applied sciences

MDPI

Article

Enhancing Cloud Computing Analysis: A CCE-Based HTTP-GET Log Dataset

Ziyad R. Alashhab [1,*], Mohammed Anbar [1,*], Shaza Dawood Ahmed Rihan [2], Basim Ahmad Alabsi [2] and Karamath Ateeq [3]

[1] National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia (USM), Gelugor 11800, Malaysia
[2] Applied College, Najran University, King Abdulaziz Street, Najran P.O. Box 1988, Saudi Arabia
[3] School of Computing, Skyline University College, University City of Sharjah, Sharjah P.O. Box 1797, United Arab Emirates; karamath.ateeq@skylineuniversity.ac.ae
* Correspondence: ziyadashhab@yahoo.com (Z.R.A.); anbar@usm.my (M.A.)

Abstract: The Hypertext Transfer Protocol (HTTP) is a common target of distributed denial-of-service (DDoS) attacks in today's cloud computing environment (CCE). However, most existing datasets for Intrusion Detection System (IDS) evaluations are not suitable for CCEs. They are either self-generated or are not representative of CCEs, leading to high false alarm rates when used in real CCEs. Moreover, many datasets are inaccessible due to privacy and copyright issues. Therefore, we propose a publicly available benchmark dataset of HTTP-GET flood DDoS attacks on CCEs based on an actual private CCE. The proposed dataset has two advantages: (1) it uses CCE-based features, and (2) it meets the criteria for trustworthy and valid datasets. These advantages enable reliable IDS evaluations, tuning, and comparisons. Furthermore, the dataset includes both internal and external HTTP-GET flood DDoS attacks on CCEs. This dataset can facilitate research in the field and enhance CCE security against DDoS attacks.

Keywords: cybersecurity; intrusion detection; dataset generation; cloud computing environment (CCE); distributed denial-of-service (DDoS) attacks; HTTP-GET; flood DDoS attacks; application-layer attacks

Citation: Alashhab, Z.R.; Anbar, M.; Rihan, S.D.A.; Alabsi, B.A.; Ateeq, K. Enhancing Cloud Computing Analysis: A CCE-Based HTTP-GET Log Dataset. *Appl. Sci.* **2023**, *13*, 9086. https://doi.org/10.3390/app13169086

Academic Editors: Konstantinos Rantos, Konstantinos Demertzis and George Drosatos

Received: 6 July 2023
Revised: 28 July 2023
Accepted: 1 August 2023
Published: 9 August 2023

1. Introduction

Cloud computing environments (CCEs) have transformed how information is consumed and shared in recent years, especially through online services [1]. For example, these services rely on the Hypertext Transfer Protocol (HTTP), an application-layer protocol, to retrieve resources like HTML documents. The HTTP often uses TLS-encrypted TCP connections and serves as the foundation for all data exchanges on the Internet.

The HTTP follows a client–server model, meaning requests are initiated by the end user, usually via a web browser. When a user requests a resource, the browser sends an HTTP request to the server, specifying the resource's location. The server sends the requested resource back to the client in an HTTP response message. Figure 1 visualizes various HTTP data exchanges via the Internet [2].

HTTP/1.0 [3] defines the GET, HEAD, and POST request methods. The more recent HTTP/1.1 [3] adds OPTIONS, PUT, DELETE, TRACE, and CONNECT request methods to the existing ones. GET and POST are the two most commonly used request messages in HTTP/1.1, with GET being more prevalent in deploying services.

As CCEs continue to gain popularity to deploy client services, they become increasingly vulnerable to distributed denial-of-service (DDoS) attacks [4]. In 2015, the number of DDoS attack incidents rose by 25% due to the increased use of CCE services and global Internet traffic. Furthermore, these attacks are expected to increase to 17 million by the end of 2020 [5]. Within CCEs, DDoS attacks can take many forms, such as HX-DDoS, XML-based DDoS, and HTTP flood attacks [5]. Figure 2 illustrates the most common protocols targeted

by application-layer attacks [6]. HTTP flood DDoS attacks are particularly concerning since they focus on specific web services and exploit vulnerabilities to prevent web servers from serving websites to users. This attack depletes the CCE's resources and services, making it a popular attack vector for malicious actors [7].

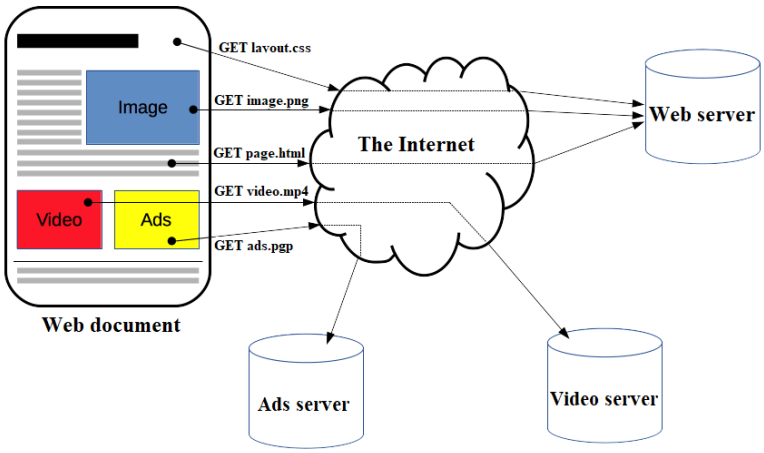

Figure 1. Data exchange through HTTP on the web.

The accuracy of many approaches at detecting attacks targeting CCEs has been problematic. The problem lies in the existing datasets used in IDS experiments, as they still cause various obstacles and limitations when testing the current approaches. Some of these issues include being outdated, lacking proper labeling, not encompassing all attack types, insufficient attack scenarios, varying environments, data format incompatibility, and inadequate dataset representation.

To overcome these obstacles and meet the new challenges in DDoS detection, it is crucial to employ a new dataset. By doing so, we can achieve higher accuracy when detecting DDoS attacks in future research [8–11].

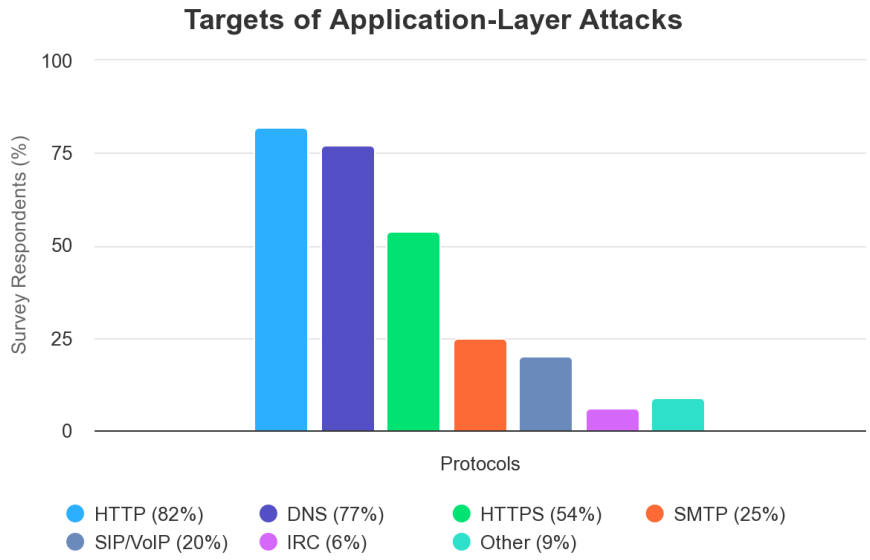

Figure 2. Most common protocols targeted by application-layer attacks.

1.1. HTTP Flood Attack

Users access CCE services through a web browser by using the GET or POST request method, which is handled by an HTTP server [1]. HTTP flood attacks are DDoS attacks that target the services, applications, or protocols within the OSI model's application layer (layer

7), such as their web services, web servers, and HTTP. Since the HTTP is the foundation of web-based requests and data exchange, these attacks can quickly deplete CCE resources and services. These attacks are particularly difficult to detect because the malicious requests are indistinguishable from normal ones.

HTTP flood attacks are typically carried out via botnets or malware-infected devices to send out multiple attack requests, resulting in a denial of service. There are two primary types of HTTP flood attacks: HTTP-GET and HTTP-POST.

- *HTTP-GET Attack:* An HTTP-GET attack involves multiple computing devices sending HTTP-GET requests for resources, such as documents or pictures, to a targeted web server (WS) until it is overwhelmed and stops responding to all requests, including legitimate ones.
- *HTTP-POST Attack:* An HTTP-POST attack targets web applications that use the HTTP-POST method to handle user-submitted data. The attacker floods the server with fake HTTP-POST requests, overwhelming its resources and causing it to crash or become unresponsive. This can disrupt legitimate traffic, making the website inaccessible and causing potential damage. These attacks are often automated and difficult to detect, making them a significant threat to online businesses and organizations.

The HTTP-GET flood DDoS attacks pose a significant security threat in the CCE. They have been identified as the most common form of application-layer attack [12]. It is essential to protect against these attacks to maintain the availability and reliability of web applications and services.

Figure 3 shows that HTTP-GET DDoS attacks are the most frequent application DDoS-layer attacks [12].

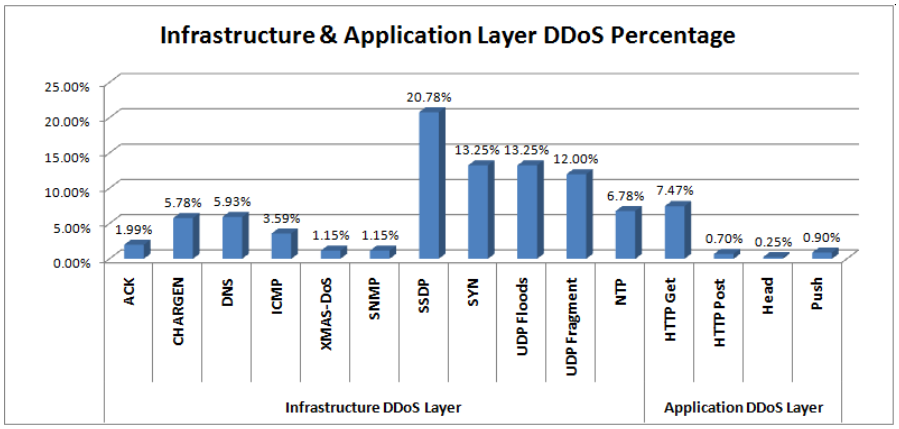

Figure 3. Majority of HTTP-GET DDoS attacks on application layer.

1.2. HTTP-GET Flood DDoS Attack in Cloud Computing Environment

In CCEs, attackers can target hosted cloud services in a structured or unstructured way, depending on the attacker's experience level and the attack's severity. In addition, the attacker's origin determines the CCE attack type as internal or external.

- *An internal attack* involves insiders, such as users misusing cloud resources or CSP employees with authorized CCE access, who know the organization's policies, network access, and cloud system security, making it easy for them to carry out attacks. They can steal and modify important data, deactivate specific processes, and deny CCE services. Installing internal Intrusion Detection Systems (IDSs) can help to identify and prevent these attacks.
- *An external attack* is carried out by outsiders, not members of the CCEs. The CCE administrators team is responsible for detecting and preventing these attacks, typically by using IDSs or firewalls.

These attacks deplete the resources of WSs and the hosting environment (the CCE), causing the services to go down. In other words, HTTP-GET flood DDoS attacks impact the

accessibility of CCE services, especially on WSs, affecting CCE-hosted services. Moreover, this attack affects the CCE resources themselves [5] because of the CCE's characteristics and nature, such as its elasticity, virtualization, and multitenancy (being on demand); this is because the resources are depleted due to more resources being exploited during the execution of the attack [13]. Such circumstances emerge when many clients simultaneously enter WSs or criminals bring the services down by utilizing attacks such as DDoS attacks against WSs on CCEs [13].

Among the attacks on the application layer, the HTTP-GET method attack stands out as the most prevalent, as it makes up the highest percentage of attacks. When a user or attacker generates an HTTP-GET request, it passes through the WS, where it is recorded along with its status and request code (representing the status of the requester's transactions). Defending against such attacks is crucial, and IDSs play a significant role in detecting them. However, the IDSs in CCEs face limitations, primarily due to the issue of evaluating classifiers trained on nonrealistic datasets from different environments, which lack essential features specific to CCEs. Furthermore, one of the most significant challenges in evaluating anomaly detection systems is the availability of appropriate public CCE-specific datasets [14,15].

The limitations mentioned earlier have served as a motivation to propose a new CCE-specific dataset that considers the unique characteristics of CCEs. This dataset is intended to be used to evaluate, test, and train IDSs, with a specific focus on HTTP-GET flood attacks at the application layer.

By developing this new dataset, we aim to address the deficiencies in the existing datasets, which often lack relevance and realism when applied to CCE scenarios. The dataset will be curated to reflect the specific challenges and intricacies of the CCE, making it more suitable to accurately evaluate the performance of IDSs at detecting and countering HTTP-GET flood attacks.

This paper is structured as follows: Section 2 presents the importance of a CCE's dataset and log files and compares the available dataset representations. Section 3 provides a literature review of the related datasets, along with some observations that motivated our work. Section 4 elaborates on the dataset design principles and significance. Additionally, Section 5 presents the requirements of an effective dataset, while Section 6 explains the proposed methodology to generate, formulate, and prepare the evaluation processes of the proposed dataset. The results and discussion are presented in Section 7. The final section, Section 8, concludes this paper and proposes several future research directions.

