of Pfring and Hyperscan to reinforce the
327 functionality of Snort for efficiency and reliability. Finally, and most importantly, caution needs
328 to be exercised in creating Snort rules to avoid the apparent challenge of many false alarm rates
329 (FAR). (W. Park & Ahn, 2017). Snort comprises five logical components that determine and
330 classify potential malicious attacks or any undesired threat against computer systems and
331 networks (Ali, Shah, Khuzdar, et al., 2018). Finally, interested readers can refer to the following
332 references for the details of the Snort components (Essid et al., 2021; Mishra et al., 2016; Shah &
333 Issac, 2018).
334 **Snort Rule Syntax**
335   Snort utilizes a flexible, lightweight, and straightforward authoritative rules-language
336 primarily written in a single line as in versions preceding to 1.8. However, the present Snort
337 versions allow the spanning of rules in multiple lines but require the addition of black (\) at the
338 end of each line; generally, Snort rules are composed of dual logical portions. For instance, the
339 rule header and the rule options (Khurat & Sawangphol, 2019). Generally, Snort rules share all
340 sections of the rule option like the general options, payload detection options, non-payload
341 detection options, and post-detection options. However, it can be specified differently depending
342 on the configuration approach. Finally, Snort rules are generally applied to the headers of the
343 application, transport, and network layers such as FTP, HTTP, ICMP, IP, UDP, and TCP

344 (Chanthakoummane et al., 2016; Khurat & Sawangphol, 2019). However, they can also be
345 applied to the packet payload, which is the adopted approach for SARG.

346   Table 1 presents a classical representation of Snort rules (Khurat & Sawangphol, 2019). The
347 two rules shown in Table 1 denote that an alert will be triggered based on an icmp traffic
348 protocol from *any* source IP address and port number to *any* destination IP address and port
349 number if the traffic content contains a *probe*. Consequently, this will show a message *probe*
350 *attack*, and the signature ID of this rule is 1000023. Similarly, the second rule is almost the same
351 as the first, except that the action is "*log*" instead of "*alert*," while the destination port number is
352 80 instead of "*any*" number.

The Rule Header

354   The Snort rule header comprises the specified actions, protocol, addresses of source and
355 destination, and port numbers. The default Snort actions are: *alert*, *log*, and *pass*, and it is a
356 required field for every Snort rule, and it defaults to *alert* if not specified explicitly.
357 Nevertheless, *drop*, *reject,* and *sdrop* are additional options for an inline mode
358 (Chanthakoummane et al., 2016; Khurat & Sawangphol, 2019).
359   The protocol field within the rule header is required and usually defaults to IP. Nonetheless,
360 it equally supports UDP, TCP, and ICMP options. The source IP is an optional field and, by
361 default, is set to *any*, as indicated in Table 1. However, it also supports a single IP address like
362 172.168.10.102 or a CIDR block like 172.168.10.0/16, which permits a range of IP addresses as
363 an input. Likewise, a source port field allows a port number or range of port numbers, and the
364 destination IP and port fields are almost the same as the source IP and port fields. Finally, the
365 direction of the monitored traffic is specified using the directional operators (->, and <-), and the
366 monitoring of source to destination (->) is the most common practice (Khurat & Sawangphol,
367 2019).

368   Table 2 presents the rule header components with typical examples. For example, the first
369 example will trigger an alert for traffic from *any* source address with various port numbers up to
370 and including 1024 denoted (:1024), which is going to 192.168.100.1, and ports that are greater
371 than and including port 600 denoted (600:). Finally, the second and last examples of Table 2 are
372 almost the same as the first except for the change in specific fields.

The Rule Options

374   The rule option unit comprises the detection engine's central functionalities yet provides
375 complete ease of use with various strengths and flexibility (Khurat & Sawangphol, 2019).
376 However, this segment can only be processed if all the preceding details have been matched.
377 Furthermore, since this unit generally requires a vast amount of processing resources and time, it
378 is recommended to limit the scope of the rules using only the necessary fields to enable real-time
379 processing without packet drops. Therefore it is recommended to only use the *message*, *content,*
380 and *SID* field for writing efficient and reliable rules. The semicolon (;) separates the rule options
381 while the option keywords are separated using the colon (:). Finally, this unit comprises four

382   main classes, but this paper will only summarize the general and payload detection options. For
383   instance, the general options have no significant effect during detection but merely provide
384   statistics about the rule, whereas the payload inspects the packet data. The general rule and
385   payload detection options include numerous parameters available to Snort users for rule creation
386   (Khurat & Sawangphol, 2019). However, only the relevant ones are selected to understand the
387   work proposed in this paper, and Table 3 presents the selected options with a typical example for
388   each.

389   **Snort Configurations and Rule Files**
390   Snort provides a rich scope of customizable configuration options for effective deployment
391   and day-to-day operations. Since Snort consists of vast configuration options, this section will
392   only highlight the necessary options to understand this work easily. Generally, the *snort.conf*
393   contains all the Snort configurations, and it includes the various customizable settings and
394   additional custom-made rules. The *snort.conf* is a sample and default configuration file shipped
395   with the Snort distribution. However, users can use the -c command line switch to specify any
396   name for the configuration file, such as */opt/snort/Snort -c /opt/snort/myconfiguration.conf*.
397   Nevertheless, *snort.conf* is the conventional name adopted by many users (Erlacher & Dressler,
398   2020). Likewise, the configuration file can also be saved in the home directory as *.snortrc* but
399   using the configuration file name as a command-line is the common practice with advantages.
400   Moreover, it also enables using variables for convenience during rule writing, such as defining a
401   variable for HOME_NET within the configuration file like *var HOME_NET 192.168.10.0/24*.
402   The preceding example enables the use of HOME_NET within various rules, and when a change
403   is needed, the rule writer only needs to change the variable's value instead of changing all the
404   written rules. For instance, *var HOME_NET [192.168.1.0/24,192.168.32.64/26]* and *var*
405   *EXTERNAL_NET any* (Mishra et al., 2016).
406   Lastly, the rules configuration also enables the Snort users to create numerous customized
407   rules using the variables within the configuration files and add them to the *snort.conf* file. The
408   general convention is to have different Snort rules in a text file and include them within the
409   *snort.conf* using the *include* keyword like *include $RULE_PATH/myrules.rules*, which permits
410   the inclusion of the rules within *myrules.rules* to the *snort.conf* file during the next start of snort.
411   Also, users can use the Snort commenting syntax (#) in front of a specified rule or the rule file
412   within the Snort configuration file to manually disable the Snort rule or the entire class of rules.
413   Similarly, Snort parsed all the newly added rules during a Snort startup to activate any newly
414   added rules. However, if there are any errors within the newly added rules, Snort will exit with
415   an error, necessitating the correct and consistent writing of Snort rules (Erlacher & Dressler,
416   2020; Mishra et al., 2016). Finally, Figs. S1 and S2 are the standard representations of the Snort
417   configuration and rule files.

418   **Synopsis of Alert Correlation**
419   Alert Correlation is a systematic multi-component process that effectively analyzes alerts
420   from various intrusion detection systems. Its sole objective is to provide a more concise and

421   high-level view of a given computer system or network, and it can prioritize IDSs alerts based on
422   the severity of the attack. Additionally, it plays a significant role for the network administrators
423   to effectively and efficiently differentiate between relevant and irrelevant attacks within a given
424   network, resulting in reliable and secured networks (Valeur et al., 2017).

425 ## MATERIALS AND METHODS

426   The section briefly discusses the methodologies and procedures used to design and
427   implement the proposed Snort Automatic Rule Generator using a sequential pattern algorithm
428   and the Security Event Correlation, abbreviated SARG-SEC. It is worthy to note that the term
429   event is the same as alerts generated during intrusion detection.

430 ### Snort Automatic Rule Generation (SARG)

431   Irrespective of the significant numbers of available literature that use other approaches of
432   automating rule generation with remarkable success (Li et al., 2006; Ojugo et al., 2012; Sagala,
433   2015), there is still a need for an effective and optimized auto-rule generator. According to the
434   authors' knowledge, none of the existing literature uses the proposed approach in this paper.
435   Therefore, this paper presents an automation method of generating content-based Snort rules
436   from collected traffic to fill this research gap. The following sections provide a concise
437   description of the design and implementation procedures.

438 ### *Content Rule Extraction Algorithm*

439   Snort rules can indicate various components, but this research targets rules that only include
440   header and payload information. The steps demonstrated in the examples below are performed to
441   generate the Snort rules automatically. For each first host, the application and service to be
442   analyzed or the traffic of malicious code is collected. Moreover, packets with the same
443   transmission direction are combined to form a single sequence in a single flow. Finally, the
444   sequence is used as an input of the sequential pattern algorithm to extract the content (Pham et
445   al., 2018).

446   *Action Protocol SourceIP SourceProt —> DestinationIP DestinationPort (Payload)*

447   *Alert tcp 192.168.0.0 192.168.0.0/24 —>IP 80 (content: "cgi-bin/phf"; offset:6; depth:24)*

448   The sequential pattern algorithm finds the candidate content while increasing the length
449   starting from the candidate content whose length is 1 in the input sequence and finally extracts
450   the content having a certain level of support. Nevertheless, the exclusive use of only packet or
451   traffic contents for rule creation can lead to significant false positives, thereby undermining the
452   effectiveness of the created rules and increasing the potential risks against computer systems and
453   networks. Therefore, additional information is analyzed and described in the rule to enable
454   efficient and effective rules against malicious attacks. As a result, this paper used the content and
455   header information of the network traffic or packets as supplementary information, thereby
456   significantly enhancing the reliability of the automatically generated rules at the end of the
457   process. Moreover, by applying the extracted content to input traffic, traffic matching the content

458   is grouped, and the group's standard header and location information is analyzed (Sagala, 2015).
459   Finally, the auto-generated Snort rules are applied to the network equipment with Snort engine
460   capability.

461   ***Sequence Configuration Steps***
462   The sequence is constructed by only extracting the payloads of the packets divided into
463   forward and backward directions of the flow. Moreover, the proposed approach will generate
464   two sequences for flows that consist of two-way communication packets, whereas the
465   unidirectional communication traffic generates a single sequence. Finally, it is worthy to note
466   that a sequence set consists of several sequences denoted **S** as shown in equation 1, and a single
467   sequence is made up of host ID and string as shown in equation 2.

468   $\text{SequenceSet} = \{S_1, S_2, ..., S_s\}$ (1)
469   $S_i = \{\text{host\_id}, <a_1a_2a_3, ..., a_n>\}$ (2)

470   ***Content Extraction Step***
471   A sequence set and a minimum support map are input in the content extraction step and
472   extract content that satisfies the minimum support. This algorithm improves the Apriori
473   algorithm, which finds sequential patterns in an extensive database to suit the content extraction
474   environment. The Content Set, which is the production of the algorithm, contains several
475   contents (C) as shown in Equation (3), and one content is a contiguous substring of a sequence
476   string as shown in Equation (4). Algorithm 1 and Algorithm 2 illustrate producing a content set
477   that satisfies a predefined minimum map from an input sequence set. Moreover, when Algorithm
478   1 performed the content extraction, the content of length one (1) is extracted from all sequences
479   of the input sequence set and stored in a content set of length 1 (L_1) as demonstrated in (Algo.1
480   Line: 1~5), the content of length 1 starting with the length is increased by 1. Also, the content of
481   all lengths is extracted and stored in its length content set (L_k) as shown in (Algo.1 Line: 6~20).

482   $\text{ContentSet} = \{C_1, C_2, ..., C_c\}$ (3)

483   $C_i = \{<a_xa_x+1 \ .... \ a_y> \mid 1 \leq x \leq y \leq n,\}$ (4)

484   $\text{Support} = \text{Number of support hosts / total number of hosts}$ (5)

485   It is worthy to note that the method to be used at C (Algo. 1.0 Line: 18) is the process
486   described in Algorithm 2.0.

487   The contents of the set L_{k-1} are created by comparing the contents of L_k. Furthermore, to
488   make the content of the set L_{k-1} by combining the contents of the set L_k, contents of the set with
489   the same length *k-2* content excluding first character and *k-2* content excluding the last character
490   are possible as shown at (Algo.2.0 Line: 1~7). For example, *abcd* and *bcde* are the contents of
491   the set *L4*. It has the same *bcd* except for *a,* and *bcd* except for *e.* Therefore, the content *abcde* of
492   the set L5 can be created by increasing the length by 1 in the same way as above. Extracting and
493   deleting content below the support level are repeated until the desired outcome.

