Computers & Security 138 (2024) 103663

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

DDoS family: A novel perspective for massive types of DDoS attacks

Ziming Zhao [a,c], Zhaoxuan Li [b,d], Zhihao Zhou [a], Jiongchi Yu [c], Zhuoxue Song [a], Xiaofei Xie [c], Fan Zhang [a,e,f,g,h,*], Rui Zhang [b]

[a] College of Computer Science and Technology, Zhejiang University, Hangzhou, 310027, China
[b] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, China
[c] School of Computing and Information Systems, Singapore Management University, Singapore, 188065, Singapore
[d] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, 100049, China
[e] ZJU-Hangzhou Global Scientific and Technological Innovation Center, Hangzhou, 311200, China
[f] Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province, Hangzhou, 310027, China
[g] Jiaxing Research Institute, Zhejiang University, Jiaxing, 314000, China
[h] Zhengzhou Xinda Institute of Advanced Technology, Zhengzhou, 450000, China

ARTICLE INFO

Keywords:
DDoS attack family
Community detection
Defense strategy
Backward packet statistics
Traffic fingerprint construction

ABSTRACT

Distributed Denial of Service (DDoS) defense is a profound research problem. In recent years, adversaries tend to complicate their attack strategies by crafting vast DDoS variants. On the one hand, this trend exacerbates both extremes of classification granularity (*i.e.*, binary and attack level) in existing machine learning methods. On the other hand, massive attack categories make the filter rule table bulky, as well as cause problems of slow reaction presented in the recent state-of-the-art DDoS mitigation system. Therefore, we propose the concept of a DDoS family to reconcile/cope with these issues. The specific technical roadmap includes traffic pattern characterization, attack fingerprint production, and cross-executed family partition by community detection. Through extensive evaluations, we demonstrate the benefits of the proposal in terms of portraying similarities, guiding model classification/unknown attack detection, optimizing defense strategies, and speeding filtering reactions. For instance, our results show that using only one rule can defend 15 types of attacks due to their homogeneous behavioral representation. Particularly, we find the interesting observation that counting the backward packet is more efficient and robust against some attacks (*e.g.*, Tor's Hammer Attack), which is very different from previous solutions.

1. Introduction

Distributed Denial of Service (DDoS) continues to be a prevalent attack on Internet today, and their prevalence escalates along with the growing number of vulnerable devices connected online. Over the past few years, DDoS detection and mitigation have garnered significant attention from the academic and industrial communities (Fayaz et al., 2015; Liu et al., 2021; Rossow, 2014). For instance, Fayaz et al. (2015) designed a series of defense policies for common DDoS attacks (*e.g.,* SYN Flood and UDP Flood). Rossow (2014) proposed proactive and reactive countermeasures against Distributed Reflective Denial of Service (DRDoS), such as preventing IP address spoofing and hardening protocol. Besides, in industrial scenarios, most devices (*e.g.,* middleboxes (Mahimkar et al., 2007)) are deployed in the traffic scrubbing centers to mitigate DDoS "as-a-service" (Zhang et al., 2020).

Nowadays, adversaries tend to craft diverse DDoS variants to complicate their attacks. According to reports from many vendors, there are at least hundreds of active DDoS attacks/variants (MAZEBOLT, 2016). Therefore, we summarize the following three main challenges suffered by security practitioners in addressing massive types of DDoS attacks.

(i) *Two extremes of classification models.* The existing machine learning (ML) models for DDoS identification are mainly either binary classification (Bartos et al., 2016; Caselli et al., 2016; Cho and Shin, 2016; Fogla et al., 2006; Fu et al., 2023, 2021; Mirsky et al., 2018; Wang et al., 2020b; Xie et al., 2022) or attack-level multi-classification (Barradas et al., 2021; Caselli et al., 2016; van Ede et al., 2020; Holland et al., 2021; Lin et al., 2022; Song et al., 2023; Zhao et al., 2023a,b). The former only provides binary results with "benign" or "attack" so that making it challenging for network operators to explicitly adopt proper