2. Background

A CCE is a model that boosts availability and offers characteristics unlike the traditional environment, including resource pooling, on-demand self-service, broad network access, measured services, elasticity, and security, to its users on demand, with minimal human intervention between the cloud service provider and the beneficiary of the service. Despite the capability and performance of CCE security, the CCE still succumbed to DDoS attacks in some situations, causing the target services used by the CCE servers to become unavailable.

The scarcity of datasets on HTTP-GET flood DDoS attacks on CCEs makes it difficult to compare and accurately evaluate new detection systems designed to identify new attacks. IDSs [16] are assessed and compared by using real labeled datasets with a comprehensive set of all conceivable attack scenarios before being deployed in a real CCE. As a result, the HTTP-GET flood DDoS attacks on CCE [17] security technology such as IDSs are limited to self-generated datasets, whose coverage (comprehensiveness), integrity, appropriateness, wholeness, and validity are not assured because of their inability to meet the strict requirements of benchmark datasets.

2.1. Cloud Computing Environment (CCE)

A CCE is a popular and promising concept that realizes the vision of computing as utility [1,13]. The CCE enables users to access services anytime, anywhere, and from any device through the Internet, thanks to its low cost and easy-to-use deployment services for applications on CCEs.

New applications with strict requirements have driven improved CCE models to deliver services. The CCE model consists of three main categories of services [18]:

- *Software as a service (SaaS)*, where clients can use CSP's applications hosted on a CCE infrastructure.
- *Platform as a service (PaaS)*, where developers can create and deploy applications by using the development platform provided by CSP.
- *Infrastructure as a service (IaaS)*, where users can rent storage units, networks, virtual machines (VMs), and other computing resources on demand [19].

Many businesses adopt the CCE model to enhance their operations due to its scalability, reliability, and cost effectiveness. The CCE services can be deployed in four different modes [19]: (i) private, (ii) public, (iii) community, and (iv) hybrid.

Virtualization is the key feature of the CCE that enables autoscaling and elasticity, where VMs can acquire more resources when needed and release unused resources when idle. Thus, a VM in the CCE can avoid a resource shortage as abundant resources are available on demand.

2.2. Virtual-Machine-Based Apache Web Server

Virtualization is the key feature of CCEs that enables the deployment and management of large-scale VM infrastructure. Any service, such as a WS, can run on single or multiple VMs. A WS is an HTTP server that provides web services to clients. One of the most popular open-sourced WS is Apache by the Apache Software Foundation, which provides secure and reliable web services that comply with current HTTP standards. Apache can also customize its functionality with a large public library of modules. Apache can support access to various databases, programming languages, scripts, and authentication mechanisms. To the user, a WS may look like a single unit but it may be a cluster of VMs that share and distribute the load of requests on resources, such as RAM and CPU.

2.3. Logging and Log Files

Logging is a common practice in computing as most software produces some logs. A log is a file that records the events that occur while an OS or other program is running (such as servers, applications, systems, etc.). Logs can also reveal application-layer attacks by analyzing the input patterns in various log sources, such as the WS's access log, CCE logs, and VM resources log. However, finding, understanding, and collecting (gathering relevant data on) these logs is a major challenge.

2.3.1. Logs of Web Server Access

Each WS has an access log that stores the elaborate information of each request submitted by users through web browsers or other applications to the WS in chronological order. Thus, each HTTP/s request to the website will be logged inside the access log. The access log file, the server log file, is created on the server side to collect the HTTP information on the user's requests. When any user requests a website from a WS, all parties responsible for communication from beginning to end, such as the PC, server, and network, will submit a few data to the WS; thus, the access log will save them.

The information is usually collected, such as the IP address of the user/s, the date of the request/s and time, the requested page/s, the status code of the request/s, and the size served measured by bytes. The other information collected includes the browsers and OSs used by the users to access the website and the site that referred visitors to your site or that was accessed directly by the user [20].

Two fundamental access log formats are usually used [21]. First, the common log format (CLF) encompasses the IP_address, client_identity, user_ID, time_stamp, request, status_code, and response_size. Second, a combined log format is the same as the first but has two additional parameters: referer and user_agent.

2.3.2. Logs of Cloud Computing Environment Logs

The CCE generates various activity logs due to the activities of the VMs, the hosts themselves, or other components such as the storage units throughout their lifetime as long as they work correctly. The activity logs in the CCE include helpful information. In these logs, data generation depends on the VM activities that serve the service, such as a web service needing more resources, cloning, migration, and more events.

2.3.3. Logs of Virtual Machine Resources

The different configurations, components, and activities of the OS in each machine produce logs of various kinds and formats. RAM and CPU are the two key hardware resources of a VM that run the VM and record useful information in the logs. The CPU and RAM are virtual resources that simulate the functions of the physical CPU and physical RAM in the virtualization environment of the CCE. They can be allocated to different VMs as needed. The CPU and RAM in any VM appear as the physical CPU and physical RAM to the VM users.

2.4. Dataset Representations

The IDS can use three main types of input datasets. The first type, a network-traffic-based representation, including flow-based and packet-based forms. The second type is a log-based representation, which records events from various sources. The third type is a hybrid-based representation, which combines web-log and system-log-based data. In the network-traffic-based representation, the packet-based form comprises the entire packet captured from the network, which contains both the header and the payload.

2.4.1. Network-Traffic-Based Representation

Detection systems that use network-traffic-based datasets can identify various types of attacks, particularly those dependent on the payload content. Despite this, these systems suffer from heavyweight characteristics because of the large volume of traffic that needs to be inspected, and they tend to be slow. As a result, they are not well suited to represent the traffic generated by modern high-speed CCEs, as the resulting datasets can be quite large. For instance, researchers who conducted experiments on a realistic 1G-BaseX network generated 6GB of traffic per minute and 8TB of packets per day [22].

As a result, a detection system that uses network-traffic-based datasets needs to handle a large volume of traffic, such as processing and analysis, resulting in an increased system runtime and complexity. Additionally, owing to the potentially sensitive or private data in the packet, anonymization is mainly used to prevent the leakage of such data by removing IP addresses, incurring additional computing costs and perhaps data loss. More often than not, errors in data usually occur after the anonymization process.

A network-traffic-based representation captures all packets transmitted across the network, without filtering, by accessing incoming and outgoing packets at the network boundary. Packet-based datasets contain raw data from all OSI layers, including layers 2 to 7. Each packet contains a header and a payload, where the header has information such as the source and destination addresses, protocol type, and packet length. The payload contains the data transmitted or received, such as email messages, web pages, or files. Wireshark and TcpDump are software tools commonly used to capture packet-based data.

Unlike a packet-based representation, which simply captures the raw data of each packet, using flow-based representations is a powerful technique that relies on aggregating packets with similar features into flows. A flow-based representation does not contain the entire packets' data; it only contains a subset that is sufficient to characterize the

traffic in the datasets. However, the choice of packet features to define a flow-based representation varies depending on the type and scope of detectable attacks and the detection system's objective. The IP address flows are often defined by common features, such as the source and destination IP addresses, source and destination port numbers, and protocol. These parameters are selected based on the detection system's purpose and the targeted attacks [23].

The major limitation of flow-based representations is that they require the detecting system to build the flows before performing the task, which adds computational overhead compared to packet-based representations. Moreover, they are ineffective at detecting attacks that exploit the data not captured in the flow, such as packet payloads.

Researchers [24–28] evaluate attack-detection algorithms or approaches by using different datasets, such as CIDDS-001, CAIDA, DEFCON, LBNL, DARPA, KDD cup99, and NSL-KDD. However, these datasets have two major drawbacks. First, their suitability in the intrusion detection field has been heavily criticized. Second, none include the HTTP-Web log dataset [29]. Moreover, some have specific flaws that reduce their effectiveness as benchmarking datasets. For instance:

- Some datasets, like CAIDA (2011), were fully anonymized, with the payload completely erased, making them less useful to researchers.
- Some datasets, like KDD 99, are unavailable for other researchers (e.g., KDD 99), including new DDoS types such as the HTTP flood [30]. The KDD 99 dataset is also based on DARPA 98 data but suffers from the same flaws [14].
- NSL-KDD is an improved version of KDD 99 that addresses some of its shortcomings. However, it still has several issues, such as not representing current real networks well due to a lack of available datasets for network-based IDSs.

On the other hand, NSL-KDD has one advantage: the number of records in the training and testing datasets is reasonable. This makes conducting the trials on the entire collection more cost effective than selecting a small part at random. As a result, the evaluation results of different researchers will be comparable and consistent.

The ISOT dataset [31] is a recent addition to the field of intrusion detection. It was systematically developed to create profiles of normal traffic and malicious traces and aims to provide a more realistic dataset for researchers. However, the dataset is limited as it only includes unique traces of specific botnet attacks.

2.4.2. Log-Based Representation

Log-based representation refers to a sequential record of events that occur in an operating system, device, or software, including failures, successes, operations, and user requests. These logs are usually stored in a predetermined location and can be classified as a web log or system log.

Web-log-based files, such as web server access logs, are text files that record all activities happening within a web service or server. The information contained in these logs includes details such as (1) the protocol type used by the users to access the server, (2) visitor sign ins, (3) the pages that receive the most or least requests, (4) the sites that refer the visitors to the server, (5) the pages viewed by the visitors, (6) the request time, (7) the browsers used, and (8) the operating system type. Access logs are useful in security, especially to detect HTTP-GET flood DDoS attacks.

System-log-based files, such as VMware log files, document a virtual machine's (VM) activities by logging its components and any startup messages. They also record expected and unexpected restarts or shutdowns, system and hardware changes, the processing status, the errors or warnings generated, cloning, migration, and other critical processes.

There is a shortage of publicly available log-based datasets compared to network-traffic-based datasets [6]. This insufficiency leads to an ineffective evaluation of host-based IDSs, which desperately require more datasets [32]. As a result, researchers often rely on network-traffic-based datasets to evaluate host-based IDSs, which frequently result in misclassifications, as evidenced by the high rate of false alarms.

Generating log-based datasets through testbeds can be challenging due to several factors, such as the infrastructure's size, configuration, and diversity and the realism and complexity of the collected data environment. Therefore, conducting attacks in an environment that closely aligns with the use case and that can produce the desired data is essential. To achieve this, we prioritize several key considerations that form the foundation of our dataset generation methodology, as discussed in Section 4.

2.4.3. Hybrid Representation

The hybrid representation combines the characteristics of both network-traffic-based and log-based representations by merging two or more datasets to evaluate the IDSs. These datasets may use the same category of representation or a different category. However, this approach has some disadvantages, such as potential conflicts in the detection systems due to the heterogeneity of the data types.

Each representation has its advantages and disadvantages. Table 1 highlights their pros and cons to comprehensively overview the three dataset representations.

A detection system that uses network-traffic-based and hybrid-based representations can identify various types of attacks on CCEs, including HTTP-GET flood DDoS attacks. However, these systems can be inefficient and sluggish due to the large traffic volume to be inspected. On the other hand, log-based representations also comprise information to detect attacks, especially HTTP-GET flood DDoS attacks on CCEs. Therefore, log-based datasets can help detect HTTP-GET flood DDoS attacks and generate similar attack indicators but are faster than network-traffic-based ones. Therefore, a log-based representation is preferable for detecting this type of attack in the CCE security domain.

Considering the limitations of network-traffic-based and hybrid-based representations and the capabilities of log-based representation, we decided to use a log-based representation to create the proposed datasets for HTTP-GET flood DDoS attacks on CCEs.

Table 1. Comparison between network-traffic-based, log-based, and hybrid-based representations.

Representation	Advantage	Disadvantage
Network-Traffic-based	- Data for the detection system are readily available, making preprocessing unnecessary.	- The detection system must inspect every packet received; it requires preprocessing to produce flows. - It exposes confidential packet information, and fewer packet details are available. - It is impossible to match attack signatures with encrypted payload cases, making it ineffective for attack detection. - In the case of massive datasets, representing CCE traffic is difficult. - It is unable to represent enough relevant (valuable) information for all attacks. - Can be extensive in size.
Log-based	- Data for the detecting system are readily available, making preprocessing unnecessary. - Generally contains more information, which includes more attack features. - There are no privacy issues. - Scalable and adaptable in high-speed CCEs. - No encrypted data or payload to affect the detection system's operation. - Represents relevant (valuable) information for attacks.	- Can be extensive in size.
Hybrid-based	- Includes more attack features.	- Conflict in detection systems from two different types of representation. - Increased computation load in detection systems to process two types of representation. - High False Positive rate and low True Negative rate in detection systems. - Conflict and complexity in most attacks' relevant (valuable) information. - Can be extensive in size.

3. Literature Review of Related Works

Datasets play a crucial role in intrusion detection to test, compare, and evaluate IDSs. They are also essential in statistical approaches, ML, artificial intelligence (AI), genetic algorithms, and data mining, which aim to identify the relationships between instances in an aggregated dataset. For instance, many researchers have created and suggested various HTTP datasets to aid in the compression, evaluation, and testing of detection systems when dealing with HTTP-based attacks.

HTTP-GET datasets can be classified as network-traffic-based or log-based datasets. But, most of them were collected from conventional network environments. Publicly available HTTP-GET log-based datasets from actual CCEs are rare. As a result, researchers must rely on publicly benchmarked datasets such as DARPA, NSL-KDD, and KDD to evaluate the effectiveness of HTTP-GET flood DDoS attack-detection systems. However, many realistic datasets lack the necessary features to construct an accurate IDS approach, such as confidential and personal data that are not publicly available. Moreover, current datasets may not represent contemporary technology.

Researchers use one or more datasets depending on the solution's demands to evaluate a detection system's ability to differentiate normal and abnormal requests. However, most of the current datasets can only evaluate and test conventional environment solutions because they do not cover the characteristics of CCEs. Additionally, most solutions that depend on network-traffic-based datasets cannot detect the HTTP flood DDoS attack traces buried inside the WS logs.