494   The final step checked for the content inclusion relationship of all lengths extracted. If we
495   find the content in the inclusion relationship, the content is deleted from the set as demonstrated
496   in (Algo.1.0). Then, we deliver the final created content set to the next step. Moreover, the
497   SequenceSet consisting of traffic from 3 hosts and minimum support of 0.6 is passed as an input.
498   The minimum support rating of 0.6 means that since the total number of hosts is 3, the content
499   must be observed in traffic generated by at least two hosts. Finally, all length one (1) content is
500   extracted when the algorithm is executed.
501   Furthermore, Algorithm 3.0 would have been given content and packet set when it represents
502   analyzing the location information of the content. The output of this algorithm is offset when
503   matched to a packet in a packet set. The matching starts in a bit, byte position, and depth is the
504   matching exit position, meaning the maximum byte position.

505   The first offset is the maximum size of the packet, and the depth initializes to 0 as shown in
506   (Algorithm 3.0 Line: 1~7), which traverses all packets in packet set and adjusts offset and depth.
507   Moreover, it checks whether the content received matches the packet, and if there exists a match,
508   then the starting byte position is obtained and compared to the current offset. If the value is less
509   than the current offset, the value changes to the current offset. Similarly, in the case of depth, if
510   the value is more significant than the current dept., the current value is obtained using the byte
511   position. Then it changes the value to the current depth (Algo.3 Line: 4~6). Likewise, to analyze
512   the header information of the extracted content, we performed a process similar to the location
513   information.
514   The analysis steps described above traversed all packets in the packet set and checked for
515   possible matching with the content. If a match exists, it will store the packet's header
516   information. After reviewing all packets, it adds the header information to the content rule if the
517   stored header information has one unique value. Additionally, in IP addresses, the CIDR value is
518   reduced to 32, 24, 16 orders which iterate until the unique value is extracted. Assuming the
519   CIDR value is set to 32, which is the class D IP address range, we will try to find an exclusive
520   value, and if not found, then apply the CIDR value to 24 to find the class C IP address, and this
521   process continues until we obtained the desired values. For instance, if the Destination IP address
522   to which that content is matched is CIDR 32, then "111.222.333.1/32" and "111.222.333.2/32"
523   are extracted. In contrast, it can also be set to CIDR 24 and extract "111.222.333.0/24".

524   **The Schematic Diagram of the Snort Automatic Rule Generator (SARG)**
525   Fig. 2 presents a comprehensive schematic diagram of SARG. The proposed auto-rule
526   generator (SARG) utilized well-known datasets used by many researchers and security experts to
527   assess various network security frameworks as discussed in (Lippmann et al., 2000). Initially, we
528   used different pcap files to simulate live attacks against SARG that facilitates an efficient and
529   effortless generation of reliable Snort rules. It is essential to note that π_{-1}, π_{-2}, π_{-3},...,π_{-n}, denotes
530   the various utilized pcap files for the auto-rule generation. Consequently, SARG relies on the
531   pcap file contents to automatically generate numerous effective Snort rules without any human
532   intervention, and λ_{-1}, λ_{-2}, λ_{-3}, λ_{-4},...,λ_{-n} represents the various auto-generated Snort rules. Next,

533 the Snort.conf file is updated based on the auto-generated rules. In addition, any device with a
534 Snort engine denoted as $\ddot{G}$ can use these auto-generated rules against incoming traffic
535 represented as D-$_1$,..., D-$_n$ to trigger alerts for malicious attempts that meet all the criteria of the
536 rules. Finally, this paper does not document the generated alerts due to the volume of the work.
537 However, the provided supplementary materials contain the codes and pcap files that any
538 interested researcher can reproduce.

539 **Overview and Significance of Alert Correlation**
540   Intrusion detection systems have recently witnessed tremendous interest from researchers due
541 to their inherent ability to detect malicious attacks in real-time (Vaiyapuri & Binbusayyis, 2020;
542 Zhou et al., 2020). In addition, it has significantly mitigated the security challenges that came
543 with ground-breaking technologies such as the Internet of Things (IoT) (Verma & Ranga, 2020),
544 big data, and artificial intelligence (Topol, 2019). However, regardless of the immense
545 contributions mentioned above, it may still generate thousands of irrelevant alerts daily, which
546 complicates the role of the security administrators in distinguishing between the essential and
547 nonessential alerts such as false positives.

548   In addressing these security challenges, specifically for IDSs, the research industry has
549 proposed significant pieces of literature with remarkable achievements in lessening the huge
550 false alarm rates of current cutting-edge proposed systems (Mahfouz et al., 2020; Zhou et al.,
551 2020). Nevertheless, irrespective of these significant achievements, there is still a need to
552 propose new approaches with much efficiency and effectiveness to help mitigate the vast false
553 alarm rates of the earlier proposed systems (Jaw & Wang, 2021). Therefore, this paper presents a
554 novel alert correlation model that correlates and prioritizes IDS alerts**.** The following sections
555 provide a succinct description of the design and implementation process of the proposed security
556 event correlation model.

557 **The Proposed Security Events Correlation (SEC)**
558   The SEC model offers efficiency and effectiveness of correlating the alerts generated by the
559 IDS, which significantly mitigates the massive false alarm rates. Furthermore, it does not need
560 previous knowledge while comparing different alerts to measure the similarity in various attacks.
561 Unlike other correlation approaches that usually follow specific standards and procedures
562 (Valeur et al., 2017; Zhang et al., 2019), we proposed a model that entails the same processes but
563 with different approaches, as shown in the overview of the proposed model below.

564   The overview of the proposed system involves various phases such as monitoring interval,
565 alert preprocessing, alert clustering, correlation, alert prioritization, and the results as shown in
566 Fig. 3. The following sections detail the various phases of SEC with their respective procedures.

567 *The Monitoring Interval*
568   In summary, this paper defined the monitoring interval as the specific configured time on the
569 security framework or monitoring model performing the alert correlation. The monitoring

570 interval of SEC is 300 days. It is essential to note that the long interval of 300 days is because
571 more days means extra alerts to correlate and aggregate, leading to effective and better results.

572 ***Alert Preprocessing***
573   The alert preprocessing stage of this work involves two sub-processes, namely, feature
574 extraction and selection. This phase systematically extracts features and their equivalent values
575 from the observed alerts. The similarity index defined below calculates the alerts with the same
576 value for existing features such as attack category, detection time, source IP address, and port
577 number. Consequently, it significantly helps identify new features from the available alerts,
578 thereby overcoming attack duplication. Also, this phase involves the selection of features for
579 alert correlation. For example, suppose the selected features such as the attack detection time,
580 category, port number, and source IP address of one alert or multiple alerts are the same. In that
581 case, we use the similarity index to select the relevant alerts and eliminate irrelevant ones such as
582 duplications and similar instances. Thus, Π denotes the similarity between two alerts, and G
583 represents the alert similarity index, and δ denotes an alert.

584 $\Pi = \text{G} (\delta a, \delta b)$, where $\delta a \neq \delta b$ (6)

585   Lastly, this section also involves alert scrubbing that uses attack type, detection time, source
586 IP address, and detection port to remove the incomplete alert data. This process plays a
587 significant role in providing reliable and consistent data.

588 ***Alert Aggregation and Clustering***
589   The existing literature has provided various descriptions of alert aggregation, such as
590 considering alerts to be similar if all their attributes match but with a bit of time difference. In
591 contrast, others extended the concept to grouping all alerts with the exact root causes by
592 aggregating alerts using various attributes. Moreover, this phase groups similar alerts based on
593 the similarity index of the extracted features. For instance, if the features of alert $\tilde{Y}$ have the
594 same value as alert $\ddot{X}$, then this phase will automatically convert these two alerts into a single
595 unified alert by removing the duplicated alerts, thereby significantly reducing the number of
596 irrelevant alerts. Finally, the clustered alerts provide an effective and efficient analysis of false
597 positives, leading to reliable and optimized security frameworks.

598   Lastly, Fig. 4 demonstrates the alert aggregation process. Firstly, we cluster all the generated
599 alerts, check their similarity using the similarity index defined in previous sections, and remove
600 all the duplicated alerts.

601 ***Correlation***
602   Based on existing literature, the primary objective of alert correlation is to identify
603 underlying connections between alerts to enable the reconstruction of attacks or minimize having
604 massive irrelevant alerts. Furthermore, scenario-based, temporal, statistical, and rule-based
605 correlations are among the most commonly utilized correlation categories in existing research.
606 The approach in this paper has fully met the attributes of the statistical correlation method, which

607   correlates alerts based on their statistical similarity. Finally, this phase consists of the alert
608   correlation process based on selected features like the IP addresses, attack category, and the
609   detected time of the generated alerts. For example, if the detection time of two independent alerts
610   satisfies the condition of COTIME denoted as $\mathbb{G}$, then the correlation is performed based on
611   selected features.

612   Fig. 5 presents the correlation engine with the extracted features for alert correlation. The
613   correlation engine used these features to efficiently correlate alerts that meet the conditions of the
614    $\mathbb{G}$.

615   ***Alert Prioritization***
616   Alert prioritization is the final phase of the proposed SEC model. It plays a significant role in
617   prioritizing the alert's severity, thereby helping the administrator identify or dedicate existing
618   resources to the most alarming malicious attacks. Although a priority tag is assigned to each
619   Snort rule, as explained earlier, we intend to extend its functionality by embedding an alert
620   prioritization within the proposed SEC model. Alert prioritization is classified into high,
621   medium, and low priority. Firstly, this paper defines the high priority as the alert counts with
622   standard features such as the data fragmentations, source IP address, port number, and attack
623   category.
624   For instance, if multiple alerts have the same destination port numbers and IP addresses, then
625   we classify these alerts as high-priority alerts. Lastly, the high priority functionality will hugely
626   minimize various challenging cyber-attacks such as DDoS and DoS because it handles the
627   standard techniques utilized in these attacks, like the IP fragmentation attack, which uses the
628   analogy of data fragmentation to attack target systems. Secondly, the medium priority alerts
629   count the number of alerts with shared features such as the same attack category, IP addresses,
630   and destination port number. Finally, low priority alerts are alert counts with standard features
631   like IP addresses, attack category, and destination port but varying values.

632   The alert prioritization categories with the various standard features discussed above are
633   shown in Fig. 6. However, it is essential to note that irrespective of the common features in each
634   category, the above section entails specified conditions that uniquely distinguish them.

635   **Overview of the Proposed Security Events Correlation (SEC)**
636   Fig. 7 demonstrates the detailed overview of the proposed SEC. First of all, SEC accepts a
637   collection of raw alerts as an input generated based on a specified monitoring interval set to 300
638   days to ensure sufficient alerts for practical analysis. Additionally, the alert preprocessing step
639   accepts these raw alerts and performs feature extraction and selection. Furthermore, the alert
640   scrubbing phase uses predefined conditions denoted as Ξ, to check for alert duplicates. If
641   duplicates exist, the scrubbing process will remove all the copies and send the alerts with no
642   duplicates to the correlation engine. Next, the correlation engine decides if the alerts are single or
643   multiple instances using predefined conditions denoted as $<=\mathbb{G}$. Finally, the alert prioritization
644   phase prioritizes the alerts using the above three categories, and the output phase presents the

645   relevant alerts as the final results. It is worthy to note that the alert outputs are the actual events
646   after removing the duplicates and the alert correlation, as shown in the equation below.

647   $\Omega = \pi - \beta$ (9)

648   Where Ω denotes the final output alerts, π represents the total number of alerts, and β denotes
649   the correlated alerts.

RESULTS AND DISCUSSION

651   This section presents a comprehensive systematic analysis and performance justification of
652   the proposed models (SARG-SEC). For example, it evaluates how well the SARG can efficiently
653   generate standard and reliable Snort rules by executing SARG against live attacks in existing
654   pcap files. Finally, this section will also highlight the results and performance evaluation of the
655   (SEC) model that significantly mitigates the challenges of the vast alerts generated by the Snort
656   IDS and the earlier proposed feature selection and ensemble-based IDS (Jaw & Wang, 2021).