* Corresponding author.
E-mail addresses: zhaoziming@zju.edu.cn (Z. Zhao), lizhaoxuan@iie.ac.cn (Z. Li), zhouzhihao@zju.edu.cn (Z. Zhou), jcyu.2022@phdcs.smu.edu.sg (J. Yu), songzhuoxue@zju.edu.cn (Z. Song), xfxie@smu.edu.sg (X. Xie), fanzhang@zju.edu.cn (F. Zhang), zhangrui@iie.ac.cn (R. Zhang).

https://doi.org/10.1016/j.cose.2023.103663
Received 3 July 2023; Received in revised form 6 November 2023; Accepted 14 December 2023
Available online 19 December 2023

countermeasures for mitigating DDoS attacks (also known as semantic gap (Liang et al., 2021; Sommer and Paxson, 2010)). While the latter[1] will present significant accuracy loss when classes increase, as well as labeling each-type attack is labor-intensive for domain-specific experts.

(ii) *Defense strategies tend to be bulky.* The recent arts demonstrate powerful superiority by combining DDoS defenses policies with emerging data plane primitives, *e.g.,* Data Plane Development Kit (DPDK) (Gong et al., 2019), Software Defined Network (SDN) (Mirsky et al., 2018; Sisodia et al., 2020), P4 Programmable Switches (Liu et al., 2021; Zhang et al., 2020). However, these approaches require the corresponding firewall rule for each new emerging DDoS attack, which will inevitably result in the rule explosion issue. In other words, the rule library will be bulky which complicates configuration/maintenance tasks and introduces additional latency.

(iii) *Slow reaction in traffic scrubbing.* Strong adversaries will vary their attack strategy so that the volume of per-type attack is below the filtering threshold. This hybrid exploit strategy could alternately launch different types of DDoS attacks to avoid triggering each filter policy as possible (Wu et al., 2014). As a result, the defense system tends to be not sensitive and takes effect slowly on traffic scrubbing. For example, it has been observed that state-of-the-art (SOTA) defense system (Zhang et al., 2020) still requires several seconds to restore normal throughput following a DDoS attack, we call this phenomenon the *slow reaction* problem. This leads to collateral damage wherein the bandwidth for legitimate traffic will be preempted, at times dropping to ∼0 Gbps under the worst circumstances.

Given the challenges faced by security practitioners in dealing with the massive types of DDoS attacks, we advocate a paradigm shift towards categorizing these DDoS into attack families to reconcile this problem. Specifically, we intend to characterize existing DDoS attacks based on their traffic pattern and then regard the similar categories as a family. For DDoS attacks in the same family, they have homogeneous behavioral patterns and we could develop the corresponding defense strategies based on the same template. Also, the built families can guide the multi-classification task of ML models, thereby achieving a suitable trade-off between accuracy and fine-grained labels. Note that our proposed DDoS families differ from the previous work of botnet families (*e.g.,* Mirai (Antonakakis et al., 2017), Hajime (Herwig et al., 2019)) analysis that aims to depict control manners and scheduling patterns for the bots (Wang et al., 2020a).

In this paper, we design a roadmap to construct DDoS families to cater to the above intention. As a high-level idea, we portray the fingerprint for per-type DDoS, calculate the similarity between different categories to generate the relation graph, and ultimately identify the family division as a community detection problem (details can be found in § 3.2). Specifically, we extract the comprehensive feature sets involving categorical and numerical data to manufacture network traffic portraits. By computing the probability distribution histogram and kernel density estimation, we obtain the per-feature fitted curve, and map it to a high-dimensional discrete vector representation through a differentiation process. Moreover, we present a cross-executed solution to mitigate the effects of sampling error that alternately build membership from the unweighted multiplex graphs and guide the weighted graph partition.

In summary, this paper makes three key contributions.