CCE security researchers often face a major challenge when securing the application layer: the lack of HTTP-GET flood DDoS attack datasets. This scarcity forces them to create custom HTTP flood DDoS attack datasets for research purposes. However, these self-generated custom datasets often fall short of benchmark datasets regarding accuracy, reliability, and validity, limiting their usefulness for scholarly endeavors. Based on the datasets used in the previous works, we present the related existing works as the following categories:

3.1. Existing Works Using Benchmark Log-Based Datasets

In their research paper, [33] attempted to create and assess HTTP flood DDoS attacks in a CCE environment by designing a testbed structure to cover and evaluate all possible attack scenarios. They used the OpenStack CCE testbed and the World Cup Football (FIFA Cup 98) dataset for their experiments.

3.2. Existing Works Using Benchmark Network-Traffic-Based Datasets

In their research paper, the authors of [34] used flow-based data and ML classifiers to detect HTTP DDoS attacks in the CCE. They tested four tree-based classifiers: AdaBoost, XGBoost, Random Forest, and Decision Tree, and they used the CIDDS-001 dataset for training and evaluation. They found that Random Forest was the best classifier with a 99.99% accuracy.

The authors of [35] proposed an approach that, if applied, may mitigate DDoS attacks in CCEs based on the traffic rate. They adopted the CICDDoS2019 dataset to train ML-based classifiers (logistic regression, support vector machine (SVM), and Random Forest), allowing the proposed approach to select the best one.

Another study proposed a method to protect a website's initial page by identifying a distinct kind of HTTP-based EDoS attack [36]. The adversary puts an astronomical amount of requests in the index page to increase the demand for resources, which exploits the elasticity of the CCE. If the resources are not elastic, it is a DDoS attack instead. The authors evaluated and parsed human browsing behavior by using the DARPA DDoS dataset to minimize this attack and distinguish between normal and abnormal requests. The evaluation and parsing helped them develop methods from strict to mild index page prevention.

The authors of [37] worked on detecting DDoS attacks in CCEs by using a black hole optimization algorithm and artificial neural networks (ANNs) for their approach. They

evaluated their approach on the NSL-KDD dataset, trained their model on 12,500 samples, and tested it on 2597 samples. They repeated their experiments 10 times and obtained the best detection accuracy of 96.30%.

The author of [38] presented a method to detect low-rate attacks in CCEs by using the Hidden Markov Model–Random Forest (HMM-RF). This method uses the HMM to extract features from the network traffic flow and trains the random classifier with these features. In addition, the method uses the Renyi entropy to estimate the attack probability and the HMM to assess the attack severity. Depending on the severity, the method trains the RF with the bootstrap aggregation technique to identify the attacked traffic flow. The authors tested their method on the KDD CUP 99 dataset, a standard dataset for network intrusion detection. Their method performed better than the Adaptive Threshold-Based Algorithm (ATBA) and Artificial Bee Colony–Artificial Neural Network (ABC-ANN) regarding the classification accuracy for recall, precision, specificity, accuracy, and F-measure.

The author of [39] presented a method to detect DDoS attacks by using a Radial Basis Function–Neural Network (RBF-NN) detector. This method applies the Bat algorithm (BA) to automatically configure the RBF-NN, which can find the optimal network structure and parameters based on the given goal function. The method tests its performance by using the KDD CUP 99 dataset, which resembles the DARPA LLS DDoS dataset. The dataset contains Smurf, HTTP flood, and UDP flooding attacks. The method classifies the web traffic data as normal or attack traffic. The BAT-RBF achieved a higher classification accuracy than the RBF and GA-RBF regarding the recall, precision, specificity, accuracy, and F-measure. The average values showed that the BAT-RBF was 0.6% more accurate than the RBF and 0.55% more accurate than the GA-RBF.

In [40], the author investigated the utilization of TensorFlow, a machine learning library, to identify HTTP DDoS attacks. Their method involved preprocessing the network traffic data by using the NFStream tool to convert them into vectors. The machine learning model trained with TensorFlow achieved a remarkable 96.54% accuracy at detecting attack transactions. It also successfully detected 92.85% of the attacks in real-life network-traffic and attack samples. The dataset employed is named the "Attack" dataset, comprising labeled attack samples, including SYN flood, UDP flood, HTTP, SCAN port, and others. The model exhibited high precision at 98.108% and an impressive F1 score of 97.997%. Notably, the model's accuracy remained consistent across the different data types, showing only minimal variations compared to the dataset used.

3.3. Existing Works Using Self-Generated Log-Based Datasets

The research in [41] used model-driven engineering methodologies to create testbeds for the IDS assessment. They proposed a method called a Radial Basis Function–Neural Network (RBF-NN) detector by using the Bat algorithm (BA) to configure the RBF-NN automatically. The BA can identify the appropriate network structure and parameters based on the supplied goal function. The authors used predetermined setup scripts and Terraform v0.10 software to create the testbed infrastructures. They deployed them as VMs on a CCE platform in OpenStack v11. They used the KDD CUP 99 dataset to evaluate their method. The dataset was similar to the DARPA LLS DDoS dataset, which includes Smurf, HTTP flood, and UDP flooding attacks. The collected web traffic data were categorized as normal or attack traffic. The BAT-RBF had a higher classification accuracy than the RBF and GA-RBF regarding the recall, precision, specificity, accuracy, and F-measure.

The authors of [42] created testbeds for the IDS assessment by using model-driven engineering methodologies. They generated datasets (AIT Log DataSet/Online Available) that include log data instead of network traffic. This way, they can assess host-based IDSs, which need log-based datasets. The WSs gather log data that contain daemon logs, syscall logs collected with the Linux audit daemon, Suricata logs, auth logs, Exim logs, mail logs, Apache access and error and user logs, and syslogs.

3.4. Existing Works Using Self-Generated Network-Traffic-Based Datasets

In their research paper, the authors of [43] proposed a method that solves both inside and outside slow HTTP DDoS attacks in the CCE, depending on the number of connections. They incorporated the proposed work within the OpenStack IaaS model to protect WSs from potential attack scenarios. Their self-generated datasets, AIT Log DataSet, are publicly available online and include log data instead of network traffic, which allows them to assess host-based IDSs that require log-based datasets. The authors used slowHTTPtest to generate the dataset to evaluate this work. The dataset includes many attacks, such as slow-HTTP-read, slow-HTTP-body, and slow-HTTP-header DDoS attacks.

The authors of [44] proposed an approach to detect slow-rate HTTP DDoS attacks in the CCE, which uses a self-generated network-traffic-based new dataset with an IDS for evaluation purposes; it uses Tor Hammer as an attacking tool and collects the traffic with Snort. There are nine features and four classes in the dataset. The dataset was applied to different ML algorithms, including Random Forest, SVM, and Naive Bayes. The SVM outperformed Random Forest and Naive Bayes in terms of accuracy and precision. The overall accuracy of the algorithms was as follows: SVM (99.7%), Naive Bayes (98.0%), and Random Forest (97.6%). These algorithms' performance, such as their recall, precision, accuracy, and so on, is evaluated by using the confusion matrix created by these algorithms.

The authors of [5] proposed a dynamic-entropy-based HTTP flood attack-detection approach that uses trace logging (a trace log) to keep track of all the network packets from sender to receiver via CCE-based VM instances and to monitor the VMs' status in real time; specifically, the active IPs of the incoming requests are determined by changing the number of time slots (window size) based on monitoring the sliding window of the dynamic entropy and traffic load. The proposed approach experiment results were compared with the current approaches, notably the adaptive negative selection algorithm and static entropy. It was noticed that the proposed method detects HTTP flood attacks with a high possibility, improving performance even when spoofing attacks are present and decreasing false alarms. Furthermore, the approach was compared to the optimized National Security Agency (NSA) algorithm, which detects the attacks more correctly since the NSA algorithm misclassifies legitimate users as attackers and employs a lot of features to detect the attack.

In their study [45], the authors proposed a method for the fast detection of HTTP-GET flood DDoS attacks in a CCE by integrating Hadoop MapReduce. They compared the execution time (processing time) of their proposed approach with Snort detection by using the pattern detection of the attack features. The results showed that their approach outperformed Snort as its execution time was shorter when congestion increased. The authors used self-generated network-traffic-based datasets to evaluate their work. They followed the firewall policy and routed the allowed traffic among the user-request traffic to the Hadoop master node. After the packet classification, they preprocessed the transmission traffic. The signature used the Map Function to send the preprocessed packets to the data node for parallel pattern matching. The packet log data were stored in the Hadoop data node of the Hadoop distributed file system (HDFS). Finally, the Hadoop master node conducted an anomaly detection by using the Hadoop MapReduce Function with a time schedule and the same file size.

In the work of [46], the authors proposed a novel approach to detecting slow HTTP DDoS attacks in the CCE and notifying the administrator in the event of attacks. The CCE platform of OpenStack was used to implement the solution. Moreover, the slowHTTPTest tool was used to launch attacks on the WS (NGINX) to create the dataset required to evaluate the detection system. The findings of the experiments showed that the attacks can be detected early on.

In the work of [47], the authors proposed an approach to detect network- and application-based flooding attacks based on a fuzzy-logic-based protection mechanism. Moreover, the authors used a self-generated network-traffic-based dataset to evaluate the work. In the experiment, the algorithm learned by using training data, and then rules were created based on the potential traffic patterns of the CCE so that the system may

deduce the traffic class based on the knowledge gained. The authors adopted fuzzy rules based on predetermined traffic features that differ considerably between an attack and a normal traffic pattern. Nevertheless, for every data center, the features may be altered depending on specific requirements, and rules can be defined accordingly. As a result, defense methods should be updated and adjusted regularly.

In the work of [48], the authors proposed an approach that works similarly to a service broker within an SOA model. The proposed approach is named filtering tree. It converts the user request in XML tree form and uses the virtual cloud defenders to protect against these attacks. The CCE users create their requests in XML, submit them over HTTP, and create system interfaces by using REST methods such as Microsoft Azure or Amazon EC2. This study aims to offer security to the Open Application programming interface (API) against XML-, HTTP-, or REST-based attacks.

In their study [49], the authors proposed an approach to protect the CCE from X-DoS/H-DoS attacks. Their approach uses Cloud TraceBack (CTB) to trace and detect these attacks. Additionally, they trained a Cloud Protector by using a back-propagation algorithm for feed-forward neural networks to detect and filter the attack traffic. The authors used a real attack traffic dataset generated by the StuPot project 2009 [50] (the page of the dataset is not available) for training. In their experiment, to generate the dataset, they set a timer for 1 s and updated the targeted website. They collected the HTTP traffic by using tcpdump and Wireshark (Tshark).

The authors generated a huge labeled corpus that included evidence of 10 different attacks against HTTP and QUIC services in their study [51]. They performed a full review of the security of the HTTP/2 and HTTP/3 protocols, detailing numerous HTTP/2 threats and their relevance to HTTP/3. Furthermore, they examined the efficiency of various attacks and created the "H23Q" dataset, which was particularly designed to measure the security of these protocols. The study also looked at the use of machine learning techniques to analyze the dataset. The results of the research were provided, demonstrating the performance of several classifiers in terms of the AUC, precision, recall, F1 score, and accuracy. The Bagging model performed the best, with an AUC of 77.60% and an F1 score of 68.77%.

3.5. The Existing Works Used Hybrid-Based Datasets

In their study [52], the authors proposed a statistical approach by using the covariance matrix to detect HTTP flood attacks. Their approach includes a training algorithm to construct normal network traffic patterns and a testing algorithm to determine the types of traffic received based on the attack behavior. The authors analyzed their findings by using a confusion matrix to assess the detection performance and provided results for external and internal CCEs. They generated their datasets for the assessment, including normal traffic by allowing ordinary end users to browse the Internet and abnormal traffic by attacking a virtual WS by using the PageRebooter tool. However, they found that the MADM performance in a private CCE was lower than when using the KDD dataset. This difference was due to the KDD dataset being collected under high surveillance and controlled conditions, while their work was not. According to the AUC and confusion matrix results, their experiment demonstrated that the MADM performed well in detecting HTTP flood attacks in the CCE. They also proved that the MADM performance with four thresholds was superior to the MADM performance with three thresholds.

In their study [53], the authors proposed the SBTA approach to detect DDoS attacks on CCEs by using SOA to trace back and detect the source of the attacks. They also proposed the cloud filter approach. Their findings revealed that a protection system combining a cloud filter and SBTA is efficient in CCEs. The authors used two datasets to evaluate their work: the first dataset, called LLDoS2.0.2, was from Lincoln Laboratory at MIT in 2000, and the second dataset was generated in their work. The results showed that combining a cloud filter and SBTA over a CCE could effectively and efficiently identify and detect attack messages. Additionally, it reduced the number of messages required to rebuild the path and compute tasks.

In their study [54], the authors proposed a method to detect HTTP-GET flood DDoS attacks on CCEs by using Hadoop MapReduce with a rule engine and an HTTP packet pattern. Their approach routes authorized the users' request traffic to enter the Hadoop master node via a firewall policy. The authors experimented with the execution time (processing time) to evaluate their method's performance by using pattern detection on the attack features. They compared their approach with Snort detection based on the access log of the WS and HTTP packet patterns to study the web-usage patterns.

In their study [55], the authors proposed a hybrid classification model for the anomaly detection (HyClass) approach to detect attacks by analyzing CCE network traffic by using a hybrid anomaly detection technique. HyClass operates in two stages: the first stage uses the Boruta algorithm for feature selection to improve the accuracy and classification efficiency, while the second stage uses an SVM with the Chaotic Optimization and Differential evolution algorithm for classification. The authors evaluated their approach's performance by using two different datasets: The first dataset, a real-time TU dataset, was captured by using Dumpcap (Wireshark) from the network traffic of a University and included packets from 400 hosts. The second dataset was the benchmark KDD'99 dataset. Their findings showed that their proposed approach was effective and reliable at detecting anomalies regarding the False Positive rate, accuracy, and detection rate in real-time scenarios.