Evaluation of the Generated Snort Rules by the proposed SARG

658   The authors of (Lippmann et al., 2000) provided a descriptive and intriguing challenge of
659   documenting an off-line IDS dataset that a wide range of security experts and researchers heavily
660   utilized to assess various security frameworks. Moreover, the paper presents a testbed that used a
661   tcpdump sniffer to produce pcap files used to evaluate the proposed solutions (SARG-SEC).
662   Finally, the supplementary pcap files also entail the various attacks utilized to evaluate SARG-
663   SEC. Accordingly, this section highlights the findings of the various conducted experiments to
664   accurately assess the performance of the SARG framework, as demonstrated in the figures
665   below.
666   The proposed SARG method has achieved decent performances on the auto-generation of
667   Snort rules, as illustrated in Fig. 8. All the findings presented in this section use various pcap
668   files as a simulation of live attacks against the proposed method to generate efficient and
669   effective Snort rules that completely meet all the criteria of the Snort rule syntax, as presented in
670   Table 3. For instance, Fig. 8 demonstrated that SARG has successfully auto-generated a Snort
671   rule with the following content: *alert udp 10.12.19.101 49680 -> any any (msg: "Suspicious*
672   *IP10.12.19.101 and port 49680 detected"; reference:Packet2Snort; classtype:trojan-activity,*
673   *sid:xxxx, rev:1)*. Based on the above auto-generated Snort rule, it is self-evident that SARG has
674   produced optimized Snort rules that meet all the criteria of Snort rule syntax as discussed in
675   (Khurat & Sawangphol, 2019). For example, the above auto-generated rule has produced a
676   descriptive and easy to analyze message of (*msg: "Suspicious IP10.12.19.101 and port 49680*
677   *detected ";),* which details the reason or cause of the alert.
678   Furthermore, SARG auto-generated another similar Snort rule by setting the source IP
679   address and port number to "*any any"* and the destination IP address and port number
680   as *10.12.19.1* and *53*, respectively. Lastly, the findings presented in Fig. 8 shows that SARG has
681   auto-generated an impressive and comprehensive Snort rule that meets all standard Snort rule
682   criteria. However, the auto-generated alert uses the *HOME_NET* variable as the source address

683   and a *msg*, *content*, *dept*, and *offset* values of *(msg: "Suspicious DNS request for fersite24.xyz.*
684   *detected"; content:" |01000001000000000000|"; depth:10; offset:2;)*, respectively. Based on the
685   above performances, the authors concluded that SARG has effectively met all the criteria of
686   creating compelling and optimized Snort rules consisting of numerous general rule parameters
687   and payload detection options (Khurat & Sawangphol, 2019).

688   Moreover, Fig. 9 presents a series of auto-generated Snort rules using several pcap files as
689   live attacks. All the auto-generated Snort rules shown in Fig. 9 have completely demonstrated to
690   meet all the standards of Snort rule creation, as discussed in much existing work (Jeong et al.,
691   2020; Khurat & Sawangphol, 2019). For instance, the use of defined variables like
692   the *HOME_NET* and ASCI values
693   of *("|05|_ldap|04|_tcp|02|dc|06|_msdcs|0C|moondustries|03";)* for the values of *"content:"* field
694   of the auto-generated Snort rules. Also, all the automatically generated rules came with auto-
695   generated messages that uniquely and vividly describe the intrusive or abnormal activity
696   whenever the alert is triggered. Consequently, this will significantly help the administrator to be
697   able to easily identify why the alert happened and quickly find solutions to mitigate any
698   malicious activities.
699   Additionally, all the auto-generated Snort rules are saved in the *local.rules* to ensure
700   consistency and avoid duplications of rules within the *local.rules* files, and then included in
701   the *snort.conf* file. As a result, it provides the administrator with the flexibility to further fine-
702   tune the generated rules to meet the specific needs of a given computer network or system. Thus,
703   irrespective of the additional efforts of fine-tuning the auto-generated rules, for example,
704   manually updating the *sid* value of the auto-generated rules and other fields for specificity,
705   SARG has undoubtedly minimized the time taken for Snort rule creation. Also, it can
706   significantly lessen the financial burden of human experts for Snort rule creation, thereby making
707   the proposed method a meaningful tool that would play a significant role in mitigating the
708   escalating cyberattacks.

709   In conclusion, the results presented in Fig. 8 and Fig. 9 have impressively demonstrated the
710   capability of auto-generating Snort rules, which considerably mitigates the need for costly
711   human capacity in creating Snort rules. However, it has downsides that we could not handle,
712   such as automatically generating the consistent *sid* values of the respective rules. As a result, we
713   came up with a prompt message to remind the users to update the *sid* value after the auto-
714   generation of the Snort rule. Also, based on the lack of expert domain, we could not justify why
715   SARG has two contents in a single generated Snort rule. Nevertheless, these challenges have no
716   negative implications while validating the auto-generated rules. Regardless, we intend to extend
717   this research to establish means of solving these research challenges.

718   **Analysis of the Single and Multiple Instances Security Event Correlations**
719   Likewise, the following sections present the results of numerous experiments that evaluate
720   the consistency and efficiency of the proposed security event correlator (SEC). Also, the findings
721   presented in the subsequent sections have demonstrated the promising performances of the SEC

722   model, which could significantly mitigate the substantial challenges of managing the vast alerts
723   generated by heterogeneous IDSs.

724   Firstly, Figs. 10 and 11 summarized the results of the experiments based on single and
725   multiple instances. For example, Fig. 10 depicts a single sample of one thousand (1000) alerts
726   used to evaluate the effectiveness of the proposed security event correlator (SEC), which
727   obtained an impressive correlation performance of removing 71 irrelevant alerts due to alert
728   duplication. Even though the 71 removed irrelevant alerts might seem insignificant, the authors
729   believed this is an excellent performance considering the dataset sample. Also, it will be fair to
730   state that processing these irrelevant alerts will waste valuable computational and human
731   resources. Similarly, Fig. 11 shows the findings of multiple instances ranging from 1000 to 4000
732   alerts as the sample sizes for the individual evaluations. Again, the results demonstrated that SEC
733   had achieved a decent correlation performance on the various sets.
734   For instance, Fig. 11 shows that out of 4000 alerts, SEC efficiently and effectively
735   compressed it to only 3710 alerts by removing a whopping 290 irrelevant alerts that could have
736   unnecessarily exhausted an organization's valuable resources, such as human and computational
737   resources. Likewise, SEC replicates a similar performance for the 2000 and 1000 alert samples,
738   eliminating a vast 250 and 71 irrelevant alerts for both instances, respectively. Therefore,
739   considering the above significant performance of SEC for both single and multiple instances, we
740   can argue that the proposed method could significantly contribute to the solutions of analyzing
741   and managing the massive irrelevant alerts generated by current IDSs.

742   **Assessment of Single Instance with Multiple Correlation Time (COTIME)**
743   This section presents numerous experiments based on a single number of inputs with varying
744   correlation times (COTIME(s)). It also further assess the performance and consistency of the
745   proposed security event correlator based on alert prioritization that efficiently demonstrated the
746   number of high, medium, and low priority alerts. Furthermore, it shows the correlation time and
747   the correlated alerts using a fixed sample size of 1000 alerts.

748   The findings presented in Figs. 12 and 13 summarized the evaluation results obtained from
749   four experiments, highlighting some exciting results. For instance, experiments one and two
750   illustrated in Fig. 12 show that using a fixed sample of 1000 alerts and a COTIME of 40 seconds,
751   SEC efficiently identified 71 alerts as correlated alerts, producing a manageable 929 alerts as
752   final relevant alerts. Similarly, out of the 929 alerts, SEC effectively categorized a considerable
753   334 alerts as high priority alerts, 209 alerts as medium alerts, and 386 low priority alerts.
754   Likewise, experiment two presented in Fig. 12 indicates that increasing the number of COTIME
755   to 60 seconds results in even better findings, such as identifying a massive 117 alerts as
756   correlated alerts, thereby leading to only 883 alerts as the final output alerts. Unlike experiment
757   one, only 163 alerts were categorized as medium priority alerts, while 334 and 386 were
758   recorded for high and low alerts, respectively.
759   Moreover, Fig. 13 further validates that an increase in COTIME is a crucial factor in the
760   performance of the SEC. For example, increasing the COTIME to 90 and 120 seconds results in

761 more correlated alerts like 124 and 128 alerts for 90 and 120 seconds, respectively. Also,
762 experiments three and four have effectively identified 157 and 154 alerts as medium alerts. The
763 final output alerts for the two experiments are 876 and 872 alerts, respectively, with a low
764 priority alert of 385. However, the values for high priority alerts for all these four experiments
765 presented in Figs. 12 and 13 recorded the same values. Nonetheless, we have conducted a series
766 of experiments to validate these similarities.

767 Similarly, the results illustrated in Figs. 14 and 15 shows that SEC obtained impressive
768 findings such as an increase of 133 and 165 correlated alerts for experiments five and six with a
769 COTIME of 180 and 300 seconds, respectively. Again, this further validates that COTIME
770 significantly correlates with the number of correlated outputs. Moreover, SEC categorized 150
771 and 131 alerts as medium alerts, while 383 and 374 alerts were low priority alerts with only 867
772 and 835 final output alerts for experiments five and six. Nevertheless, like the previous
773 experiments, the values for high priority remain as 334 alerts for both experiments, as shown in
774 Fig. 14.

775 Finally, and most importantly, Fig. 15 presents some notable and exciting findings that reveal
776 interesting correlations among the chosen factors of the conducted experiments. For instance,
777 experiment seven obtained a massive 474 correlated alerts due to a 600 seconds increase of
778 COTIME. As a result, it resulted in a much proportionate distribution of alerts into various
779 categories like 223 high priority alerts, 227 low priority alerts, a negligible 78 medium alerts
780 with only 526 final output alerts. Similarly, SEC achieves exciting results when COTIME is 900
781 seconds, such as a manageable final output of 424 alerts dues to the vast 576 correlated alerts.
782 Also, experiment eight presented in Fig. 15 effectively and efficiently categorized the final
783 output into 178, 65, and 181 alerts for high, medium, and low priority, respectively. As a result,
784 it would be fair to conclude that SEC can significantly assist the network or system
785 administrators with a considerably simplified analysis of alerts. Moreover, based on the results
786 presented above, it is self-evident that while COTIME increases, the number of output alerts
787 decreases. Therefore, we can conclude that the number of correlated alerts and COTIME are
788 directly proportional to each other.

789 **Assessment of the Time Factor on Alert Correlations**
790 Similarly, this section meticulously conducted more experiments to evaluate the correlation
791 of time factors and the performance of the proposed unique approach (SEC). Considering that
792 the amount of COTIME has significantly influenced the outcomes of alert correlation achieved
793 by SEC, this section intends to validate this apparent relationship.

794 The results illustrated in Figs. 16 and 17 present the evaluation performance of correlation
795 time against the correlated alerts with some interesting findings. For instance, Fig. 16 shows the
796 time lag or time complexity on static COTIME with multiple instances ranging from 1000 to
797 4000 alerts. Moreover, time complexity or lag is the time difference between various sample
798 inputs. Similarly, Fig. 16 confirms a continuous increase in the time (COTIME) as the value of

799   the inputs increases, which validates the results presented in Figs. 13, 14, and 15. For instance,
800   increasing the inputs to 1500, 2000, and 2500 alerts has increased COTIME to 42, 45, and 48
801   seconds, respectively. Likewise, the sample of 3500 and 4000 alerts recorded 52 and 55 seconds
802   of COTIME, respectively. Based on these performances, the authors have concluded that
803   COTIME is a crucial factor in SEC's better performance, as shown in Fig. 16 and preceding
804   evaluations.

805   Moreover, Fig. 17 shows a more compelling assessment of COTIME with correlated alerts.
806   For instance, there is a continuous rise in correlation time and the correlated alerts, like it takes
807   40 seconds COTIME to correlate 71 correlated alerts. Also, the increase of COTIME to 90, 120,
808   and 182 seconds has achieved 117, 124, and 128 correlated alerts, respectively. Furthermore, the
809   considerable rise of COTIME to 300, 600, and 900 seconds has achieved some interesting
810   findings such as 165, 474, and 576 correlated alerts, which is a significant and impressive
811   performance for SEC. Based on the above results, it will be fair to conclude that a massive
812   increase in COTIME could lead to reliable and effective alert correlation. However, this could
813   also lead to the demand for more processing power and other similar burdens. Irrespective of
814   these challenges, SEC has achieved its objective of significantly minimizing the massive
815   irrelevant alerts generated by heterogeneous IDSs. Also, SEC has enabled easy analysis of alerts,
816   which has always been a considerable challenge for system and network administrators.