- We meticulously scrutinize the challenges posed to security practitioners by the multitude of DDoS attack types. To this end, we propose the concept of the DDoS attack family to facilitate addressing the above issues.
- We design a roadmap for DDoS family construction, encompassing the stages of traffic pattern characterization, attack fingerprint production, cross-executed community detection including membership generation of the unweighted multiplex graphs and weighted graph division.
- We collect 89 types of real-world DDoS traffic from security vendors and the local Internet Service Provider (ISP) to conduct the attack family partition experiments, then use 18 types of new attacks to develop family extensibility research. The empirical evaluations demonstrate the effectiveness of the family division in terms of attack behavior analysis, ML classification guidelines, defense strategy optimization, and filtering reaction acceleration. We release the code, and the DDoS dataset after removing the private information is available on our repository.[2]

This paper is organized as follows, § 2 investigates background and related work on network traffic detection, DDoS attack defense, and community detection. § 3 elaborates on the motivation for the application scenarios and methodology. § 4 presents the roadmap of family division construction, including the feature extraction, fingerprint production, and family division. § 5 produces a series of evaluations in terms of entire-spectrum family construction, subset attributes analysis, model classification guidelines, defense strategy optimization, and traffic scrubbing evaluation. § 6 provides deep insights with respect to stability, robustness, and extensibility. In § 7, we in-depth discuss regarding involving customized granularity, real-world deployment, limitations, and future works. Finally, we conclude this paper in § 8.

2. Background and related work

In this section, we overview the existing techniques and research directions in the current DDoS detection and mitigation landscape. Meanwhile, we introduce the background and applications of community detection.

2.1. Network traffic detection

Traffic detection aims to detect various intrusion attacks by analyzing the characteristics of packets or flows. A series of models have been proposed that can be roughly divided into binary classification and multi-classification approaches. (i) *Binary classification models.* Whisper (Fu et al., 2021) utilizes sequential information based on the frequency domain features to detect malicious traffic. Kitsune (Mirsky et al., 2018) discovers abnormal behavior by using AutoEncoder to examine each packet. Other arts (Tang et al., 2020; Xu et al., 2020) portray the patterns of benign and malicious traffic to identify the attacks and keep the business stable. These binary classification models only alert "attack" but lack more details about attack types that are beneficial for defense strategies.

(ii) *Multi-classification models.* Mousika (Xie et al., 2022) introduces the decision tree that is translated from the deep neural network with knowledge distillation. FlowLens (Barradas et al., 2021) devises a compact representation of packet length and adjusts the granularity of the flow's frequency distribution intervals to classification with ML models. FS-Net (Liu et al., 2019) uses the bi-GRU model to learn the sequence features and classifies common applications. Other works (Ciucu et al., 2014; Korczynski and Duda, 2014; Saha et al., 2019; Shen et al., 2017) employ the methods such as clustering, Markov, and recurrent neural networks (RNN) to identify the multiple attacks based on the statistical and sequential features. Deep learning indeed brings more potential for traffic characterization and identification, yet the massive categories inevitably cause troubles in terms of model accuracy and ground-truth label quality. Our DDoS family is promising to reconcile the two extremes in ML classification.

[1] Performing multi-classification ML tasks could facilitate fine-grained analysis for defense strategies customization, benefit forensic analysis, profile homologous attack organizations, *etc.*

[2] https://github.com/Secbrain/DDoS_Family.

2.2. DDoS attack defense

DDoS mitigation is a profound research problem. The community has developed many approaches, which can be broadly categorized into filtering-based approaches (Argyraki et al., 2005; Ioannidis and Bellovin, 2002; Liu et al., 2008; Mahajan et al., 2002; Savage et al., 2000; Song and Perrig, 2001), capability-based approaches (Liu et al., 2010, 2016; Yaar et al., 2004; Yang et al., 2005), overlay-based systems (Andersen, 2003; Dixon et al., 2008; Keromytis et al., 2002), systems based on future Internet architectures (Andersen et al., 2008; Naylor et al., 2014; Zhang et al., 2011), and other variants (Fayaz et al., 2015; Gilad et al., 2016; Walfish et al., 2006). It is not difficult to find that most of the research hotspots focus on are in bandwidth capability, dataplane primitives, *etc.* We intend to pioneer a new direction in the current DDoS landscape, *i.e.,* examining attacks at a family level to discover deep insights and optimize defense designs.