In their study [56], the authors proposed a flow-confidence-based discrimination approach to CCE servers hosting multimedia services such as audio and video streaming. Their approach discriminates between DDoS attacks and flash crowd events and has been proven successful, efficient, and cost effective in ensuring users' QoE during flash crowd events in real-world scenarios. The authors used two benchmark datasets, CAIDA and FIFA World Cup 98, to evaluate their proposed approach and support their theoretical claims.

In their study [27], the authors developed a deep-learning-based classifier to detect DDoS attacks in CCEs. Three datasets were used to test the proposed TEHO-DBN classifier: the KDD cup database, a synthetic database created for DDoS detection, and a server log information database. The user's service requests were collected and grouped as part of the log information. The Bhattacharyya distance measure was used to reduce the training time of the classifier by selecting some important features from the log file.

In their study [57], the authors proposed a fuzzy bat clustering approach that uses a deviated anomalous score to detect HTTP flood DDoS attacks. Their approach inspects attacks by grouping similar input patterns and analyzing them by reading logs to extract the related features. The authors generated a dataset to evaluate their work by using various attacking tools conducted through VMs by sending harmful packets to the targeted WS. They configured Snort in the node controller to monitor the VM network traffic activities. The network traffic log and access log trace were used to determine the evidence source. The authors identified the suspect source by locating the event correlation between the suspicious source list and the VM supplied by a cloud service provider.

In their research paper [58], the authors proposed a hybrid ML-based approach to detect DDoS attacks in CCEs by using black-hole optimization and extreme ML (ELM) models. They evaluated their approach's performance by using various benchmark datasets and achieved good results: CICDDoS2019 with an accuracy of 99.80%, CICIDS2017 with an accuracy of 99.50%, ISCX IDS 2012 with an accuracy of 92.19%, and NSL KDD with an accuracy of 99.23%. The authors also compared their approach's performance with various systems based on back-propagation ANNs, ANNs trained with black-hole optimization, and ELM models.

In their study [59], the authors proposed a voting extreme learning machine (V-ELM) approach to protect CCE services by detecting DDoS attacks on CCEs. Their approach monitors incoming and outgoing network traffic to detect floods of fake request packets. A V-ELM is a type of ANN. The authors evaluated their approach's performance by using two benchmark datasets: the ISCX intrusion detection dataset with an accuracy of 92.11% and the NSL-KDD dataset with an accuracy of 99.18%. They also compared their approach's performance with other approaches based on AdaBoost, Random Forest, the extreme

learning machine, an ANN trained with black-hole optimization, and a back-propagation ANN. Their proposed approach performed better than these approaches. Additionally, they analyzed their approach's performance by using different parameter values, such as the number of hidden layer neurons in a single ELM and the number of ELMs in a V-ELM.

In their study [42], the authors proposed an approach to detect DDoS attack traffic in private CCEs by checking against a threshold of requests by using Servlet and XML Coding. In their experiment, their approach prevented the source of the attack on a website supported by the Cloud Flare web application firewall within 10 min. They decreased the time to stop the attack to around 10 s and confirmed their findings' validity. The authors evaluated their approach's performance by launching a DDoS attack against a web application by using a tool and Java code on Templates. They analyzed and understood the traffic patterns stored in the Tomcat WS log files with an access matrix. They used the threshold of the attack traffic in the network that was attempted tp be used on the SCO website for greater accuracy. They estimated and compared the hit rate threshold from NASA datasets to identify the attack with their current study. Their results showed that when the threshold reached the higher traffic limit (value) obtained in the World Cup 98 datasets, they instantly prevented the attack traffic by using XML and Servlet Coding.

4. Dataset Design Principles and Significance

Effectively assessing, comparing, implementing, and deploying new IDSs for HTTP-GET flood DDoS attacks on CCEs by using existing datasets is challenging. Before deploying an IDS in a real environment, it should be evaluated and compared with other superior existing IDSs by using genuine and valid labeled datasets that mimic the real deployment environment by covering all possible attack types and scenarios. However, the lack of efficient benchmark HTTP-GET flood DDoS attacks on CCE datasets is a critical problem for the CCE security community. As a result, the IDSs for HTTP-GET flood DDoS attacks on CCEs are often tested by using self-prepared (self-generated, not benchmark) datasets on CCEs or benchmark network-traffic-based datasets (not CCEs) or benchmark log-based datasets (not CCEs). These datasets' coverage, accuracy, and validity are not guaranteed due to their scarcity compared to valid benchmark datasets.

4.1. Dataset Design Principles

Many issues with the current log datasets produced in testbeds arise from shortcomings in the system infrastructures and environments where the data were collected. Therefore, the testbed design process must adapt to the requirements of the data to be generated. To address this, we followed design principles that served as the foundation for our testbed generation technique. We will briefly discuss each of these principles.

- Reproducibility: to accurately reproduce the log dataset, the testing environment must be able to revert to a previous state.
- Flexibility: Establishing a testbed requires considerable manual effort. As such, it is cost efficient to utilize a flexible and isolated portion of the environment that permits iterative development through modifications and enhancements.
- Authenticity: to guarantee a representative assessment of the IDS capacity on the CCE, realistic log data should be gathered under factual scenarios in CCEs.
- Availability: for the proper benchmarking and evaluation of IDS detection capabilities using various techniques, it is crucial that any generated datasets be made publicly accessible.
- Utilizability: merely having access to datasets is insufficient to evaluate IDSs. It requires ground truth, which involves labeling, numeric representation, and data normalization.

4.2. The Significance of Datasets

In the field of IDSs, a variety of datasets have been utilized. These IDSs can be classified into several categories: ML-based, DL-based, data-mining, and rule-based systems.

For instance, DL-based and ML-based IDSs depend on network traffic data [60], represented by a set of traffic features, to run their detection models. As such, the datasets used in the learning or training and testing stages of IDSs must be validated and efficient. The significance of having datasets for CCE requests can be outlined as follows:

- To illustrate attack behavior: IDSs must comprehend the behavior of illegitimate requests. This aids in the identification of the features and characteristics that differentiate these requests from legitimate ones.
- To ensure experimental repeatability: Repeating experiments by using the same dataset is necessary to enhance a specific detection system. This enables the production of accurate, valid, reliable, and consistent results.
- To validate new IDSs: CCEs are susceptible to various attacks, prompting the creation of new detection solutions. Each unique IDS, however, must be validated and tested against a dataset to assess its reliability.
- To compare new IDSs with existing ones: evaluating the efficiency of an IDS and comparing it to other IDSs by using datasets is essential to achieve significant improvements.
- To effectively tune parameters and configurations: The performance of most IDSs is influenced by specific configurations and/or parameters. Conducting various experiments, such as cross-validations, to determine the optimal parameter values is crucial.
- To determine the most important feature sets: The validity of the selected feature sets positively impacts the detection accuracy. Multiple tests using datasets are required to select the best set of features to represent normal and abnormal requests.

Evaluating IDSs in the context of cloud computing environments is crucial due to the rising prevalence of cloud-based services and their unique security challenges. Cloud environments are complex and dynamic and involve multitenancy, virtualization, and shared responsibilities. The existing evaluation approaches are inadequate because they often lack cloud-specific testing scenarios and features and may not scale effectively in dynamic cloud settings. Additionally, IDSs may be blind to the hypervisor layer, limiting their ability to detect attacks targeting virtualized infrastructure. To address these shortcomings, there is a need for a dataset collected from real-world implementations in CCEs that contains CCE-specific features from multiple sources, incorporates cloud-specific testing scenarios, and considers specific attacks. These efforts will aid in developing effective IDSs tailored to cloud computing environments.

5. Requirements for an Effective Dataset

Obtaining an effective dataset for use with IDSs in CCEs is a challenging endeavor due to the stringent criteria that must be satisfied by any potential dataset. Several key characteristics have been identified to generate and develop an efficient and comprehensive IDS dataset. In fact, to be deemed effective, a prospective dataset must fulfill five requirements [61]. These requirements include:

- **Realistic dataset:** the dataset must be constructed by using a real-life CCE, for instance, a private CCE.
- **Diverse scenarios:** a dataset with varied characteristics, including different sizes and types of attacks, is more reliable, durable, and dependable for comparing and testing IDSs.
- **Accurate and complete labeling:** requests must be accurately and completely labeled as normal or abnormal to properly evaluate the IDSs based on their evaluation metrics, including False Positive rates and detection rates.
- **Balanced and sufficient dataset size:** class labeling should not affect the size of normal and abnormal requests.
- **Representative features:** The dataset should contain a comprehensive set of well-defined features to classify attacks. To ensure the proper validation of the IDSs,

the dataset's features must be relevant and specifically meaningful. Representative features help distinguish between normal and abnormal requests.

In addition, with respect to heterogeneity, the dataset should be gathered from multi-sources that are relevant and impact the environment during the attacks in order to cover all aspects of the carried-out attacks and to be used to detect the attacks.

6. The Proposed HTTP-GET Log CCE Dataset Methodology

In the field of intrusion detection, IDSs typically analyze network traffic or requests to detect any suspicious activity. However, the currently available HTTP attack datasets collected from conventional environments are not suitable to evaluate IDSs' performance against HTTP-GET flood DDoS attacks on CCEs, as discussed in Section 2. Therefore, a substitute dataset that meets the requirements to evaluate IDSs against HTTP-GET flood DDoS attacks on CCEs is necessary. This dataset should have relevant and high-quality features and should be suitable for investigating and researching HTTP-GET flood DDoS attacks on CCEs. For it to be considered a reliable benchmark to evaluate research that aims to secure the HTTP on CCEs, the dataset must meet the standards outlined in Sections 3 and 5.

This section discusses the proposed dataset, including its preparation. The proposed dataset was produced through a three-stage methodology, as illustrated in Figure 4.

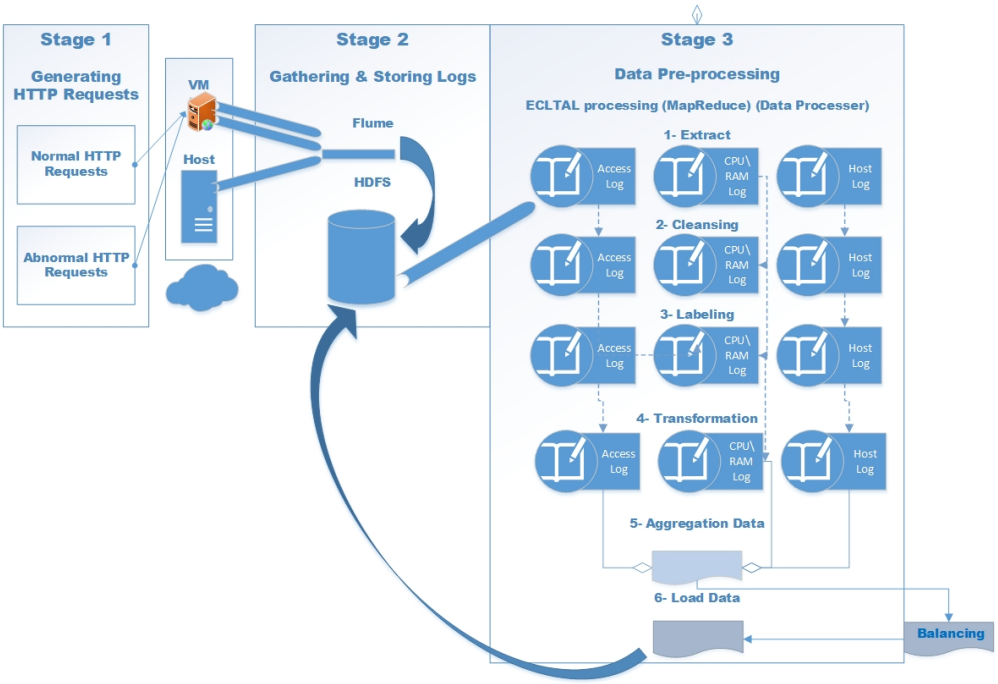

Figure 4. Methodology of the proposed dataset.

The HTTP-GET Log CCE dataset used to evaluate our proposed method to detect DDoS attacks in CCEs comprises three types of logs: the WS access log, CCE-VM activity log, and VM resource log. We created a WS and hosted it on a private CCE. We then generated 200,000 requests to access the WS from external and internal sources. Of these requests, 802 were normal and the rest were abnormal, meaning they were part of an HTTP-GET flood DDoS attack. We labeled the requests as normal or abnormal based on their request status, CCE-VM activities, and resource statuses. We also extracted 26 features from the logs, as shown and described in Table 2. The last column (27th) is the class label. We aggregated the logs by their timestamp for each request, as shown in stage three of Figure 4.

Table 2. Fields and descriptions of the proposed dataset.