Conclusions

818   Data security has been a massive concern over the past decades due to the considerable high-
819   tech progression that has positively influenced human society in many aspects. However, the
820   illegal mining of data due to the vulnerabilities of security mechanisms has enabled malicious
821   users to compromise and exploit the integrity of existing systems, thereby causing colossal havoc
822   to individuals, governments, and even private sectors. Consequently, this leads to the necessity to
823   deliver reliable and effective security mechanisms by harnessing various techniques to design,
824   develop and deploy optimized IDSs, for example, Snort-based IDS. Nonetheless, existing studies
825   have shown that manually creating Snort rules, which is highly stressful, costly, and error-prone,
826   remain a challenge.
827   Therefore, this paper proposed a practical and inclusive approach comprising a Snort
828   Automatic Rule Generator and a Security Event Correlator, abbreviated SARG-SEC. Firstly, this
829   paper provides a solid and sound theoretical background for both Snort and alert correlation
830   concepts to enlighten the readership with the essential idea of understanding the presented
831   research content. Additionally, this paper presents an efficient and reliable approach (SARG) to
832   augment the success of Snort, which significantly minimizes the stress of manually creating
833   Snort rules. Moreover, SARG utilizes the contents of various pcap files as live attacks to
834   automatically generate optimized and effective Snort rules that meet the entire criteria of Snort
835   rule syntaxes as described in existing literature (Khurat & Sawangphol, 2019). The results
836   presented in this paper have achieved impressive performances in the auto-generation of Snort
837   rules, with little knowledge of how the contents of the rules are generated. Furthermore, the auto-

838 generated Snort rules could serve as a beginning point for turning Snort into a content defense
839 method that considerably lessens data leakages.

840   Moreover, this paper posits an optimized and consistent Security Event Correlator (SEC) that
841 considerably alleviates the current massive challenges of managing the immense alerts
842 engendered by heterogeneous IDSs. This paper evaluated SEC based on single and multiple
843 instances of raw alerts with consistent and impressive results. For example, utilizing a single
844 sample of 1000 alerts and multiple instances of 1000 to 400 alerts, SEC effectively and
845 efficiently identified 71 and 290 alerts as correlated alerts. Furthermore, this paper uses a single
846 instance of 1000 alerts with varying COTIME to further measure the performance and stability
847 of SEC based on alert prioritization that competently revealed the number of high, medium, and
848 low priority alerts. For example, SEC achieved excellent performances on a single instance of
849 1000 alerts like 133 and 165 correlated alerts for experiments five and six with a COTIME of
850 180 and 300 seconds, respectively. Furthermore, SEC identified 383 and 374 low priority alerts,
851 whereas 150 and 131 alerts are medium alerts with only 867 and 835 final output alerts for
852 experiments five and six.

853   Lastly, to further confirm the significant correlation between the number of correlated
854 outputs and COTIME. Experiments seven and eight present exciting results that demonstrated
855 some intuitive relationships amongst these chosen factors. For instance, an increase of COTIME
856 to 600 and 900 seconds shows a much balanced and acceptable alert prioritization into numerous
857 categories like final output alerts of only 526, 227 low priority alerts, 223 high priority alerts,
858 and a negligible 78 medium alerts. Based on the above findings, it would be fair to conclude that
859 SARG-SEC can considerably support or serve as a more simplified alert analysis framework for
860 the network or system administrators. Also, it can serve as an efficient tool for auto-generating
861 Snort rules. As a result, SARG-SEC could considerably alleviate the current challenges of
862 managing the vast generated alerts and the manual creation of Snort rules.

863   However, notwithstanding the decent performance of the SARG-SEC, it has some apparent
864 downsides that still necessitate some enhancements. For instance, the auto-generation of
865 consistent *sid* values of the respective rules and why SARG has two contents in a single
866 generated Snort rule. Similarly, we acknowledged that the sample sizes of alerts and the apparent
867 relationship of correlated alerts with COTIME could challenge our findings. Nevertheless, we
868 aim to extend this research to establish means of solving these research challenges. In the future,
869 we intend to: (i) Extend SARG's functionality to efficiently generate consistent *sid* values for the
870 auto-generated Snort rules and establish concepts of how SARG automatically generated the
871 contents of the rules. (ii) Investigate the apparent relationship of COTIME and correlated alerts
872 and present solutions to how we can correlate a larger sample size of alerts within an acceptable
873 time frame to mitigate the need for unceasing computing resources. (iii) Finally, evaluate SEC
874 within a live network environment instead of pcap files and auto-generate reliable and efficient
875 Snort rules using the knowledge of the proposed anomaly IDS (Jaw & Wang, 2021).

876 ## Acknowledgment

877   Special thanks to Professor Wang Xue Ming for the guidance and supervision. Similarly, the
878 authors are grateful to the editor and reviewers for their valuable comments and suggestions.
879 Lastly, Sherriffo Ceesay, and Eva Lee Redding, thanks for the insightful comments, motivation,
880 and advice.

881 **ADDITIONAL INFORMATION AND DECLARATIONS**

882

883 **Data Availability**
884 The raw code can be available upon request or can be found as a
885 https://github.com/KhadimJr/SARG-SEC.git

886 **Supplemental Information**
887 Supplemental information for this article is submitted with the manuscript

888 ## References

889 Aickelin, U., Twycross, J., & Hesketh-Roberts, T. (2007). Rule generalisation in intrusion
890   detection systems using Snort. *International Journal of Electronic Security and Digital*
891   *Forensics*, *1*(1), 101–116. https://doi.org/10.1504/IJESDF.2007.013596

892 Al-Issa, Y., Ottom, M. A., & Tamrawi, A. (2019). EHealth Cloud Security Challenges: A
893   Survey. *Journal of Healthcare Engineering*, *2019*. https://doi.org/10.1155/2019/7516035

894 Ali, S., Shah, R., & Issac, B. (2018). Performance comparison of intrusion detection systems and
895   application of machine learning to Snort system. *Future Generation Computer Systems*, *80*,
896   157–170. https://doi.org/10.1016/j.future.2017.10.016

897 Ali, S., Shah, R., Khuzdar, T., Issac, B., & Jacob, S. M. (2018). Intelligent Intrusion Detection
898   System Through Combined and Optimized Machine Learning. *International Journal of*
899   *Computational Intelligence and Applications*, *June*.
900   https://doi.org/10.1142/S1469026818500074

901 Anderson, J. P. (1980). Computer security threat monitoring and surveillance. *Technical Report*
902   *James P Anderson Co Fort Washington Pa*, 56. https://doi.org/citeulike-article-id:592588

903 Arrington, B., Barnett, L. E., Rufus, R., & Esterline, A. (2016). Behavioral modeling intrusion
904   detection system (BMIDS) using internet of things (IoT) behavior-based anomaly detection
905   via immunity-inspired algorithms. *2016 25th International Conference on Computer*
906   *Communications and Networks, ICCCN 2016*.
907   https://doi.org/10.1109/ICCCN.2016.7568495

908 Aung, Y. Y., & Min, M. M. (2018). An analysis of K-means algorithm based network intrusion
909   detection system. *Advances in Science, Technology and Engineering Systems*, *3*(1), 496–
910   501. https://doi.org/10.25046/aj030160

911 Bhosale, K. S., Nenova, M., & Iliev, G. (2020). Intrusion Detection in Communication Networks
912   Using Different Classifiers. *Techno-Societal 2018*, 19–28. https://doi.org/10.1007/978-3-
913   030-16962-6_3

914 Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection:

915   Methods, systems and tools. *IEEE Communications Surveys and Tutorials*, *16*(1), 303–336.
916   https://doi.org/10.1109/SURV.2013.052213.00046

917 Bul'ajoul, W., James, A., & Pannu, M. (2015). Improving network intrusion detection system
918   performance through quality of service configuration and parallel technology. *Journal of*
919   *Computer and System Sciences*, *81*(6), 981–999. https://doi.org/10.1016/j.jcss.2014.12.012

920 Chanthakoummane, Y., Saiyod, S., Benjamas, N., & Khamphakdee, N. (2016). Improving
921   intrusion detection on snort rules for botnets detection. *Lecture Notes in Electrical*
922   *Engineering*, *376*, 765–779. https://doi.org/10.1007/978-981-10-0557-2_74

923 Damaševičius, R., Venčkauskas, A., Toldinas, J., & Grigaliūnas, Š. (2021). Ensemble-based
924   classification using neural networks and machine learning models for windows pe malware
925   detection. *Electronics (Switzerland)*, *10*(4), 1–26.
926   https://doi.org/10.3390/electronics10040485

927 Dang, Q. (2019). Studying machine learning techniques for intrusion detection systems To cite
928   this version : HAL Id : hal-02306521. *Studying Machine Learning Techniques for Intrusion*
929   *Detection Systems To Cite This Version : HAL Id : Hal-02306521.*

930 Dwivedi, N., & Tripathi, A. (2015). Event correlation for intrusion detection systems.
931   *Proceedings - 2015 IEEE International Conference on Computational Intelligence and*
932   *Communication Technology, CICT 2015*, *April*, 133–139.
933   https://doi.org/10.1109/CICT.2015.111

934 Einy, S., Oz, C., & Navaei, Y. D. (2021). The Anomaly- And Signature-Based IDS for Network
935   Security Using Hybrid Inference Systems. *Mathematical Problems in Engineering*, *2021*.
936   https://doi.org/10.1155/2021/6639714

937 Elrawy, M. F., Awad, A. I., & Hamed, H. F. A. (2018). Intrusion detection systems for IoT-
938   based smart environments: a survey. *Journal of Cloud Computing*, *7*(1).
939   https://doi.org/10.1186/s13677-018-0123-6

940 Erlacher, F., & Dressler, F. (2020). On High-Speed Flow-based Intrusion Detection using Snort-
941   compatible Signatures. *IEEE Transactions on Dependable and Secure Computing*, 1–1.
942   https://doi.org/10.1109/TDSC.2020.2973992

943 Essid, M., Jemili, F., & Korbaa, O. (2021). Distributed architecture of snort IDS in cloud
944   environment. *Advances in Intelligent Systems and Computing*, *1181 AISC*(December 2019),
945   100–111. https://doi.org/10.1007/978-3-030-49342-4_10

946 Ferebee, D., Dasgupta, D., Schmidt, M., & Wu, Q. (2011). Security visualization: Cyber security
947   storm map and event correlation. *IEEE SSCI 2011: Symposium Series on Computational*
948   *Intelligence - CICS 2011: 2011 IEEE Symposium on Computational Intelligence in Cyber*
949   *Security*, *September 2015*, 171–178. https://doi.org/10.1109/CICYBS.2011.5949412

950 Fitni, Q. R. S., & Ramli, K. (2020). Implementation of ensemble learning and feature selection
951   for performance improvements in anomaly-based intrusion detection systems. *Proceedings -*
952   *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and*
953   *Communications Technology, IAICT 2020*, 118–124.
954   https://doi.org/10.1109/IAICT50021.2020.9172014

955 Grammatikis, P. R., Sarigiannidis, P., Dalamagkas, C., Spyridis, Y., Lagkas, T., Efstathopoulos,
956   G., Sesis, A., Pavon, I. L., Burgos, R. T., Diaz, R., Sarigiannidis, A., Papamartzivanos, D.,
957   Menesidou, S. A., Ledakis, G., Pasias, A., Kotsiopoulos, T., Drosou, A., Mavropoulos, O.,
958   Subirachs, A. C., … Arce, A. (2021). SDN-Based Resilient Smart Grid: The SDN-
959   microSENSE Architecture. *Digital*, *1*(4), 173–187. https://doi.org/10.3390/digital1040013

960 Jabbar, M. A., & Aluvalu, R. (2018). A Signature-based Intrusion Detection System for the
961   Internet of Things. *IET Conference Publications*, *2018*(CP747).
962   https://doi.org/10.1049/cp.2018.1419

963 Jaw, E., & Wang, X. (2021). Feature Selection and Ensemble-Based Intrusion Detection System:
964   An Efficient and Comprehensive Approach. *Symmetry 2021, Vol. 13, Page 1764*, *13*(10),
965   1764. https://doi.org/10.3390/SYM13101764

966 Jeong, J. H., Lee, S. Y., Park, G. H., & Choi, S. G. (2020). Rule conversion mechanism between
967   NIDPS engines. *International Conference on Advanced Communication Technology,
968   ICACT*, *2020*, 581–584. https://doi.org/10.23919/ICACT48636.2020.9061387

969 Jyothsna, V., & Prasad, K. M. (2019). Anomaly-Based Intrusion Detection System. *Computer
970   and Network Security*. https://doi.org/10.5772/INTECHOPEN.82287

971 Kagara, B. N., & Md Siraj, M. (2020). A Review on Network Intrusion Detection System Using
972   Machine Learning. In *International Journal of Innovative Computing* (Vol. 10, Issue 1).
973   Springer International Publishing. https://doi.org/10.11113/ijic.v10n1.252

974 Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection
975   systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1).
976   https://doi.org/10.1186/s42400-019-0038-7

977 Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., & Alazab, A. (2020). Hybrid intrusion
978   detection system based on the stacking ensemble of C5 decision tree classifier and one class
979   support vector machine. *Electronics (Switzerland)*, *9*(1).
980   https://doi.org/10.3390/electronics9010173