2.3. Community detection

Community detection, as the method to understand the structure of large and complex networks, has found use in metabolic networks (Guimera and Nunes Amaral, 2005), mobile phone networks (Ahn et al., 2010; Li et al., 2014), airline transportation networks (Lambiotte et al., 2019), and social networks (Amin et al., 2017; Tong et al., 2016). Recent works use community detection tasks to detect similar groups based on the network communication graph (Baldesi et al., 2018; Ruehrup et al., 2013). Considering the distinct advantages of the community detection algorithm in discovering the connection relationship, we intend to use it to search the DDoS family partitions. Subsequently, we will clarify the details of this specific motivation.

3. Motivation

3.1. What benefits DDoS family brings?

In addressing the existing challenges, we summarize the advantages of our DDoS family involving the following aspects. (i) Providing deep insights into attack similarity (§ 5.1 and § 5.2). Through characterizing the attack traffic to mine their similarities and further give novel insights into attack mechanisms. For instance, we find that counting backward packets could be a more robust scheme, which diverges from conventional methods. (ii) Guiding the model classification task (§ 5.3). Conducting the ML model classification at the family level to reconcile the two extremes between binary and attack-level. Therefore, it can facilitate the trade-off between detection granularity and accuracy. (iii) Optimizing defense strategies (§ 5.4). On the one hand, combining defense strategies for similar attacks to optimize the rule library to be lightweight. On the other hand, exploring new perspectives (*e.g.,* backward packet statistics) can significantly enhance the robustness of defense mechanisms. (iv) Speeding up the reaction of filters (§ 5.5). Based on the above-optimized rules, aggregating the thresholds thereby improves the sensitivity for compound attacks and reduces reaction time.

3.2. Why not use the existing clustering?

Readers may be concerned about why not use the unsupervised clustering algorithms to produce the DDoS family, we elucidate this problem in two aspects.

(i) *Optimization objective.* The primary objective of clustering is to differentiate between individual instances (*i.e.,* focus on the node of a graph), while community detection seeks to partition relationships within the network (*i.e.,* focus on the edge of a graph). For example, K-means (Hartigan and Wong, 1979), as a distance-based method, tends to identify divisions where samples in each cluster are tightly concentrated. The sum of the distances of all samples from their cluster centers is called the distortion cost function, and minimizing its value is the optimization objective of K-means. Another typical representative is the density-based approach DBSCAN (Ester et al., 1996), which tends to split samples into several sets that per-group density is large enough. Different from these clustering methods, the most common metric of community detection is modularity (Traag et al., 2019). It refers to maximizing the difference between the actual number of edges in a community and the expected number of such edges. In other words, it focuses on the graph's edges to partition the tightly connected subgraphs. Therefore, the optimization objective of community detection aligns more closely with our intended approach.

(ii) *Search algorithm.* Unsupervised clustering algorithms come with certain limitations. K-means necessitates the predefined number of input clusters, while DBSCAN struggles with uneven density distributions. A more crucial problem is that they would focus more on the global metric while lacking some fine-grained observation for relationships between node members. For instance, the clustering algorithms can produce a set of samples that meet the pre-defined threshold (distance or density). However, the community detection might verify the connection within partitions, *e.g.,* further optimizing sub-partitions if there exists significant isolation (more details in § 4.4). Meanwhile, compared to clustering algorithms, community detection produces more stable partition results even with different hyperparameter settings (as evaluated in § 6.1). Overall, we intend to identify the DDoS family division as a community detection problem,[3] which is more suitable than clustering given the above considerations. Furthermore, we also conduct a series of evaluations for these methods in § 5.1.