Features	Features' Source	Event Records	Field	Description
Existing	Access log	(HTTP-GET) requests	Client_IP	The IP address of requesting host
			Client_Identity	Determines the user's identity
			User_ID	The HTTP authentication determines the username
			Date	The date when the request was received
			Time	The time when the request was received
			Zone	The time zone
			Method_type	The request method of HTTP used in each request (GET/POST/etc.)
			Request	The user's request
			HTTP_Protocol	The request protocol used
			Status_Code	The status of request for user
			Size_of_Response	The size of the response
			Referrer	The page that linked to this Uniform Resource Locator (URL)
			User_Agent	Type of Internet browser and OS used to access the website
New	CCE log	VM Activity	VM_CPU	The number of CPUs allocated for the VM
			VM_RAM	The size of RAM allocated for the VM
			VM_vMotion	The motion (migration) of the VM over the hosts
			VM_clone	Number of instances of the VM
	VM resources log	VM central processing unit (CPU)	CPU_usr	The CPU usage by a user's processes
			CPU_sys	The CPU usage by system's processes
			CPU_idl	The number of idle processes in the CPU (total usage)
			CPU_wai	The number of waiting processes in the CPU (total usage)
			CPU_stl	The CPU usage by CPU stl
		VM random access memory (RAM)	RAM_used	The total usage of RAM
			RAM_free	The total nonusage of RAM
			RAM_buff	Memory used by kernel buffers
			RAM_cach	Memory used by the page cache and slabs

6.1. Generating HTTP-GET Requests

The protocol of the HTTP is a request-and-response protocol with a client and server architecture in which search engines, robots, and web browsers behave as HTTP clients and the WS works as a server. The server receives an HTTP request from an HTTP client as a request message that includes information such as the request method. The request method, such as GET or POST, denotes the manner used on the resource defined by the request Uniform Resource Identifier (URI). The HTTP-GET method is employed to obtain data through the provided URI of a WS. HTTP-GET requests should solely reclaim data and have no other influence on them.

We can generate HTTP-GET requests (normal/abnormal) to build the dataset in various environments, such as traditional, virtual simulation software (CloudSim, CloudAnalyst, Greencloud, and iCanCloud) and CCEs. The traditional and virtual environments are the least complicated, but the simulation software is the simplest and fastest build environment because it does not require setting up physical or virtual machines. However, these environments cannot mimic the natural CCE behavior, as the requests may contain an unrealistic representation of CCEs and biased characteristics that do not reflect real-world network conditions and traffic patterns.

Thus, these environments do not appear viable for creating an effective dataset of HTTP-GET flood DDoS attacks on CCEs. Another option is to build up and install a full-fledged actual CCE, hosts, and network equipment. This option can provide the highest level of realism in creating the dataset. However, procuring and managing the entire CCE is prohibitively expensive. Therefore, the last alternative is a compromise between these two options, which involves employing an existing CCE while taking utmost care not to disrupt its production or functionality. Figure 5 shows the topology of the used private CCE.

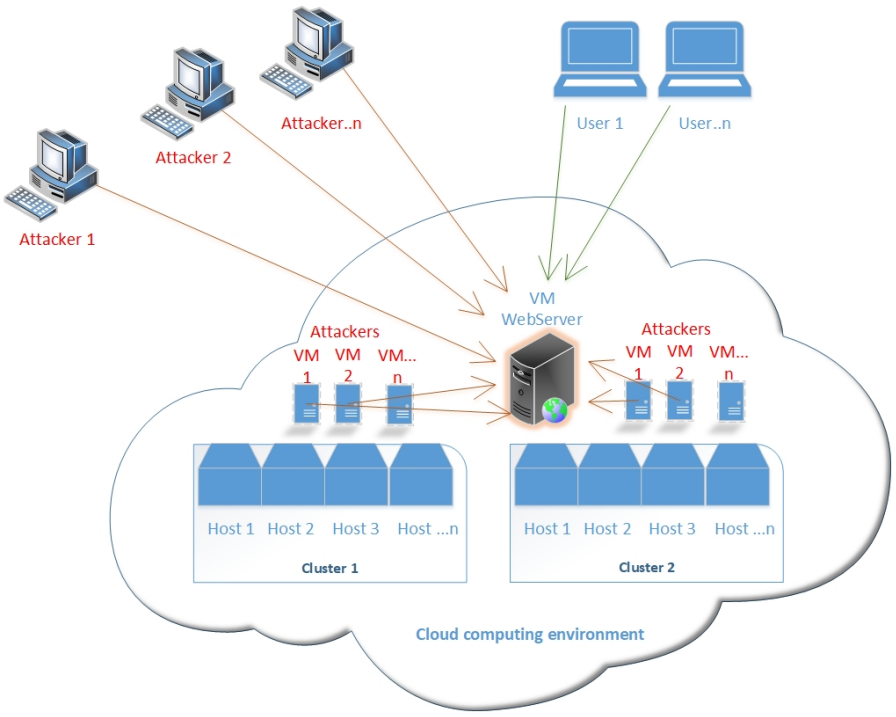

Figure 5. Private CCE topology to generate HTTP-GET requests.

- **Generating Normal HTTP-GET Requests:** Generating normal requests can be achieved through various means. One common approach is to utilize existing commercial tools that manually generate requests. However, such tools cannot fully replicate natural user behavior, resulting in requests with unrealistic and biased characteristics. Moreover, they cannot simulate human actions like typing and rerequesting, rendering them unsuitable for creating effective datasets. An alternative method is to employ human users to create requests through browsers, which can provide high realism when generating datasets.
  As highlighted in Section 5, achieving realism is crucial to create high-quality datasets. This entails producing requests that closely resemble real-world behavior. To accomplish this, we developed our dataset by using a private CCE environment as the source of the normal requests. The dataset includes HTTP requests made by two users employing the most popular browsers (Microsoft Edge, Firefox, and Google Chrome) from two external computers outside the CCE. The topology of the private CCE used in this study is illustrated in Figure 5.
- **Generating Abnormal HTTP-GET Requests:**
  To create an effective dataset, including a diverse range of attack and normal request scenarios is crucial. However, achieving this in a real-world environment can be challenging due to security concerns and restrictions. To address this, we conducted HTTP-GET flood DDoS attacks within the same CCE to ensure that the produced requests contained realistic and biased characteristics specific to the CCE. This is essential to create a truly realistic dataset that can be used to evaluate and compare the effectiveness of different research works.

While HTTP-GET flood DDoS attacks can potentially cause significant harm to a weakly protected CCE, we conducted these attacks within the same CCE to create realistic datasets. Figure 5 illustrates the topology of the private CCE used in this study. Our proposed dataset's HTTP-GET flood DDoS attacks were intentionally malicious and designed to cause harm. We launched these requests by using Goldeneye, one of the most widely used tools for conducting HTTP-GET flood DDoS attacks. The attacks targeted victims with HTTP-GET flood DDoS requests originating from multiple attacker sources, simulating a realistic HTTP-GET flood DDoS attack scenario.

Including a diverse range of attack scenarios ensures that all conceivable attack cases and behaviors are represented and included, thus contributing to accurate detection models.

Given that the proposed dataset is meant for CCE protection and attack detection on the application layer, it would be incomplete unless several attacking scenarios were performed in the CCE. The attacking machines (VMs) used in this study were built inside and outside the CCE, as shown in Figure 5.

All the available HTTP-GET flood DDoS attacks by Goldeneye were performed in this dataset by using seven attacking machines, virtual and nonvirtual, both from inside and outside the CCE.

6.2. Gathering and Storing Logs

This stage involves collecting and storing all the logs from the sources that changed during the HTTP-GET request generation. These sources include the access log, the VM activities log, and the VM resources log. These logs have large volumes (vast) and require a proper solution to handle them. Figure 4 illustrates the method of the proposed dataset.

The Hadoop ecosystem is a common solution for big data [62]. It offers a high-availability distributed solution for big data challenges. Several platforms use Hadoop to spread their data. These platforms are Microsoft Azure HD Insights, Apache Hadoop, IBM BigInsights, Hortonworks, and Cloudera tools [63]. Hadoop has the main components, such as a software system for distributed storage called HDFS, a data-collecting tool called Flume, and a MapReduce programming model to process big data.

This solution can manage a large volume and complexity of raw data logs from different sources. It can take these data logs and put them into HDFS by Flume. Then, it can process them through MapReduce (the data processor).

- **Gathering the log using Flume**
  In this stage, we used Flume [64,65] to collect and store all the logs from the sources that changed during the HTTP-GET requests generation. These sources are the access log, the VM activities log, and the VM resources log. These logs have large volumes (vast) and require a proper solution to handle them. Using Flume can improve the detection and prediction of DDoS attacks [63]. Furthermore, Flume can gather and aggregate large and complex amounts of log data from multiple sources without changing or affecting the original logs during the transfer from sources to the storage system.
- **Logs repository**
  The HDFS supports fast repository storing and the splitting of vast and complex data to guarantee the availability of logs data on the Hadoop nodes, which helps with parallel processing as the system can equip it and allow MapReduce [63,64] to be feasible for the analysis; additionally, it can filter out the basic features and convert the raw data logs into a clean dataset that is easily readable without modifying the content.

6.3. Data Preprocessing

The stored data in the HDFS must undergo preprocessing before testing and implementation with any classifiers. The processing in MapReduce is conducted in parallel, meaning simultaneously on several distributed Java-based devices, allowing it to process massive quantities of data in a very short time reliably with a fault tolerance [63,64]. The Hadoop data processor (MapReduce) [66] will perform the extraction, cleansing, labeling, trans-

forming, aggregating, and loading data (ECLTAL) processes inside the HDFS. The ECLTAL process processes and analyzes the data in the following steps:

- **1. Extract:** This step involves extracting and initializing the data stored in the HDFS for processing. The main purpose of this step is to extract only the necessary data by filtering the significant records from log files and excluding any irrelevant or redundant features from the HDFS by using as few resources as possible. The desired data are identified and separated from the other log records during extraction. In this research, the data were extracted from three sources: access logs, CCE logs, and VM resources logs. A total of 26 features were extracted from these resources, as shown in Table 2. These features include both those used in the existing research and new features proposed based on experimental observations of HTTP-GET flood attacks on CCEs. It is important to note that these 26 features constitute the proposed dataset and are considered significant at detecting HTTP-GET flood attacks. The newly proposed features are specific to CCEs, giving our dataset a unique advantage compared to existing datasets that lack CCE-specific features. This distinction is crucial as it can significantly impact the performance of IDSs.
- **2. Cleansing:** This process is commonly referred to as data cleaning or data scrubbing, and its purpose is to prepare the data for direct use in detection systems. In this research, the collected data from access logs, CCE logs, and VM resources logs undergo a cleaning process. During this process, erroneous, missing, or inconsistent values are identified and resolved.
- **3. Labeling:** A labeled dataset is a collection of samples marked with one or more labels. Labeling generally begins with a collection of unlabeled data and tags are added to each piece. The proposed dataset is extremely useful in evaluating detection systems primarily relying on labeled data (i.e., requests). In this research, the records in the proposed dataset were labeled as normal or abnormal. This labeled dataset can be used to evaluate the performance of supervised machine-learning-based IDSs.
- **4. Transformation:** The transformation step applies a rule set before the data transformation. MapReduce is commonly used to transform big data by using parallel data processing to transform the data format, such as by removing symbols from date and time values and numerical values and normalizing the numerical features. The transformation step consists of the following substeps:
  - **Formatting:** The collected data may contain unwanted symbols such as brackets (e.g., [] () {}), double quotations (e.g., “, ”), or colons (e.g.,:, ;). These symbols are inconsistent with the purpose of using the dataset effectively. Therefore, removing useless characters (symbols) is very important as they may skew the IDSs’ performance.
  - **Numeric:** Numerical data refer to data offered in the format of numbers instead of in any other form or descriptive writing. In this step, all categorical data were transformed to numerical data before being used to evaluate the performance of the IDSs.
  - **5.: Feature Scaling:** In the process of feature scaling, the features are rescaled to make them more suitable for training. Scaling is crucial during training as it ensures that all the features are within the same range. Several scaling strategies, such as Mean Normalization or Standard Scaling, can be used to achieve this. In this research, MinMax scaling was chosen for this problem after thorough testing, as it demonstrated a superior performance. MinMax scaling can be calculated by using Equation (1):

$$Value\_Normalized = \frac{(Value - Value\_Min)}{(Value\_Max - Value\_Min)} \quad (1)$$

  - **6. Aggregation:** The aggregation of the data is the process of combining the three logs (WS access, VM activities, and VM resources) and presenting it in a

summarized and formatted manner in one file based on the time stamp, such as a CSV file (rows/columns) as required for the destination usage, to provide the final dataset for evaluation. The volume and quality of the data gathered determine the accuracy of the data analysis and the resulting outcomes. Thus, data aggregation is a critical step in dataset processing.

– **7. Load Data:** Loading the dataset as a CSV file is the second last step in the ECLTAL process. This step loads the processed dataset into a new directory (path) in HDFS to evaluate the dataset through classification algorithms.

The additional step of data preprocessing is balancing, which involves balancing the data under two classes of labels: normal and abnormal. A common balancing technique employed in the WEKA platform is outlined in [67].

In SMOTE, the fewer labeled records are replicated to oversample the data. A classifier is improved through replication and adding random instances based on the 'well-known' distribution of the classes in a dataset. It was utilized with the WEKA platform to balance the proposed dataset, as shown in Figure 6.

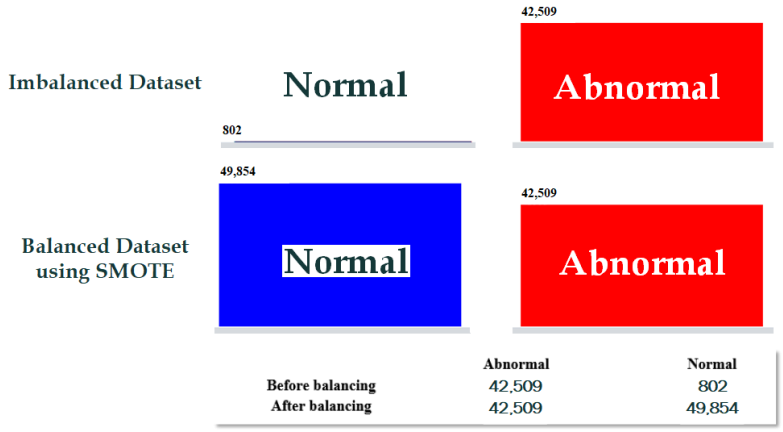

Figure 6. Example of WEKA dataset's SMOTE balancing technique.