981 Khurat, A., & Sawangphol, W. (2019). An Ontology for SNORT Rule. *JCSSE 2019 - 16th
982   International Joint Conference on Computer Science and Software Engineering:
983   Knowledge Evolution Towards Singularity of Man-Machine Intelligence*, 49–55.
984   https://doi.org/10.1109/JCSSE.2019.8864190

985 Kushwaha, P., Buckchash, H., & Raman, B. (2017). Anomaly based intrusion detection using
986   filter based feature selection on KDD-CUP 99. *IEEE Region 10 Annual International
987   Conference, Proceedings/TENCON*, *2017-Decem*, 839–844.
988   https://doi.org/10.1109/TENCON.2017.8227975

989 Li, Z., Sanghi, M., Chen, Y., Kao, M. Y., & Chavez, B. (2006). Hamsa: Fast signature generation
990   for zero-day polymorphic worms with provable attack resilience. *Proceedings - IEEE
991   Symposium on Security and Privacy*, *2006*, 32–46. https://doi.org/10.1109/SP.2006.18

992 Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., & Das, K. (2000). The 1999 DARPA off-
993   line intrusion detection evaluation. *Computer Networks*, *34*(4), 579–595.
994   https://doi.org/10.1016/S1389-1286(00)00139-0

995   Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection
996   systems: A survey. *Applied Sciences (Switzerland)*, *9*(20).
997   https://doi.org/10.3390/app9204396

998   Liu, M., Xue, Z., Xu, X., Zhong, C., & Chen, J. (2019). Host-Based Intrusion Detection System
999   with System Calls. *ACM Computing Surveys*, *51*(5), 1–36. https://doi.org/10.1145/3214304

1000 Lyu, R., He, M., Zhang, Y., Jin, L., & Wang, X. (2021). Network intrusion detection based on an
1001   efficient neural architecture search. *Symmetry*, *13*(8), 1–23.
1002   https://doi.org/10.3390/sym13081453

1003 Mahfouz, A., Abuhussein, A., Venugopal, D., & Shiva, S. (2020). Ensemble classifiers for
1004   network intrusion detection using a novel network attack dataset. *Future Internet*, *12*(11),
1005   1–19. https://doi.org/10.3390/fi12110180

1006 McIntosh, T., Jang-Jaccard, J., Watters, P., & Susnjak, T. (2019). The Inadequacy of Entropy-
1007   Based Ransomware Detection. *Communications in Computer and Information Science*,
1008   *1143 CCIS*, 181–189. https://doi.org/10.1007/978-3-030-36802-9_20

1009 Meryem, A., & Ouahidi, B. EL. (2020). Hybrid intrusion detection system using machine
1010   learning. In *Network Security* (Vol. 2020, Issue 5). Springer Singapore.
1011   https://doi.org/10.1016/S1353-4858(20)30056-8

1012 Miller, A. (2019). The intrinsically linked future for human and Artificial Intelligence
1013   interaction. *Journal of Big Data*, *6*(1), 1–9. https://doi.org/10.1186/S40537-019-0202-
1014   7/METRICS

1015 Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). *Kitsune: An Ensemble of
1016   Autoencoders for Online Network Intrusion Detection. February*, 18–21.
1017   https://doi.org/10.14722/ndss.2018.23204

1018 Mishra, V., Vijay, V. K., & Tazi, S. (2016). Intrusion detection system with snort in cloud
1019   computing: Advanced IDS. *Advances in Intelligent Systems and Computing*, *408*, 457–465.
1020   https://doi.org/10.1007/978-981-10-0129-1_48

1021 Niyaz, Q., Sun, W., Javaid, A. Y., & Alam, M. (2015). A deep learning approach for network
1022   intrusion detection system. *EAI International Conference on Bio-Inspired Information and
1023   Communications Technologies (BICT)*. https://doi.org/10.4108/eai.3-12-2015.2262516

1024 Nyasore, O. N., Zavarsky, P., Swar, B., Naiyeju, R., & Dabra, S. (2020). Deep Packet Inspection
1025   in Industrial Automation Control System to Mitigate Attacks Exploiting Modbus/TCP
1026   Vulnerabilities. *Proceedings - 2020 IEEE 6th Intl Conference on Big Data Security on
1027   Cloud, BigDataSecurity 2020, 2020 IEEE Intl Conference on High Performance and Smart
1028   Computing, HPSC 2020 and 2020 IEEE Intl Conference on Intelligent Data and Security,
1029   IDS 2020*, 241–245. https://doi.org/10.1109/BIGDATASECURITY-HPSC-
1030   IDS49724.2020.00051

1031 Ojugo, A. A., Eboka, A. O., Okonta, O. E., Yoro, R. E., & Aghware, F. O. (2012). Genetic
1032   Algorithm Rule-Based Intrusion Detection System (GAIDS). *Journal of Emerging Trends
1033   In Computing Information Systems*, *3*(8), 1182–1194. http://www.cisjournal.org

1034 Othman, S. M., Mutaher Ba-Alwi, F., Alsohybe, N. T., & Zahary, A. T. (2018). Survey on

1035   Intrusion Detection System Types. *International Journal of Cyber-Security and Digital*
1036   *Forensics*, *7*(4), 444–462. https://www.researchgate.net/publication/329363322

1037   Papastergiou, S., Mouratidis, H., & Kalogeraki, E.-M. (2020). Cyber Security Incident Handling,
1038   Warning and Response System for the European Critical Information Infrastructures
1039   (CyberSANE). *Communications in Computer and Information Science*, *1000*, 476–487.
1040   https://doi.org/10.1007/978-3-030-20257-6_41

1041   Park, J. H. (2019). Advances in future internet and the industrial internet of things. *Symmetry*,
1042   *11*(2). https://doi.org/10.3390/sym11020244

1043   Park, W., & Ahn, S. (2017). Performance Comparison and Detection Analysis in Snort and
1044   Suricata Environment. *Wireless Pers Commun (2017) 94:241–252*, *94*, 241–252.
1045   https://doi.org/10.1007/s11277-016-3209-9

1046   Pham, N. T., Foo, E., Suriadi, S., Jeffrey, H., & Lahza, H. F. M. (2018). Improving performance
1047   of intrusion detection system using ensemble methods and feature selection. *ACM*
1048   *International Conference Proceeding Series*. https://doi.org/10.1145/3167918.3167951

1049   Sagala, A. (2015). Automatic SNORT IDS rule generation based on honeypot log. *Proceedings -*
1050   *2015 7th International Conference on Information Technology and Electrical Engineering:*
1051   *Envisioning the Trend of Computer, Information and Engineering, ICITEE 2015*, 576–580.
1052   https://doi.org/10.1109/ICITEED.2015.7409013

1053   Saleh, A. I., Talaat, F. M., & Labib, L. M. (2019). A hybrid intrusion detection system (HIDS)
1054   based on prioritized k-nearest neighbors and optimized SVM classifiers. *Artificial*
1055   *Intelligence Review*, *51*(3), 403–443. https://doi.org/10.1007/s10462-017-9567-1

1056   Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., & Ng, A. (2020).
1057   Cybersecurity data science: an overview from machine learning perspective. *Journal of Big*
1058   *Data*, *7*(1). https://doi.org/10.1186/s40537-020-00318-5

1059   Saxena, A. K., Sinha, S., & Shukla, P. (2017). General study of intrusion detection system and
1060   survey of agent based intrusion detection system. *Proceeding - IEEE International*
1061   *Conference on Computing, Communication and Automation, ICCCA 2017*, *2017-*
1062   *Janua*(May), 417–421. https://doi.org/10.1109/CCAA.2017.8229866

1063   Sekharan, S. S., & Kandasamy, K. (2018). Profiling SIEM tools and correlation engines for
1064   security analytics. *Proceedings of the 2017 International Conference on Wireless*
1065   *Communications, Signal Processing and Networking, WiSPNET 2017*, *2018-Janua*, 717–
1066   721. https://doi.org/10.1109/WiSPNET.2017.8299855

1067   Shah, S. A. R., & Issac, B. (2018). Performance comparison of intrusion detection systems and
1068   application of machine learning to Snort system. *Future Generation Computer Systems*, *80*,
1069   157–170. https://doi.org/10.1016/j.future.2017.10.016

1070   Sommestad, T., Holm, H., & Steinvall, D. (2021). Variables influencing the effectiveness of
1071   signature-based network intrusion detection systems. *Information Security Journal: A*
1072   *Global Perspective*. https://doi.org/10.1080/19393555.2021.1975853

1073   Suarez-Tangil, G., Palomar, E., De Fuentes, J. M., Blasco, J., & Ribagorda, A. (2009).
1074   Automatic rule generation based on genetic programming for event correlation. *Advances in*

1075   *Intelligent and Soft Computing*, *63 AISC*, 127–134. https://doi.org/10.1007/978-3-642-
1076   04091-7_16

1077   Suarez-Tangil, Guillermo, Palomar, E., Ribagorda, A., & Sanz, I. (2015). Providing SIEM
1078   systems with self-adaptation. *Information Fusion*, *21*(1), 145–158.
1079   https://doi.org/10.1016/j.inffus.2013.04.009

1080   Tama, B. A., Comuzzi, M., & Rhee, K. H. (2019). TSE-IDS: A Two-Stage Classifier Ensemble
1081   for Intelligent Anomaly-Based Intrusion Detection System. *IEEE Access*, *7*, 94497–94507.
1082   https://doi.org/10.1109/ACCESS.2019.2928048

1083   Tasneem, A., Kumar, A., & Sharma, S. (2018). Intrusion Detection Prevention System using
1084   SNORT. *International Journal of Computer Applications*, *181*(32), 21–24.
1085   https://doi.org/10.5120/ijca2018918280

1086   Topol, E. J. (2019). High-performance medicine: the convergence of human and artificial
1087   intelligence. *Nature Medicine*, *25*(1), 44–56. https://doi.org/10.1038/s41591-018-0300-7

1088   Vaiyapuri, T., & Binbusayyis, A. (2020). Application of deep autoencoder as an one-class
1089   classifier for unsupervised network intrusion detection: a comparative evaluation. *PeerJ*
1090   *Computer Science*, *6*, 1–26. https://doi.org/10.7717/peerj-cs.327

1091   Valeur, F., Vigna, G., & Kruegel, C. (2017). A Comprehensive Approach to Intrusion Detection
1092   Alert Correlation. *IEEE Transactions on Dependable and Secure Computing*, *1*(July), 146–
1093   169. https://doi.org/10.1109/TDSC.2004.21

1094   Verma, A., & Ranga, V. (2020). Machine Learning Based Intrusion Detection Systems for IoT
1095   Applications. *Wireless Personal Communications*, *111*(4), 2287–2310.
1096   https://doi.org/10.1007/s11277-019-06986-8

1097   Vokorokos, L., & BaláŽ, A. (2010). Host-based intrusion detection system. *INES 2010 - 14th*
1098   *International Conference on Intelligent Engineering Systems, Proceedings*, 43–47.
1099   https://doi.org/10.1109/INES.2010.5483815

1100   Xue, C. T. S., & Xin, F. T. W. (2016). Benefits and Challenges of the Adoption of Cloud
1101   Computing in Business. *International Journal on Cloud Computing: Services and*
1102   *Architecture*, *6*(6), 01–15. https://doi.org/10.5121/ijccsa.2016.6601

1103   Zhang, K., Zhao, F., Luo, S., Xin, Y., & Zhu, H. (2019). An Intrusion Action-Based IDS Alert
1104   Correlation Analysis and Prediction Framework. *IEEE Access*, *7*, 150540–150551.
1105   https://doi.org/10.1109/ACCESS.2019.2946261

1106   Zhou, Y., Cheng, G., Jiang, S., & Dai, M. (2020). Building an efficient intrusion detection
1107   system based on feature selection and ensemble classifier. *Computer Networks*, *174*.
1108   https://doi.org/10.1016/j.comnet.2020.107247

1109

Figure 1

A classical architecture of host and network-based intrusion detection system.

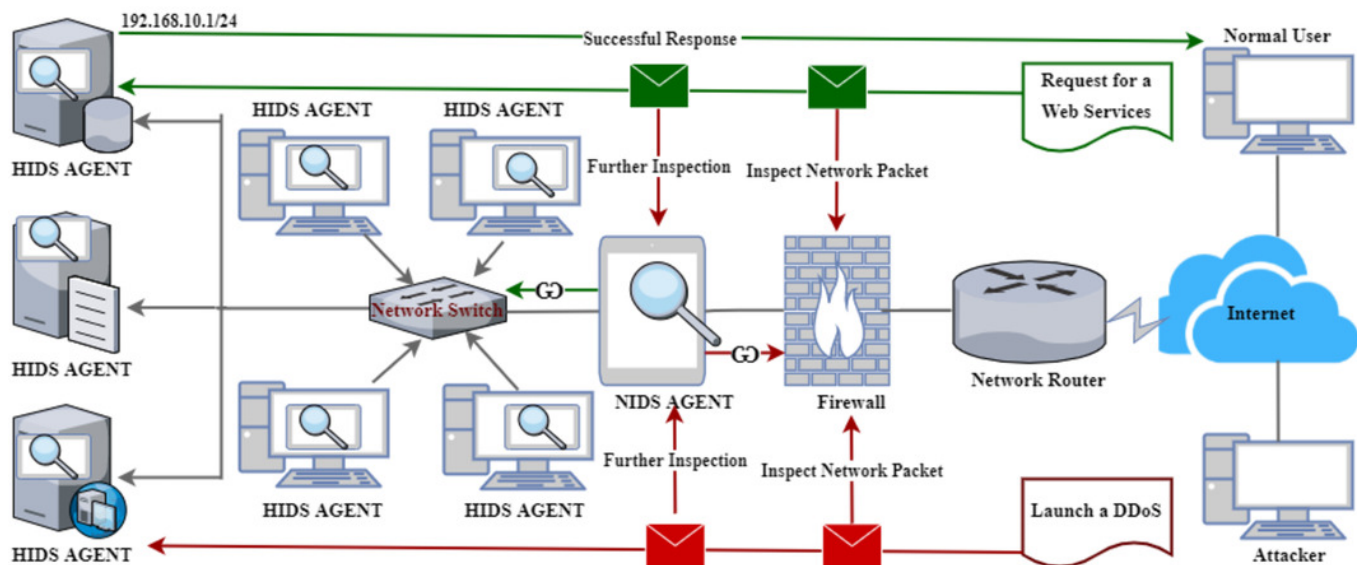



Figure 2

The schematic diagram of the snort automatic rule generator (SARG).

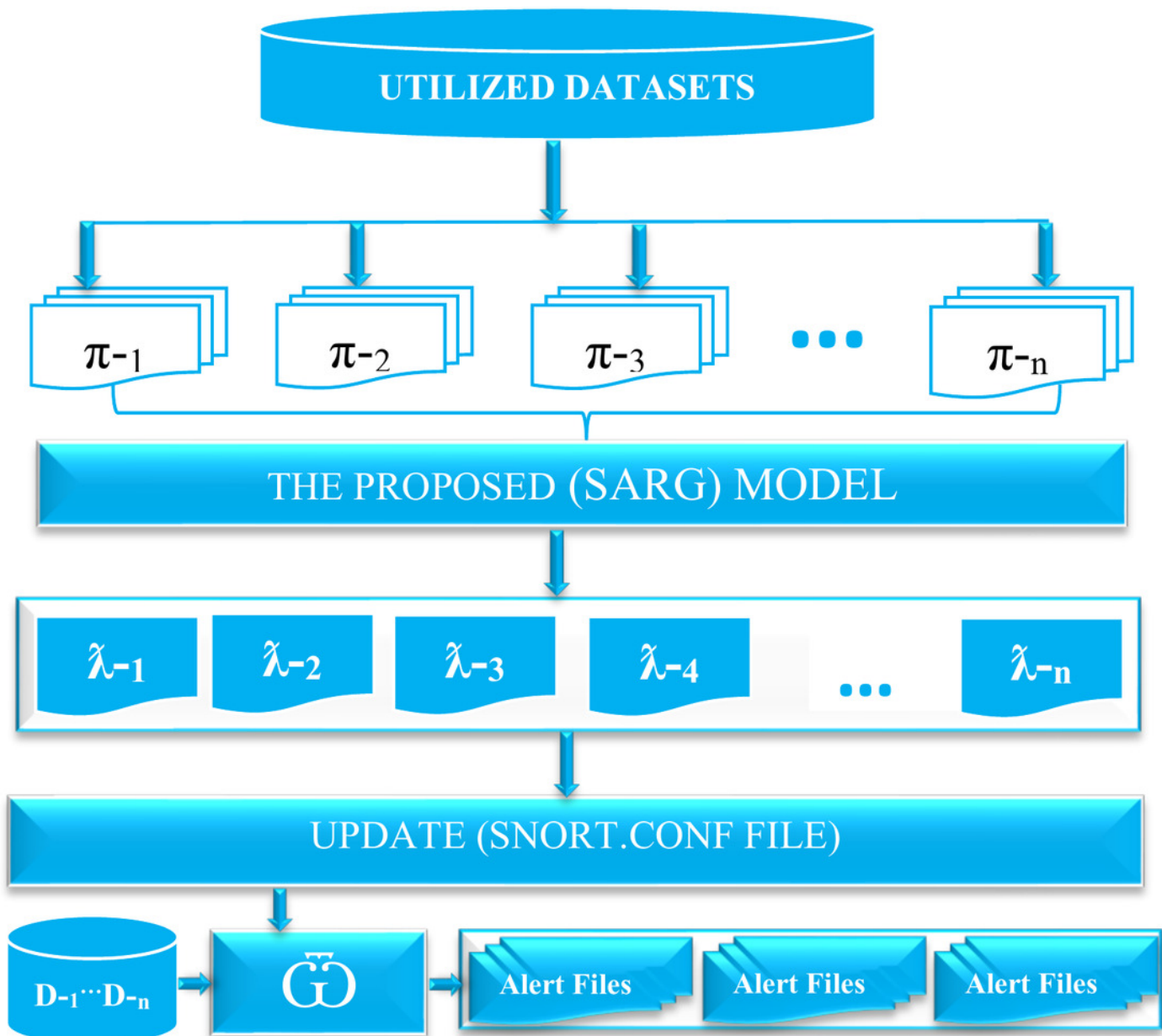



Figure 3

An overview of the proposed security event correlator (SEC).

Figure 4

Illustration of the alert aggregation process.

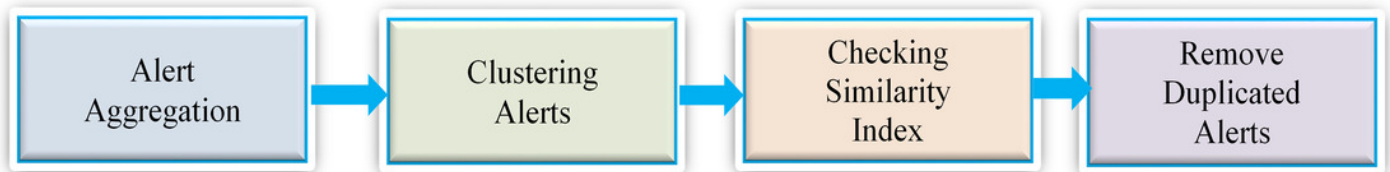



Figure 5

The correlation engine with the selected features for alert correlation.

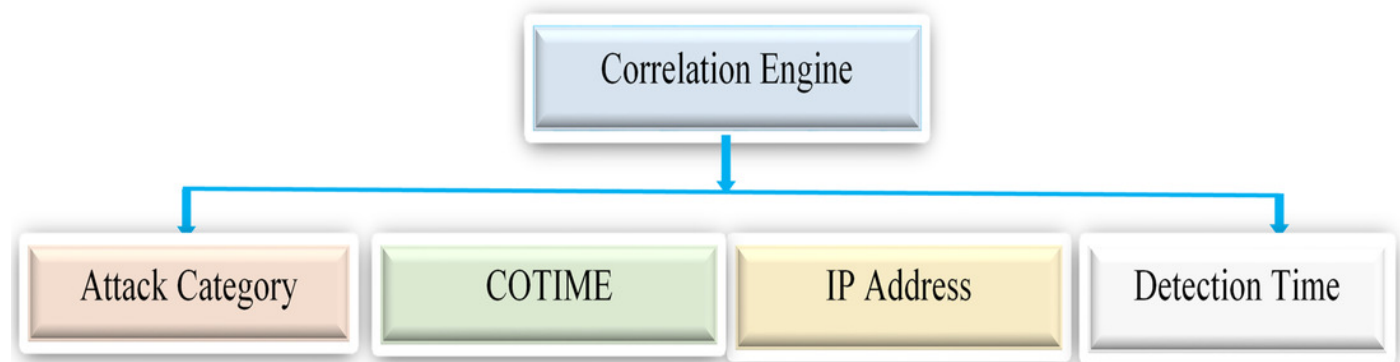



Figure 6

Representation of alert prioritization categories with respective common features.

Figure 7

The detailed overview of the proposed security event correlator (SEC).

Figure 8

Auto generated snort rules using pcap files as live attacks.

Figure 9

A demonstration of auto-generated snort rules using various pcap files as live attacks.

Figure 10

Evaluation of single instance alert aggregation and correlation.

Figure 11

Assessment of multiple instances alert aggregation and correlation.

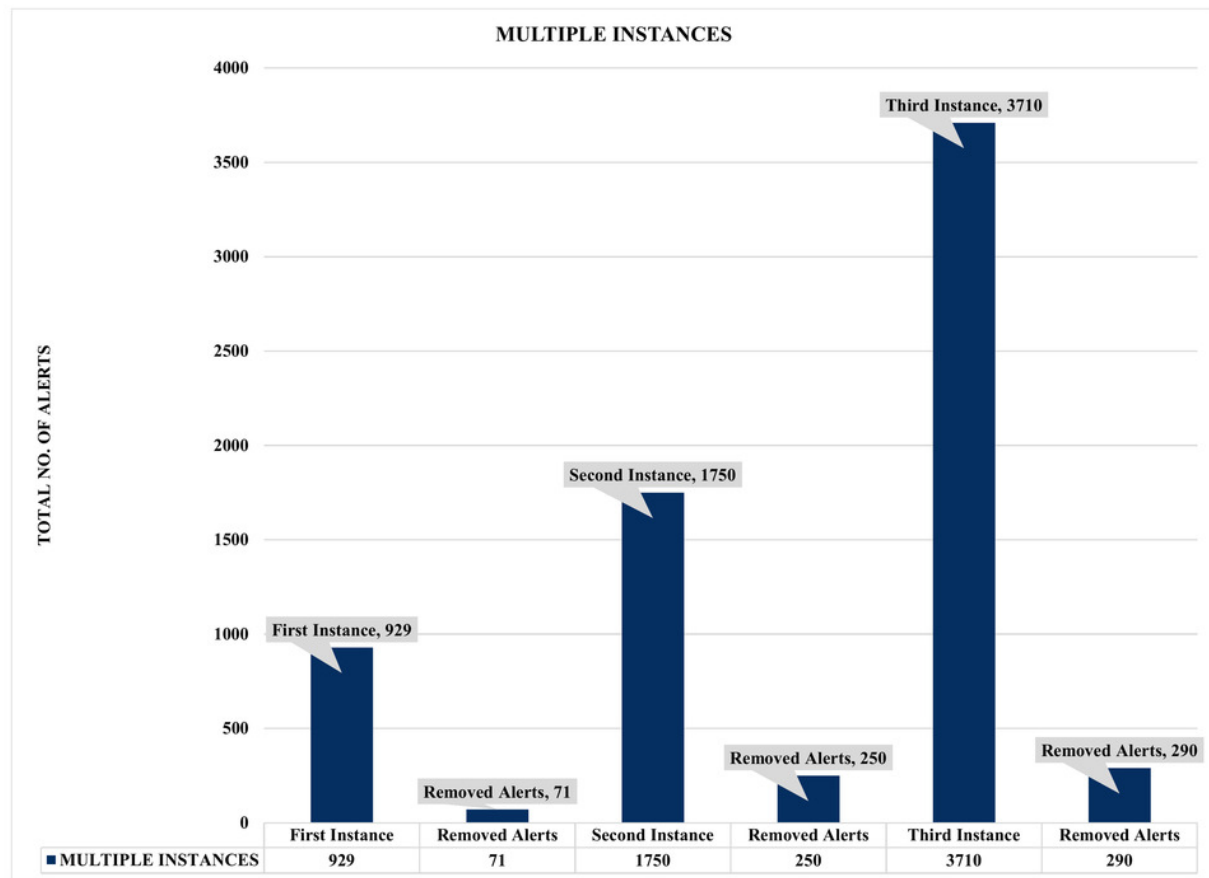



Figure 12

Illustration of the first two experiments of alert prioritization with multiple COTIME.