6.4. Dataset Evaluation

In this section, for practical purposes, the proposed HTTP-GET Log CCE dataset is evaluated with nine classification algorithms to prove that using it side by side with AI detection engines is possible. Because the primary purpose of these experiments is to demonstrate that the proposed dataset is accurate and credible when used to assess detection methods for HTTP-GET flood DDoS attacks on CCEs, the classifiers chosen are simple ones with no enhancement or parameter tuning because the goals of these experiments are to demonstrate that the dataset is trustworthy and dependable enough to be employed to assess the detection methods of HTTP-GET flood DDoS attacks on CCEs. Furthermore, we seek to demonstrate that the proposed features can competently differentiate normal behaviors and HTTP-GET flood DDoS attack behaviors on CCEs.

The following two subsections present the definition of the performance/evaluation metrics and classification/evaluation algorithms of the proposed HTTP-GET Log CCE dataset.

6.4.1. Evaluation Metrics

The proposed dataset was evaluated by using nine evaluation metrics to demonstrate the Log-based dataset representation's effectiveness and features. These metrics are the True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, F1 score (F-Measure), Matthews Correlation Coefficient (MCC), Receiver Operating Characteristics (ROC), precision–recall (PRC), and detection accuracy (DA).

The power to successfully classify attack records as an attack from all of the existent attacks is measured by the DA. Equation (2) [39] can be used to compute the DA:

$$DA = \frac{TP + TN}{TP + TN + FP + FN} * 100\% \quad (2)$$

where TP = True Positive, FN = False Negative, FP = False Positive, and TN = True Negative.

The failure rate of classifying normal requests (instances falsely classified as a given class) as normal requests from the total number of normal requests is known as the FPR. Equation (3) can be used to calculate the FPR:

$$FPR = \frac{FalsePositive}{FalsePositive + TrueNegative} * 100\% \quad (3)$$

Precision is another evaluation metric that represents the value of correct positive outcomes divided by the classifier's predicted number of positive results, computed by using Equation (4):

$$Prec = \frac{TruePositive}{TruePositive + FalsePositive} \quad (4)$$

All the positive samples that should have been reported (recall) are calculated by dividing the number of valid positive results by the total number of relevant samples and can be computed by using Equation (5):

$$Rec = \frac{TruePositive}{TruePositive + FalseNegative} \quad (5)$$

The accuracy of a test is verified by the F-Measure. The Harmonic Mean of the precision and recall is the F-Measure (calculated based on the precision and recall). The F-Measure has a range of [0, 1]. It indicates how robust the classifier is (it does not ignore a large number of instances) along with its precision (how many instances it successfully classifies). High precision but low recall offers an incredibly accurate result but also misses many difficult-to-classify instances. The higher the F-Measure, the better the model's performance. The F-Measure aims to figure out a balance between both precision and recall. It can be stated mathematically as Equation (6):

$$F1score = \frac{2 * Prec * Rec}{Prec + Rec} \quad (6)$$

The correlation coefficient for MCC represents a measure of quality between the dependent classes. Unlike sensitivity (recall), precision, and accuracy, its value ranges from minus one to one. A value near minus one indicates a weak prediction, a zero value indicates a fully random prediction, and a value near one indicates a strong and good performance prediction. It can be stated mathematically as Equation (7):

$$MCC = \frac{TP * FN - FP * FN}{(TP + FP)(TP + FN)(TN + FP)(TN + FN)} \quad (7)$$

The ROC area measurement is a measure of classifier performance, and several plots provide visual representations. It is one of the essential values output by WEKA. It gives an idea of how the classifiers are performing in general. But, interpreting ROC curves requires particular caution when used with imbalanced datasets.

When evaluating binary classifiers on unbalanced datasets, the PRC plot is more informative than the ROC plot, which is a graph depicting the tradeoff between the TPR and FPR. Basically, we calculate the TPR and FPR for each threshold and plot them on a single chart. Naturally, the higher the TPR and the lower the FPR for each threshold, the better; therefore, classifiers with more top-left side curves are better.

The TPR, also known as the sensitivity, measures how accurate a test is. The TPR is the probability that an actual positive will test positive (instances correctly classified as a given class).

6.4.2. Analyzing the Performance of ML Models

The evaluation metrics in this research were calculated by using a range of commonly used ML classifiers. To avoid the need for reimplementation, the classifiers utilized in this experiment were taken from the widely used WEKA tool [68]. The classifiers chosen were distinct from each other based on their type, the number of features, and the classification performance. The selected classifiers were Naive Bayes [69], Decision Tree (J48) [70], SVM [71], K-nearest neighbors (KNN) [72], neural networks [73], Random Forest [74], Random Tree [75], Single Conjunctive rules [74], and Naive Bayes Multinomial [75].

The performance of these supervised ML classifiers is compared in this research, which is known to have excellent accuracy levels with minimal computing costs. These classifiers are described in detail below.

7. Result Discussion

The purpose of this section is to evaluate the effectiveness of the proposed dataset at evaluating the performance of the various ML classifiers listed in Section 6.4.2. Furthermore, we aim to benchmark the proposed dataset against existing well-known datasets to highlight the uniqueness of the proposed dataset. This section comprises the following subsections.

7.1. Quantitative Comparison

In this section, we assess the performance of various ML classifiers by using the dataset proposed in our study. The primary objective is to determine how well each classifier performs at detecting and classifying instances within the dataset and to quantify their performance metrics. To achieve this, we employed the classifiers listed in Section 6.4.2 with their default parameters. These classifiers were selected for their ability to handle both binary and multiclass classification tasks effectively.

Tables 3 and 4 present the evaluation metrics and findings obtained by using the most commonly used classifiers for classification tasks on the proposed dataset. Two approaches were utilized to test the classifiers: the "supplied test set" and the "cross-validation test".

The "supplied test set" involves training the classifiers on one dataset (the training dataset) and then testing them on a separate dataset (the testing dataset). Specifically, 70% of the data are retained for classifier training, and the remaining 30% are used for testing to calculate the model's accuracy.

Table 3. Evaluation metrics of executing the classifiers of the proposed dataset (supplied set test).

Classifier	TPR %	FPR %	Prec %	Rec %	F1 %	MCC %	ROC %	PRC %	DA %
Naive Bayes	0.940	0.051	0.946	0.940	0.940	0.886	0.997	0.994	93.9625
Decision Tree (J48)	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
SVM	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
KNN	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
Neural networks	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
Random Forest	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
Random Tree	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	99.9964
Conjunctive rules	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
Naive Bayes Multinomial	0.914	0.082	0.920	0.914	0.914	0.834	0.979	0.981	91.4248

Table 4. Evaluation metrics of executing the classifiers on the proposed dataset (cross-validation test).

Classifier	TPR %	FPR %	Prec %	Rec %	F1 %	MCC %	ROC %	PRC %	DA %
Naive Bayes	0.939	0.053	0.945	0.939	0.939	0.884	0.997	0.994	93.8915
Decision Tree (J48)	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
SVM	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
KNN	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	99.993
Neural networks	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
Random Forest	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Random Tree	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	99.993
Conjunctive rules	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	100
Naive Bayes Multinomial	0.967	0.029	0.968	0.967	0.967	0.935	0.991	0.992	96.729

On the other hand, the "cross-validation test" (with ten folds) creates multiple samples from the training dataset. The model is repeatedly run ten times, with one fold saved for validation and the other nine folds used for model training in each run. The average of all ten folds yields the cross-validation result. This approach trains the model on various random data selections, enhancing its reliability and robustness [76].

By employing these evaluation methods, this study aims to comprehensively assess the performance of the classifiers and determine their effectiveness at handling the proposed dataset for classification tasks. Using supplied test sets and cross-validation tests ensures a thorough evaluation of the classifiers' capabilities and provides valuable insights for model selections and performance comparisons.

Tables 3 and 4 show that the proposed dataset and features performed efficiently in all the classifiers tested. Furthermore, the obtained evaluation metrics were strong and consistent across each testing approach, confirming that the employed features and representation are appropriate for detecting this attack.

Some classifiers already achieved excellent results; therefore, not all need improvements. However, researchers can try to improve those that did not attain high values by modifying the features by using feature selection techniques to reduce them in order to enrich the features, extract extra features, or optimize the classifiers. Furthermore, because these classifiers were used with their default settings, parameter adjustment/tuning can enhance the evaluation metrics achieved for the classifiers that did not attain high values.

Based on these findings, it was confirmed that the proposed dataset fulfilled the final requirement of an effective dataset. Furthermore, by applying any suggested classification methods in the optimization zone to the dataset and matching the classification abilities of the approach to actual outcomes, as illustrated in Tables 3 and 4, this dataset can be employed for a classification evaluation of any proposed method in the optimizing field. Furthermore, the provided dataset may be adopted to know the differences between normal requests and HTTP-GET flood DDoS attack distinctions on CCEs to provide additional detection systems.

7.2. Qualitative Comparison

A network-traffic-based dataset comprising the data of the lower layers of the OSI reference model does not contain sufficient information for detection algorithms to distinguish attack and normal HTTP-GET requests. In addition, the scarcity of publicly available log-based datasets on CCEs motivates us to generate one that contains HTTP-GET flood DDoS attacks and normal HTTP-GET requests on CCEs.

Dataset availability is a major issue for ML/DL intrusion detection researchers due to legal and privacy concerns. Existing HTTP datasets in different environments may fulfill the need of a certain group of researchers but often fail to meet the aims of others, and many are not publicly accessible.

Even if available, publicly accessible datasets are often altered to protect users' anonymity, limiting complete data access. Therefore, this research aims to develop an alternative HTTP-based reference dataset that meets the requirements for detecting HTTP GET-based attacks on CCEs and overcomes the shortcomings of existing benchmark datasets. The proposed dataset was evaluated and found to be reliable enough for online access by other researchers.

This section compares the proposed dataset to existing HTTP datasets based on various criteria to highlight the uniqueness of our dataset. These criteria are:

- Availability of two separate dataset versions: Having two versions of the dataset, one for testing and the other for training, is necessary to evaluate and test the detection systems. This approach allows for the simulation of real-world implementation by training the system on one dataset and testing it on another.
- Labeled dataset: Labeling a dataset is crucial to validate the creation and improvement in detection systems. Each record in the dataset must be assigned a deterministic class label that indicates its source (normal or abnormal). An unlabeled dataset is unsuitable for evaluating detection systems' accuracy.
- Availability of CCE configuration information: The CCE configuration includes information on the infrastructure, servers, and devices required to create the CCE and generate requests. This information is vital for CCE security researchers to understand the nature of the requests generated on CCEs and other relevant data to understand the dataset better.
- Online accessibility: The publicly available dataset allows researchers to evaluate and compare their work fairly with other studies using the same dataset. Therefore, the proposed dataset is freely available on the Internet [77].
- Variety of attack scenarios: Evaluating a detection system by using a dataset comprising various attacks ensures that the system has been exposed to diverse attack scenarios. The metrics used to measure attacks, such as the number of requests sent, frequency, destination, source, and time, can differ between attacks.

Table 5 summarizes the qualitative comparisons of the proposed dataset with the previously published datasets, addressing the issues and difficulties of the latter.

Table 5. Qualitative comparison between proposed dataset and existing datasets.

Dataset	Class Label	Attack Type	Attack Scenario	Environment Type	Availability	Data Format	Representation (Source)
Sask-HTTP (Saskatchewan HTTP) [78]	unlabeled	HTTP requests	Nondiverse	Non-CCE	Available with restricted access	Common Log	log-based
NASA-HTTP [79]	unlabeled	HTTP requests (flash crowd)	Nondiverse	Non-CCE	Available with restricted access	Common Log	log-based
ClarkNet-HTTP [80]	unlabeled	HTTP requests	Nondiverse	Non-CCE	Available with restricted access	Common Log	log-based
Calgary-HTTP [81]	unlabeled	HTTP requests	Nondiverse	Non-CCE	Available with restricted access	Common Log	log-based
SDSC-HTTP [82]	unlabeled	HTTP requests	Nondiverse	Non-CCE	Available with restricted access	Common Log	log-based
EPA-HTTP [83]	unlabeled	HTTP GET, POST, HEAD	Nondiverse	Non-CCE	Available with restricted access	Common Log	log-based
CDX 2009 [84]	unlabeled	HTTP GET	Nondiverse	Non-CCE	Available with restricted access	Common Log	log-based
FIFA World Cup 98 [85]	unlabeled	HTTP requests (flash crowd)	Nondiverse	Non-CCE	Available with restricted access	Common Log	log-based
DARPA [86]	labeled	DOS, user to root (U2R), remote to local (R2L), probing attack	diverse	Non-CCE	Unavailable	tcpdump	traffic-based