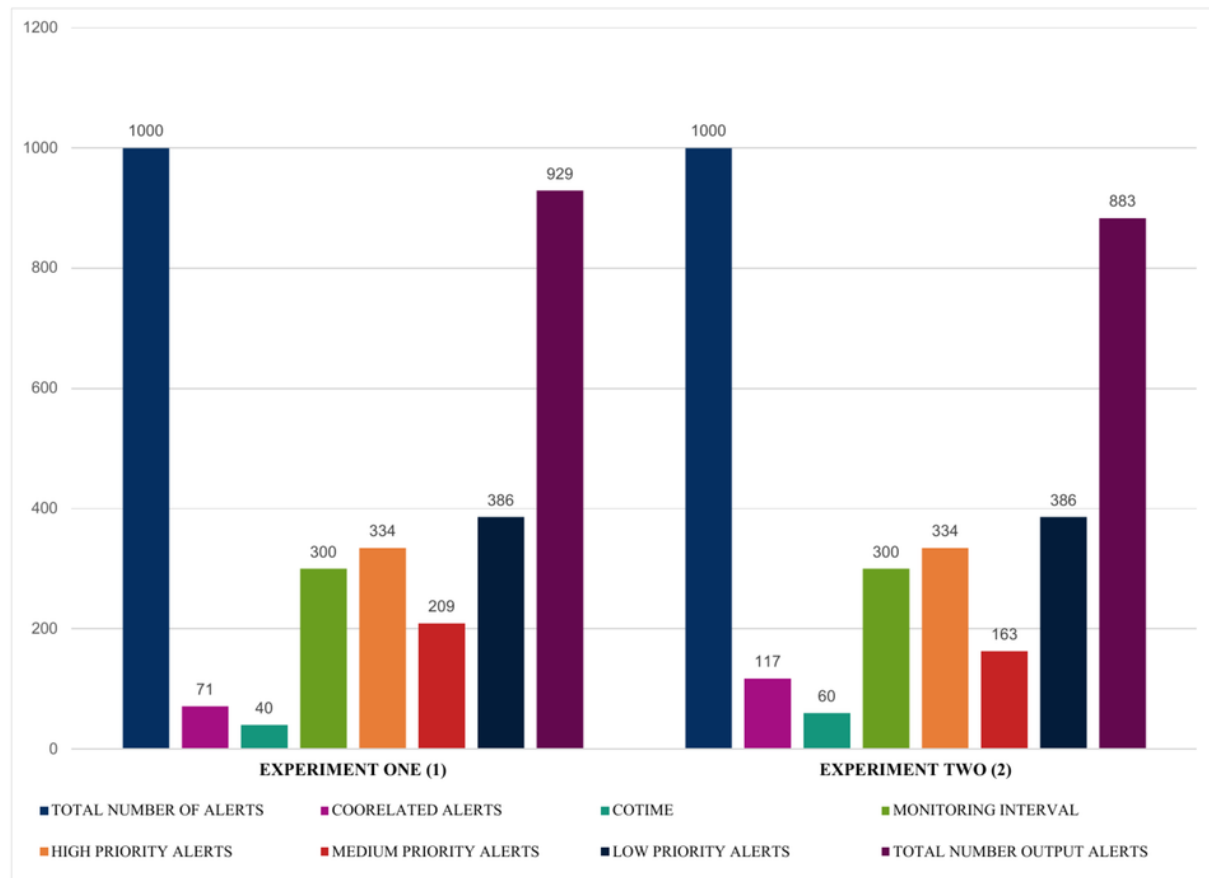



Figure 13

Demonstration of single instances with multiple COTIME for alert prioritization.

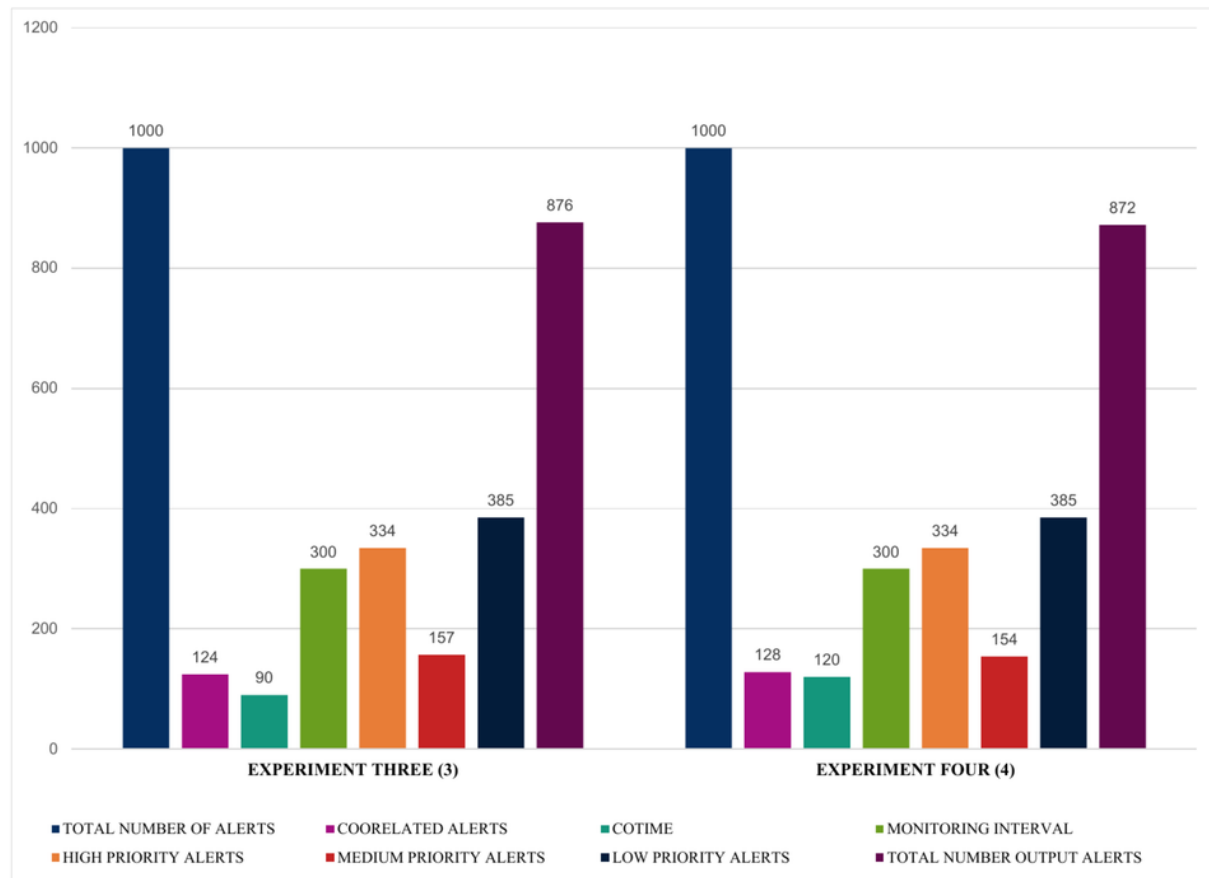



Figure 14

Further illustration of alert prioritization of single instances with multiple COTIME.

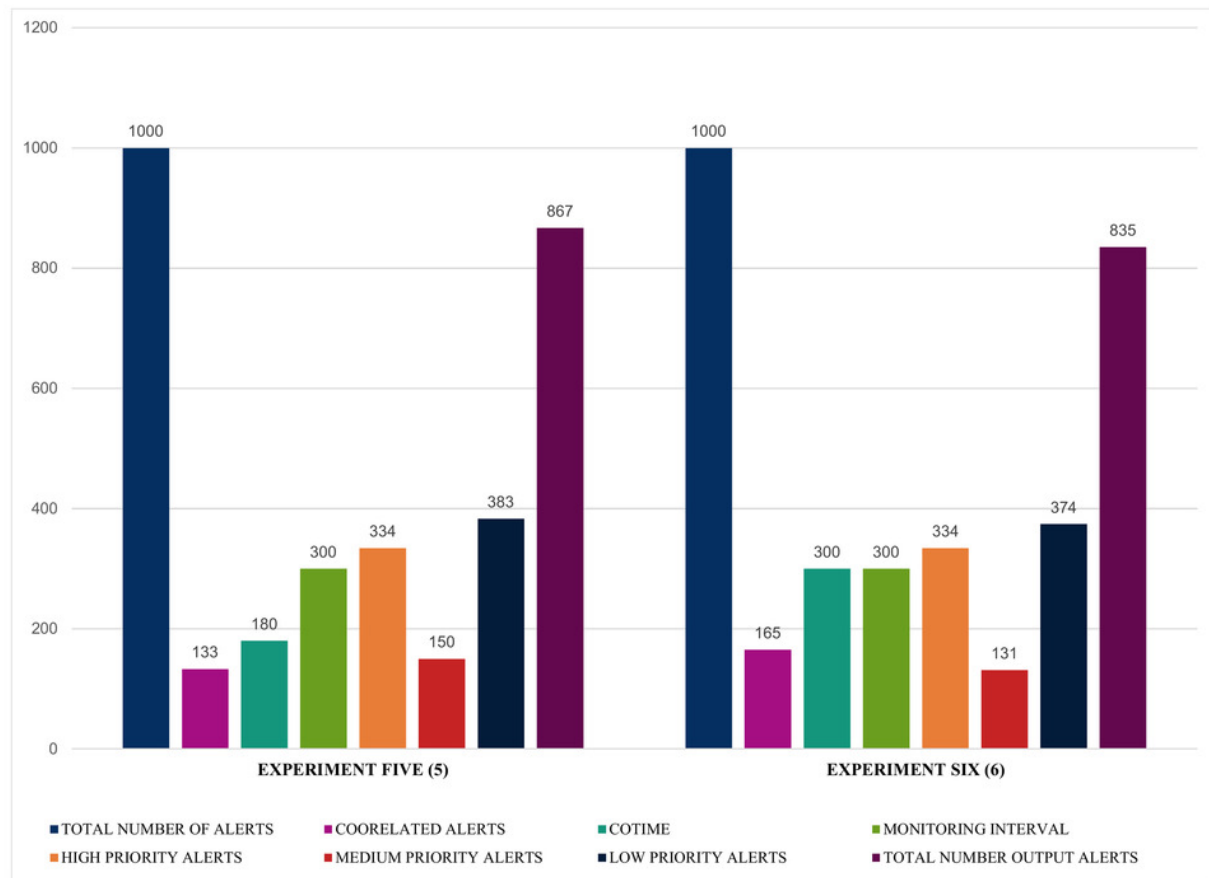



Figure 15

Alert prioritization for single instances with multiple correlation time (COTIME).

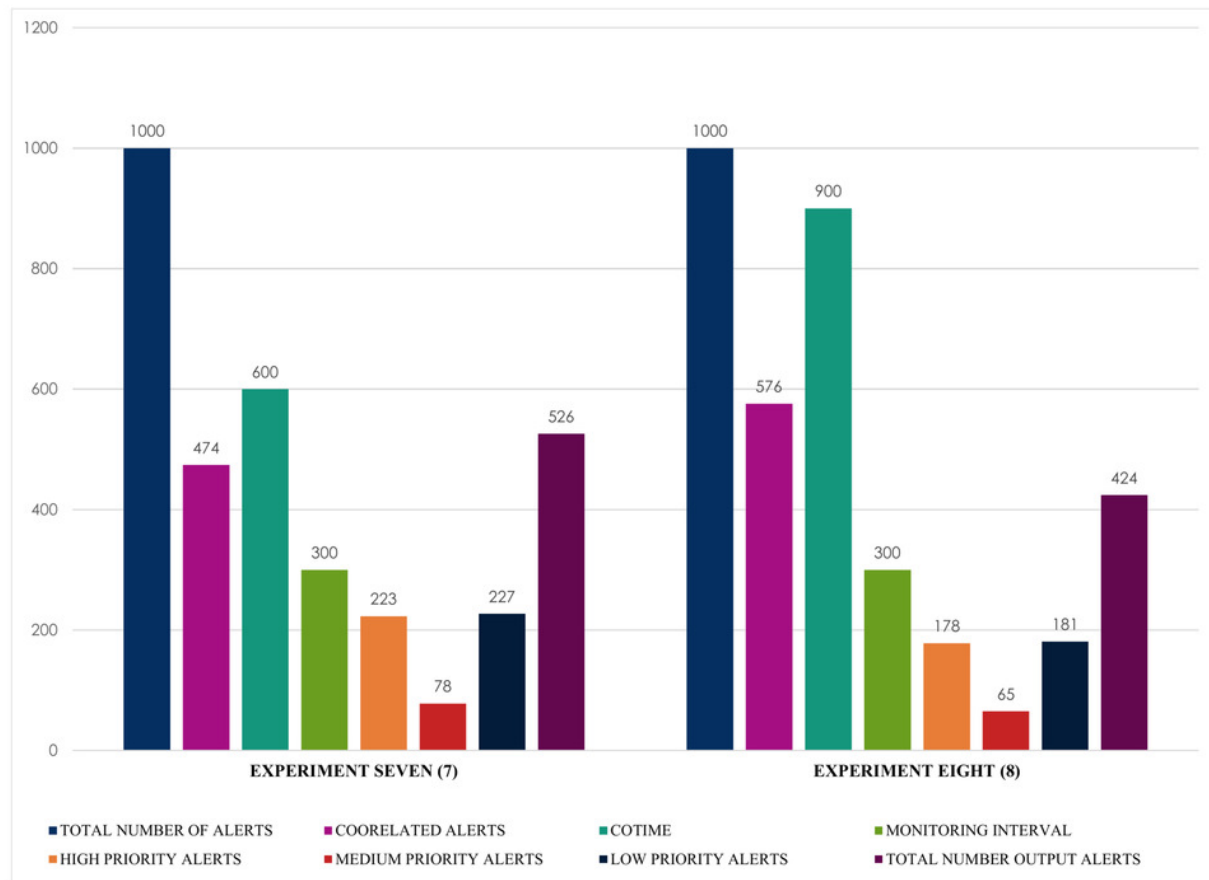



Figure 16

Time complexity graph with multiple instances.