Table 5. *Cont.*

Dataset	Class Label	Attack Type	Attack Scenario	Environment Type	Availability	Data Format	Representation (Source)
CSIC 2010 HTTP DATASET [87]	labeled	SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, HTTP (web application), etc.	diverse	Non-CCE	Available with no restricted access	packets	traffic-based
CIDDS-001 and CIDDS-002 [88]	labeled	DoS, HTTP WS, Brute Force, Port Scans. Transport Protocol (e.g., ICMP, TCP, or UDP)	Nondiverse	Non-CCE and CCE (virtual environment)	Available with no restricted access	flow-based	traffic-based
CICIDS2017 [89]	labeled	Dos, DDoS, HTTP, HTTPS, FTP, SSH, email protocols	diverse	Non-CCE	Available with no restricted access	flow-based	traffic-based
CICDDoS2019 [90]	labeled	DNS, TFTP, WebDDoS, etc.	diverse	Non-CCE	Unavailable	flow-based	traffic-based
CSE-CIC-IDS2018 [91]	labeled	Dos, DDoS, Brute Force (FTP, SSH), DDoS + Port Scan (UDP, TCP, HTTP/HTTPS), DoS attack, Infiltration (Nmap, portscan), SMTP, POP3, etc.	diverse	CCE	Available with no restricted access	flow-based	traffic-based
KDD cup 99 [92]	labeled	DoS, ICMP, TCP, UDP, HTTP	diverse	Non-CCE	Available with restricted access	flow-based	traffic-based
NSL-KDD [93]	labeled	DoS, ICMP, TCP, UDP, HTTP	diverse	Non-CCE	Available with no restricted access	flow-based	traffic-based
AIT Log Dataset [41]	labeled	Webmail (nmap scan, smtp-user-enum, webshell, nikto scan, etc.)	diverse	CCE	Available with no restricted access	combined log	log-based
Harvard Dataverse, V1 [94]	labeled	HTTP (application server)	diverse	CCE	Unavailable	combined log	log-based
UNB ISCX 2012 [14]	labeled	HTTP, SMTP, SSH, IMAP, POP3, FTP, etc.	diverse	Non-CCE	Available with restricted access	flow-based	traffic-based
Kyoto [95]	labeled	HTTP, HTTPs, FTP, SSH, mail traffic data, etc.	diverse	Non-CCE	Available	packets	traffic-based
ECML-KDD [96]	labeled	HTTP, Cross-Site Scripting, SQL, XPATgives our dataset	nondiverse	Non-CCE	restricted access	packets	traffic-based
Alkassasbeh et al. 2016 [30]	labeled	HTTP flood, SIDDOS, UDP flood, and Smurf	nondiverse	Non-CCE	restricted access	packets	traffic-based
Our dataset	labeled	HTTP (WS-HTTP flood DDoS attack)	diverse	CCE	Available with no restricted access	combined log	log-based

Table 5 shows that publicly accessible datasets are often generated in simulated or virtual environments rather than real CCEs. However, these environments cannot accurately imitate actual CCE behaviors, resulting in biased characteristics that do not have the features of CCEs and rendering them ineffective at detecting attacks. Additionally, the datasets collected in a CCE give our dataset the necessary features to improve detection.

A labeled dataset incorporates the evaluation of detection systems. However, most public dataset are not labeled, making them unusable for assessing these systems. Another issue is their representation. Most publicly accessible datasets were not collected from log files, making them unable to properly represent HTTP-GET flood attack behaviors. The proposed dataset is different from the existing dataset in the following ways:

- Real-World Implementation in CCEs: Unlike many datasets generated in simulated or virtual environments, the proposed dataset is implemented in a real CCE environment. This real-world implementation ensures that the dataset reflects the complexities and nuances present in actual CCE scenarios, providing a more accurate representation of the challenges faced by modern Intrusion Detection Systems.
- Focus on Specific Attack Type: The dataset is specifically tailored to focus on HTTP-GET flooding distributed denial-of-service (DDoS) attacks. While some existing datasets may cover a wide range of attack types, the proposed dataset's dedicated focus allows for in-depth analysis and specialized detection techniques for this prevalent and critical threat.
- CCE-Specific Features from Multiple Sources: The proposed dataset is enriched with CCE-specific features collected from three distinct sources: access logs, CCE logs, and VM resources. Our dataset of diverse data sources provides our data with unique advantage over existing datasets that lack CCE-specific features. Incorporating information from various sources within the CCE environment makes the dataset more comprehensive and representative of real-world network scenarios. This distinction is crucial because the availability of CCE-specific features allows IDSs to leverage context-aware information and tailor their detection mechanisms to the specific characteristics of CCEs. As a result, the proposed dataset empowers researchers to design IDSs that are better equipped to detect and mitigate threats in CCEs effectively.

Finally, we suggest the following mitigation plan to prevent HTTP-GET flooding attacks on CCEs:

- Integrating a trustworthy DDoS protection service that specializes in mitigating HTTP GET deluge attacks is crucial. This service can detect and block malicious traffic before it reaches the target system, ensuring it is accessible to legitimate users.
- Implementing rate-limiting and throttling mechanisms on the targeted system controls the number of GET requests from specified IP addresses or users within a given time. This mitigates the effects of deluge attacks by preventing a single source from saturating the system with excessive requests.
- It is essential to deploy a web application firewall (WAF) specifically configured to detect and thwart malicious HTTP GET inundation attempts. The WAF can analyze incoming traffic, recognize suspicious patterns, and implement predefined rules to differentiate between legitimate requests and attack traffic, thereby increasing the system's resistance to deluge attacks.

8. Conclusions and Future Work

The HTTP-GET message is vital for any WS to work correctly. However, it is vulnerable to DoS and DDoS attacks, and HTTP GET-based attacks are among the most dangerous and devastating. Many proposed detection systems have detected DDoS attacks based on HTTP-GET. But, more often than not, the evaluation and comparison of these systems rely on self-generated datasets, and most existing datasets represent traditional environments, which are ineffective for CCEs. Furthermore, nonqualified features from non-CCEs are used to describe these datasets, resulting in misclassification problems for classifiers.

Meanwhile, CCE-based datasets with nonsecurity objectives usually have limited attack scenarios, are unavailable for others to use, or contain unlabeled requests. As a result, they cannot be used to create, train, or/and test detection systems in CCEs for HTTP GET-based DDoS attack scenarios.

The proposed dataset fulfilled all the requirements for an adequate dataset since it was created from real-world requests on CCEs, which is crucial since the nature and environ-

ment of the CCE are different from that of non-CCEs. It also includes all possible scenarios of HTTP-GET DDoS attacks and properly and wholly labeled requests, represented by using representative features, and a balanced ratio of normal and attack requests. Furthermore, the employed representation and features were evaluated by using several classifiers (with their default settings) to achieve an acceptable and reliable detection accuracy and False Positive rates. Furthermore, the proposed datasets outperformed the existing datasets qualitatively (according to the criteria) in several metrics.

Although the proposed dataset is currently limited to IPv4-based HTTP-GET flood DDoS attacks on CCEs, future work will include other HTTP-based attacks, including HTTP-POST-based ones and IPv6-based attack requests. In addition, future work on the proposed dataset should keep updating and expanding it by using various tools to increase the number of attacks engaging in abnormal activity and the ratio of malicious requests. In addition, more clients (normal requests) will perform routine requests to avoid using balancing techniques in the future, such as SMOTE. These activities add more realistic dataset requests and are necessary for labeling and validating them.

Author Contributions: Writing—original draft preparation, Z.R.A. and M.A.; writing—review and editing, Z.R.A., M.A., S.D.A.R., B.A.A. and K.A.; project administration, Z.R.A.; resources, Z.R.A. and M.A.; supervision, Z.R.A. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the General Research Funding program, grant code (NU/RG/SERC/12/3).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data is accessible at [77].

Acknowledgments: The authors are thankful to the Deanship of Scientific Research at Najran University for funding this work under the General Research Funding program.

Conflicts of Interest: The authors declare that they have no conflict of interest to report regarding the present study.

References

1. Alashhab, Z.R.; Anbar, M.; Singh, M.M.; Hasbullah, I.H.; Jain, P.; Al-Amiedy, T.A. Distributed Denial of Service Attacks against Cloud Computing Environment: Survey, Issues, Challenges and Coherent Taxonomy. *Appl. Sci.* **2022**, *12*, 12441.
2. MDN, M. An Overview of HTTP—HTTP | MDN. Available online: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview (accessed on 30 May 2023).
3. Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berners-Lee, T. Hypertext Transfer Protocol–HTTP/1.1, RFC-editor, California, 1999. RFC2616. 2006. Available online: https://www.rfc-editor.org/info/rfc2616 (accessed on 30 May 2023).
4. Al-Ani, A.K.; Anbar, M.; Manickam, S.; Al-Ani, A. DAD-Match: Technique to Prevent DoS Attack on Duplicate Address Detection Process in IPv6 Link-local Network. *J. Commun.* **2018**, *13*, 317–324. [CrossRef]
5. Sree, T.R.; Bhanu, S. Detection of http flooding attacks in cloud using dynamic entropy method. *Arab. J. Sci. Eng.* **2018**, *43*, 6995–7014. [CrossRef]
6. Navarro, J.; Deruyver, A.; Parrend, P. A systematic survey on multi-step attack detection. *Comput. Secur.* **2018**, *76*, 214–249.
7. Al-Ani, A.K.; Anbar, M.; Manickam, S.; Al-Ani, A.; Leau, Y.B. Proposed DAD-match security technique based on hash function to secure duplicate address detection in IPv6 link-local network. In Proceedings of the 2017 International Conference on Information Technology, Singapore, 27–29 December 2017; pp. 175–179.
8. Kumar, R.; Lal, S.P.; Sharma, A. Detecting denial of service attacks in the cloud. In Proceedings of the 2016 IEEE 14th International Conference on Dependable, Autonomic and Secure Computing, 14th International Conference on Pervasive Intelligence and Computing, 2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Auckland, New Zealand, 8–12 August 2016; pp. 309–316.
9. Karim, A.M.; Güzel, M.S.; Tolun, M.R.; Kaya, H.; Çelebi, F.V. A new generalized deep learning framework combining sparse autoencoder and Taguchi method for novel data classification and processing. *Math. Probl. Eng.* **2018**, *2018*, 3145947.
10. Rawashdeh, A.; Alkasassbeh, M.; Al-Hawawreh, M. An anomaly-based approach for DDoS attack detection in cloud environment. *Int. J. Comput. Appl. Technol.* **2018**, *57*, 312–324.

11. Doriguzzi-Corin, R.; Millar, S.; Scott-Hayward, S.; Martinez-del Rincon, J.; Siracusa, D. LUCID: A practical, lightweight deep learning solution for DDoS attack detection. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 876–889. [CrossRef]
12. Bhardwaj, A.; Subrahmanyam, G.; Avasthi, V.; Sastry, H. Three tier network architecture to mitigate ddos attacks on hybrid cloud environments. In Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, Udaipur, India, 4–5 March 2016; pp. 1–7.
13. Alashhab, Z.R.; Anbar, M.; Singh, M.M.; Leau, Y.B.; Al-Sai, Z.A.; Alhayja'a, S.A. Impact of coronavirus pandemic crisis on technologies and cloud computing applications. *J. Electron. Sci. Technol.* **2021**, *19*, 100059.
14. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A.A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* **2012**, *31*, 357–374. [CrossRef]
15. Sommer, R.; Paxson, V. Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 305–316.
16. Bahashwan, A.A.; Anbar, M.; Hasbullah, I.H.; Alashhab, Z.R.; Bin-Salem, A. Flow-Based Approach to Detect Abnormal Behavior in Neighbor Discovery Protocol (NDP). *IEEE Access* **2021**, *9*, 45512–45526. [CrossRef]
17. Al Ashhab, Z.; Anbar, M.; Singh, M.; Alieyan, K.; Ghazaleh, W.A. Detection of HTTP flooding DDoS attack using Hadoop with MapReduce: A survey. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 1609–1620.
18. Ghazaleh, W.I.; Ahmad, W.A. A technical feasibility for adoption of clou d computing in King Abdulaziz University, Saudi Arabia. *Int. J. Sci. Res. (IJSR)* **2017**, *6*, 11.
19. John, J.; Norman, J. Major vulnerabilities and their prevention methods in cloud computing. In *Advances in Big Data and Cloud Computing*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 11–26.
20. Jain, R.K.; Kasana, D.R.; Jain, D.S. Efficient web log mining using doubly linked tree. *arXiv* **2009**, arXiv:0907.5433.
21. Apache. Apache Module mod_log_config—Apache HTTP Server Version 2.4. Available online: http://httpd.apache.org/docs/current/mod/mod_log_config.html (accessed on 30 May 2023).
22. Winter, P.; Hermann, E.; Zeilinger, M. Inductive intrusion detection in flow-based network data using one-class support vector machines. In Proceedings of the 2011 4th IFIP International Conference on New Technologies, Mobility and Security, Paris, France, 7–10 February 2011; pp. 1–5.
23. Sperotto, A.; Pras, A. Flow-based intrusion detection. In Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, Dublin, Ireland, 23–27 May 2011; pp. 958–963.
24. Patil, N.V.; Krishna, C.R.; Kumar, K.; Behal, S. E-Had: A distributed and collaborative detection framework for early detection of DDoS attacks. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 1373–1387.
25. Alsirhani, A.; Sampalli, S.; Bodorik, P. Ddos detection system: Utilizing gradient boosting algorithm and apache spark. In Proceedings of the 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), Quebec, QC, Canada, 13–16 May 2018; pp. 1–6.
26. Muthuperumal Periyaperumal, R.; Ramasamy, G.; Azees, M.; Sakthi, U. FACVSPO: Fractional anti corona virus student psychology optimization enabled deep residual network and hybrid correlative feature selection for distributed denial-of-service attack detection in cloud using spark architecture. *Int. J. Adapt. Control Signal Process.* **2022**, *36*, 1647–1669.
27. Velliangiri, S.; Karthikeyan, P.; Vinoth Kumar, V. Detection of distributed denial of service attack in cloud computing using the optimization-based deep networks. *J. Exp. Theor. Artif. Intell.* **2021**, *33*, 405–424. [CrossRef]
28. Hsieh, C.J.; Chan, T.Y. Detection DDoS attacks based on neural-network using Apache Spark. In Proceedings of the 2016 International Conference on Applied System Innovation (ICASI), Okinawa, Japan, 26–30 May 2016; pp. 1–4.
29. Kheir, N. Analyzing http user agent anomalies for malware detection. In *Data Privacy Management and Autonomous Spontaneous Security*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 187–200.
30. Alkasassbeh, M.; Al-Naymat, G.; Hassanat, A.B.; Almseidin, M. Detecting distributed denial of service attacks using data mining techniques. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 436–445. [CrossRef]
31. Morgan, J.; Zincir-Heywood, A.N.; Jacobs, J.T. A benchmarking study on stream network traffic analysis using active learning. In *Recent Advances in Computational Intelligence in Defense and Security*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 249–273.
32. Čeponis, D.; Goranin, N. Towards a robust method of dataset generation of malicious activity for anomaly-based HIDS training and presentation of AWSCTD dataset. *Balt. J. Mod. Comput.* **2018**, *6*, 217–234.
33. Dhanapal, A.; Nithyanandam, P. An OpenStack based cloud testbed framework for evaluating HTTP flooding attacks. *Wirel. Netw.* **2021**, *27*, 5491–5501. [CrossRef]
34. Muraleedharan, N.; Janet, B. An HTTP DDoS Detection Model Using Machine Learning Techniques for the Cloud Environment. In *Advances in Computing and Network Communications*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 685–698.
35. Rajapraveen, K.; Pasumarty, R. A Machine Learning Approach for DDoS Prevention System in Cloud Computing Environment. In Proceedings of the 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India, 16–18 December 2021; pp. 1–6.
36. Saini, B.; Somani, G. Index page based EDoS attacks in infrastructure cloud. In Proceedings of the International Conference on Security in Computer Networks and Distributed Systems, Trivandrum, India, 13–14 March 2014; pp. 382–395.
37. Kushwah, G.S.; Ali, S.T. Detecting DDoS attacks in cloud computing using ANN and black hole optimization. In Proceedings of the 2017 2nd International Conference on Telecommunication and Networks (TEL-NET), Noida, India, 10–11 August 2017; pp. 1–5.