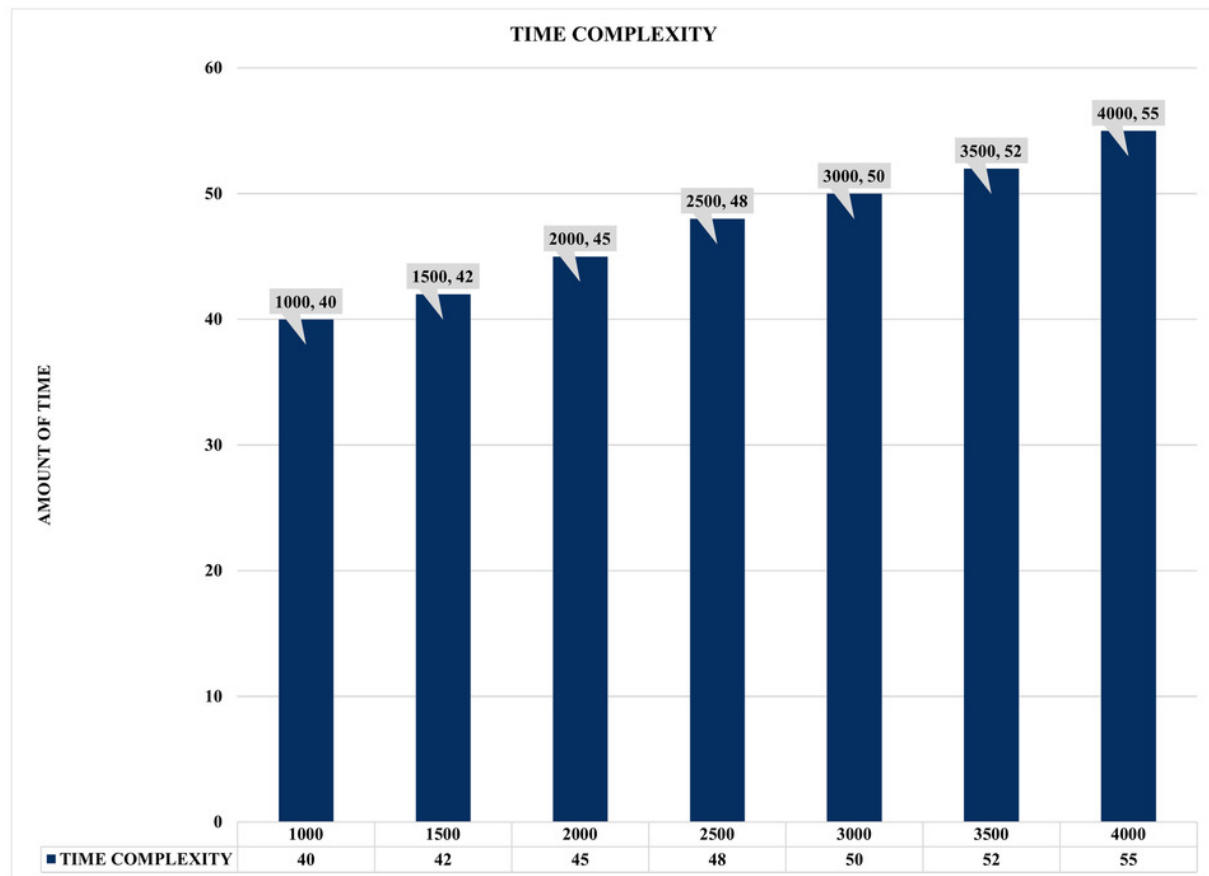



Figure 17

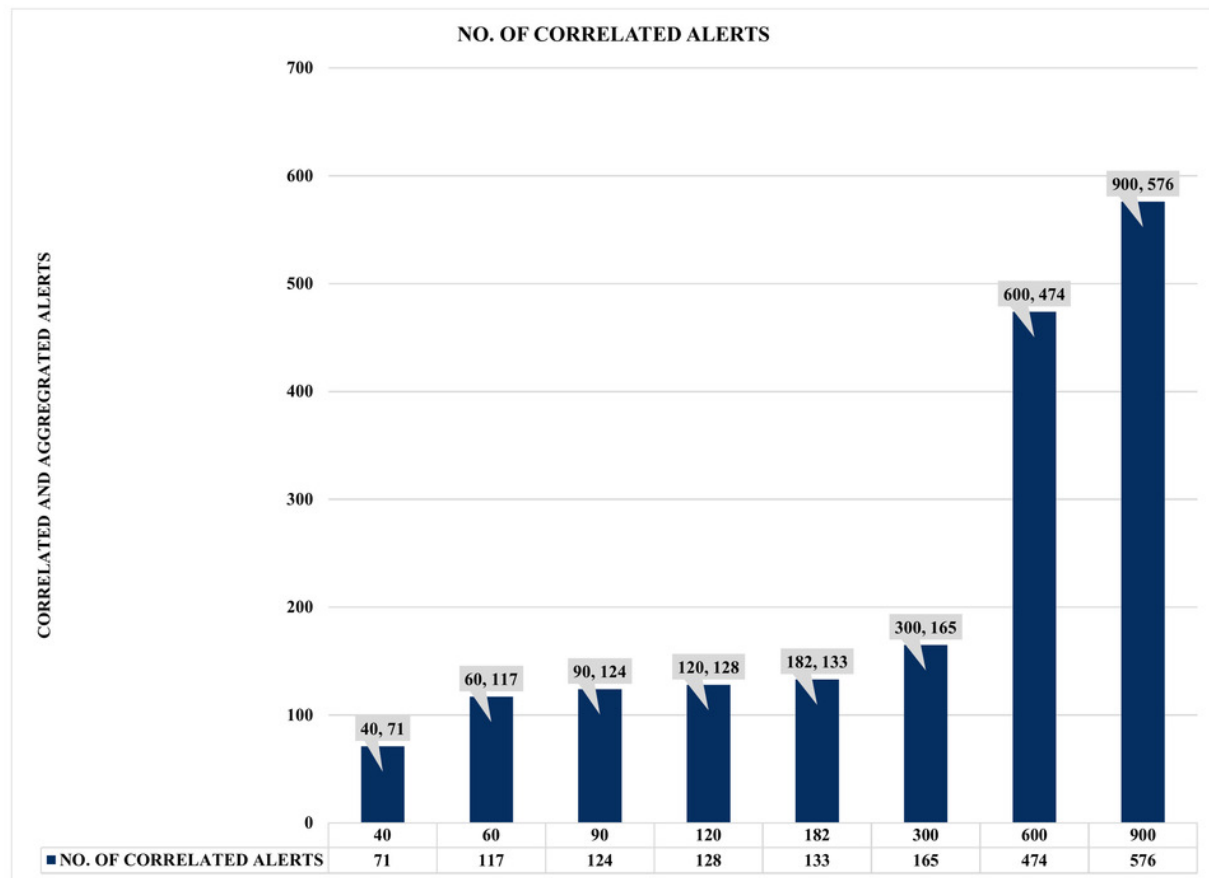
Comparison of correlation time (COTIME) with correlated alerts.

A classical representation of Snort rules syntax.

Table 1 A classical representation of Snort rules syntax									
Action	**Proto col**	**Src. IP**	**Src. Port**	**Direc tion**	**Dst. IP**	**Dst. Port**	**Parameters**		
alert	tcp	icmp	any	- >	any	any	(msg: "probe attack";	content: "probe";	sid: 1000023)
log	tcp	any	any	- >	any	80	(msg: "HTTP attack";	content: !"GET";	sid: 1000024)

A typical representation of the rule header components.

Table 2 A typical representation of the rule header components						
Action	**Protocol**	**Source. IP**	**Source Port**	**Direction**	**Destination IP**	**Destination Port**
alert	tcp	any	:1024	- >	192.168.100.1	600:
log	icmp	!172.16.10.0/16	any	- >	172.16.10.252	1:6000
alert	tcp	!192.168.10.0/24	any	< >	192.168.10.0/24	1:1024

Standard illustration of the rule header components.

Table 3 A typical illustration of the rule header components							
General rule options					**Payload options**		
msg	**sid**	**rev**	**gid**	**priority**	**content**	**offset**	**depth**
(msg:"Anomaly detected";	sid: 1000027;	rev: 3;	gid: 1000032;	priority: 1;	content: "probe-attack";	offset:4;	depth:12;

Content extractor (CE) based algorithm.

Algorithm 1.0: Content extractor (CE) based algorithm
Input: *SequenceSet = {S₁, S₂... Sₛ}, minimumsupport*
Output: ContentSet = {C₁, C₂, …. Cᴄ}

CE (SeqSet, MinimumSupport)
01 **begin**
02   ***for each** Seq **S** in the Seq Set **do***;
03   ***for each** Character **a** in the **S** do*
04               $L_1 = L_1\ U_a$;
05   ***END** // blocks of statements*
06   ***END** // blocks of statements*
06   *K_o=2.0;*
07   *while L_{k-1} = O **do***
08   ***for each** content L_{k-1} **do***;
09   ***for** I=1 to Sequence **do***;
10   ***if** (S_i include c)*;
11   *Count = Count +1*;
12   ***END**// blocks of statements*
13   ***END**// blocks of statements*
14   ***if** ((count /s) < minsupp)*
15   *$L_{k-1} = L_{k-1} -C$;*
16   ***END**// blocks of statements*
17   ***END**// blocks of statements*
18   *L_k=candi_gene (L_{k-1})*
19   *K_0++;*
20   ***END**// blocks of statements*
21   ***ContentSet** = A_{lk}*
22   **Return** *Content Set*
23 **end**;

***Representation**: L_k : Length k ,CE: Content extractor*

The sub content extract (SCE) based algorithm.

Algorithm 2.0: The sub content extract (SCE) based algorithm

Input: L_{k-1}

Output: L_k

Candi_gen(Lk-1),

01 **begin**

02   *for each* content p0 in L_{k-1} do;

03   *foreach* content q_0 in L_{k-1} do

04   if $((p.a_2 = q.a_1) \;\&\&\; (p.a_3 = q.a_2) \;\&\&\; (p.a_{k-1} = q.a_{k-2}))$ then;

05             $l_k - l_k \; U < p.a_1, p.a_2, \quad , p.a_{k-1}, q.a_{k-1} >;$

06   *END // blocks of statements*

06   *END // blocks of statements*

07   *END // blocks of statements*

08   *Return L_k*

09 **end**;

Representation: L_k : Length K , CE: Content Extractor

Locate data extract (LDE) based algorithm.

Algorithm 3.0: Locate data extract (LDE) based algorithm
Input: *Content, Packet set = { p₁, p₂,   pₙ}*
Output: *Address location, Depth*

ACL (cont , packet set),
01 **begin**
02   ***AL** = Maximum_ Packet_ Size;*
03   ***Dept** =0;*
04   ***for each** packet t in packet Set do;*
05   ***if(t.isMatchContent(cont)) then***
06   ***al = minimum** (offset, t.getStartMatch(cont));*
07   ***depth=maximum** (offset, t.getStartMatch(cont));*
06   ***END** // blocks of statements*
07   ***END** // blocks of statements*
08   ***Return** address location, depth;*
09 **end**;

*****Representation:** ACL = analysis content location Al= depth maximum byte point each match*