38. Mugunthan, S. Soft computing based autonomous low rate DDOS attack detection and security for cloud computing. *J. Soft Comput. Paradig. (JSCP)* **2019**, *1*, 80–90.
39. Velliangiri, S.; Premalatha, J. Intrusion detection of distributed denial of service attack in cloud. *Clust. Comput.* **2019**, *22*, 10615–10623.
40. Chovanec, M.; Hasin, M.; Havrilla, M.; Chovancová, E. Detection of HTTP DDoS Attacks Using NFStream and TensorFlow. *Appl. Sci.* **2023**, *13*, 6671.
41. Landauer, M.; Skopik, F.; Wurzenberger, M.; Hotwagner, W.; Rauber, A. Have it your way: Generating customized log datasets with a model-driven simulation testbed. *IEEE Trans. Reliab.* **2020**, *70*, 402–415.
42. Shahanaz Begum, I.; Geetharamani, G. DDoS attack detection and prevention in private cloud environment. *Int. J. Innov. Eng. Technol. (IJIET)* **2016**, *7*, 527–531.
43. Dhanapal, A.; Nithyanandam, P. The Slow HTTP DDOS Attacks: Detection, Mitigation and Prevention in the Cloud Environment. *Scalable Comput. Pract. Exp.* **2019**, *20*, 669–685. [CrossRef]
44. Wani, A.R.; Rana, Q.; Saxena, U.; Pandey, N. Analysis and detection of DDoS attacks on cloud computing environment using machine learning techniques. In Proceedings of the 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 4–6 February 2019; pp. 870–875.
45. Choi, J.; Choi, C.; Ko, B.; Choi, D.; Kim, P. Detecting Web based DDoS Attack using MapReduce operations in Cloud Computing Environment. *J. Internet Serv. Inf. Secur.* **2013**, *3*, 28–37.
46. Dhanapal, A.; Nithyanandam, P. The slow HTTP distributed denial of service attack detection in cloud. *Scalable Comput. Pract. Exp.* **2019**, *20*, 285–298.
47. Iyengar, N.; Banerjee, A.; Ganapathy, G. A fuzzy logic based defense mechanism against distributed denial of service attack in cloud computing environment. *Int. J. Commun. Netw. Inf. Secur.* **2014**, *6*, 233.
48. Karnwal, T.; Sivakumar, T.; Aghila, G. A comber approach to protect cloud computing against XML DDoS and HTTP DDoS attack. In Proceedings of the 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science, Bhopal, India, 1–2 March 2012; pp. 1–5.
49. Chonka, A.; Xiang, Y.; Zhou, W.; Bonti, A. Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks. *J. Netw. Comput. Appl.* **2011**, *34*, 1097–1107.
50. Deakin University. StuPot Project, "HXDoS Dataset", Deakin University. Available online: http://www.deakin.edu.au/~chonka/ (accessed on 30 May 2023).
51. Chatzoglou, E.; Kouliaridis, V.; Kambourakis, G.; Karopoulos, G.; Gritzalis, S. A hands-on gaze on HTTP/3 security through the lens of HTTP/2 and a public dataset. *Comput. Secur.* **2023**, *125*, 103051. [CrossRef]
52. Aborujilah, A.; Musa, S. Cloud-based DDoS HTTP attack detection using covariance matrix approach. *J. Comput. Netw. Commun.* **2017**, *2017*, 7674594. [CrossRef]
53. Yang, L.; Zhang, T.; Song, J.; Wang, J.S.; Chen, P. Defense of DDoS attack for cloud computing. In Proceedings of the 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), Zhangjiajie, China, 25–27 May 2012; Volume 2, pp. 626–629.
54. Choi, J.; Choi, C.; Ko, B.; Kim, P. A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment. *Soft Comput.* **2014**, *18*, 1697–1703.
55. Garg, S.; Kaur, K.; Kumar, N.; Batra, S.; Obaidat, M.S. HyClass: Hybrid classification model for anomaly detection in cloud environment. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–7.
56. Bhushan, K.; Gupta, B.B. A novel approach to defend multimedia flash crowd in cloud environment. *Multimed. Tools Appl.* **2018**, *77*, 4609–4639. [CrossRef]
57. Raja Sree, T.; Mary Saira Bhanu, S. Detection of HTTP flooding attacks in cloud using fuzzy bat clustering. *Neural Comput. Appl.* **2020**, *32*, 9603–9619. [CrossRef]
58. Kushwah, G.S.; Ranga, V. Distributed denial of service attack detection in cloud computing using hybrid extreme learning machine. *Turk. J. Electr. Eng. Comput. Sci.* **2021**, *29*, 1852–1870. [CrossRef]
59. Kushwah, G.S.; Ranga, V. Voting extreme learning machine based distributed denial of service attack detection in cloud computing. *J. Inf. Secur. Appl.* **2020**, *53*, 102532. [CrossRef]
60. Al-Amiedy, T.A.; Anbar, M.; Belaton, B.; Kabla, A.H.H.; Hasbullah, I.H.; Alashhab, Z.R. A systematic literature review on machine and deep learning approaches for detecting attacks in RPL-based 6LoWPAN of internet of things. *Sensors* **2022**, *22*, 3400. [CrossRef]
61. Sperotto, A.; Schaffrath, G.; Sadre, R.; Morariu, C.; Pras, A.; Stiller, B. An overview of IP flow-based intrusion detection. *IEEE Commun. Surv. Tutor.* **2010**, *12*, 343–356. [CrossRef]
62. Omolara, A.E.; Jantan, A.; Abiodun, O.I.; Singh, M.M.; Anbar, M.; Kemi, D. State-of-the-art in big data application techniques to financial crime: A survey. *Int. J. Comput. Sci. Netw. Secur.* **2018**, *18*, 6–16.
63. Birjali, M.; Beni-Hssane, A.; Erritali, M. Analyzing social media through big data using infosphere biginsights and apache flume. *Procedia Comput. Sci.* **2017**, *113*, 280–285. [CrossRef]
64. Gutierrez, J.N.P.; Lee, K. An Attack-based Filtering Scheme for Slow Rate Denial-of-Service Attack Detection in Cloud Environment. *J. Multimed. Inf. Syst.* **2020**, *7*, 125–136. [CrossRef]

65. Joshi, A.; Joshi, K.; Krishnapuram, R. *On Mining Web Access Logs*; UMBC Computer Science and Electrical Engineering Department: Baltimore, MD, USA, 1999.
66. Katrawi, A.H.; Abdullah, R.; Anbar, M.; Abasi, A.K. Earlier stage for straggler detection and handling using combined CPU test and LATE methodology. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 4910. [CrossRef]
67. Baldi, M.; Baralis, E.; Risso, F. Data mining techniques for effective flow-based analysis of multi-gigabit network traffic. In Proceedings of the IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCom 2004), Split, Croatia, 10–13 October 2004; pp. 330–334.
68. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA data mining software: An update. *ACM SIGKDD Explor. Newsl.* **2009**, *11*, 10–18. [CrossRef]
69. Webb, G.I.; Keogh, E.; Miikkulainen, R. Naïve Bayes. *Encycl. Mach. Learn.* **2010**, *15*, 713–714.
70. Urooj, U.; Al-rimy, B.A.S.; Zainal, A.; Ghaleb, F.A.; Rassam, M.A. Ransomware detection using the dynamic analysis and machine learning: A survey and research directions. *Appl. Sci.* **2021**, *12*, 172. [CrossRef]
71. Cengiz, A.; Budak, M.; Yağmur, N.; Balçik, F. Comparison between random forest and support vector machine algorithms for LULC classification. *Int. J. Eng. Geosci.* **2023**, *8*, 1–10.
72. Peterson, L.E. K-nearest neighbor. *Scholarpedia* **2009**, *4*, 1883.
73. Lawrence, J. *Introduction to Neural Networks*; California Scientific Software: Nevada City, CA, USA, 1993.
74. Herrera-Silva, J.A.; Hernández-Álvarez, M. Dynamic feature dataset for ransomware detection using machine learning algorithms. *Sensors* **2023**, *23*, 1053. [CrossRef]
75. Choudhury, S.; Bhowal, A. Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection. In Proceedings of the 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Avadi, India, 6–8 May 2015; pp. 89–95.
76. Berrar, D. Cross-Validation. *Encycl. Bioinform. Comput. Biol. ABC Bioinform.* **2019**, *1–3*, 542–545. [CrossRef]
77. Alashhab, Z.; Anbar, M. CCE-DataSet. Available online: https://sites.google.com/view/cce-dataset/home (accessed on 30 May 2023).
78. University of Saskatchewan. Saskatchewan-HTTP—Seven Months of HTTP Logs from the University of Saskatchewan WWW Server. Available online: http://ita.ee.lbl.gov/html/contrib/Sask-HTTP.html (accessed on 30 May 2023).
79. NASA Kennedy Space Center. NASA-HTTP—Two Months of HTTP Logs from the KSC-NASA WWW Server. Available online: http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html (accessed on 30 May 2023).
80. Deakin University. Laura Bottomley, ClarkNet-HTTP. Available online: http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html (accessed on 30 May 2023).
81. University of Calgary. Calgary-HTTP—A Year of HTTP Logs from the University of Calgary CS WWW Server. Available online: http://ita.ee.lbl.gov/html/contrib/Calgary-HTTP.html (accessed on 30 May 2023).
82. San Diego Supercomputer Center. SDSC-HTTP—A Day of HTTP Logs from the SDSC WWW Server. Available online: http://ita.ee.lbl.gov/html/contrib/SDSC-HTTP.html (accessed on 30 May 2023).
83. Research Triangle Park. EPA-HTTP—A Day of HTTP Logs from the EPA WWW Server. Available online: http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html (accessed on 30 May 2023).
84. USMA. Cyber Research Center—Data Sets | United States Military Academy West Point. Available online: https://www.westpoint.edu/centers-and-research/cyber-research-center/data-sets (accessed on 30 May 2023).
85. Arlitt, M.; Jin, T. A workload characterization study of the 1998 world cup web site. *IEEE Netw.* **2000**, *14*, 30–37. [CrossRef]
86. Mit. DARPA. Available online: https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusiondetection-scenario-specic-datasets (accessed on 30 May 2023).
87. Fing. Projects · GSI/Web-Application-Attacks-Datasets · GitLab. Available online: https://gitlab.fing.edu.uy/gsi/web-application-attacks-datasets (accessed on 30 May 2023).
88. Ring, M.; Wunderlich, S.; Grüdl, D.; Landes, D.; Hotho, A. Creation of flow-based data sets for intrusion detection. *J. Inf. Warf.* **2017**, *16*, 41–54.
89. Unb. IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 30 May 2023).
90. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 1–3 October 2019; pp. 1–8.
91. Unb. IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Available online: https://www.unb.ca/cic/datasets/ids-2018.html (accessed on 30 May 2023).
92. University of California, Irvine. KDD Cup 1999 Data. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 30 May 2023).
93. Canadian Institute for Cybersecurity, University of New Brunswick. NSL-KDD Datasets. Available online: https://www.unb.ca/cic/datasets/nsl.html (accessed on 30 May 2023).
94. Zaker, F. Online Shopping Store—Web Server Logs. Available online: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/3QBYB5 (accessed on 30 May 2023). [CrossRef]

95. Song, J.; Takakura, H.; Okabe, Y. Description of Kyoto University Benchmark Data. 2006. Available online: http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf (accessed on 15 March 2023).
96. Raıssi, C.; Brissaud, J.; Dray, G.; Poncelet, P.; Roche, M.; Teisseire, M. Web analyzing traffic challenge: Description and results. In Proceedings of the ECML/PKDD, Warsaw, Poland, 17–21 September 2007; pp. 47–52.