4. Roadmap for family construction

4.1. Overview

Our roadmap for DDoS attack families construction is shown in Fig. 1. First, *n*-dimensional features are extracted from packet-level to session-level involving the categorical and numerical data. By sliding windows, we study the entire session as well as its several snippets. Next, our fingerprint production for each type of DDoS refers to calculating the probability distribution histogram, smoothing by the kernel density estimation, and further generating the fingerprint matrix with a differential process. Subsequently, we consider the family division as a community detection task and design a cross-executed process to alleviate the influence of random error from sampling. Finally, the DDoS families would advance the ML model and optimize the defense deployment. We will elaborate on the design details following.

4.2. Feature extraction

The meta fields are aggregating from per packet to the traffic session with the 5-tuple index, *i.e.,* $\{Source\ IP, Source\ Port, Destination\ IP, Destination\ Port, Protocol\}$. It contains two kinds of data, categorical attributes and real-valued attributes (numerical). Table 1 shows the extracted features for each bi-directional flow. Among them, the categorical-type data are from layer 2∼7 involving 14 fields such as $TLS.handshake$. These features represent some mapping relations and have no numerical meaning, so we encode them with one-hot. For the numerical-type data, we process per session as 6 groups, including the entire session and 5 snippets with sliding windows $\{[0, 0.1s), [0.1s, 0.5s), [0.5s, 1.5s), [1.5s, 10s), [10s, \infty)\}$. In each group, the feature computations are executed from the forward, backward,

[3] Therefore, we need to convert the relationship between various types of DDoS attacks into a graph structure representation as the input of the community detection algorithm. In addition, graph-structured data also has the advantages of rich semantic information, prone to visualization, and conduces exploration of connection patterns.

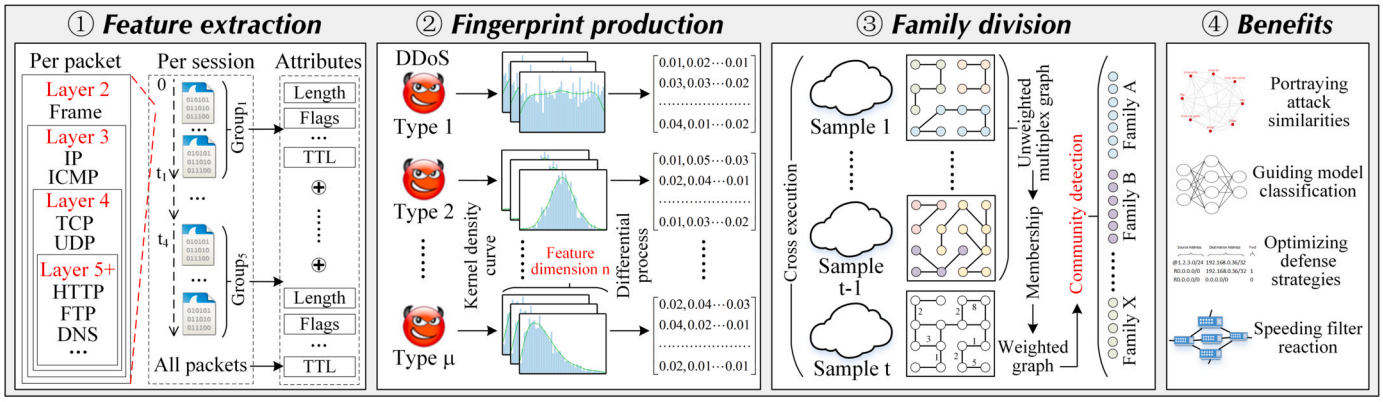


Fig. 1. The roadmap of proposed DDoS attack family construction.

Table 1
The feature set. "TTL" denotes "time to live".

Categorical			
Frame	TCP	UDP	ICMP
frame.protocols	tcp.srcport, tcp.dstport	udp.srcport, udp.dstport	icmp.type, icmp.code
DNS	HTTP	TLS	
dns.flags	request.method, response.code	record.version, record.content_type, handshake, handshake.type	

Numerical			
Time window	Direction		
	Forward	Backward	Bi-dir
whole, [0, 0.1), [0.1, 0.5), [0.5, 1.5), [1.5, 10), [10,)	Sum		
	Frame	IP	TCP
	packet_num, duration	flags_df, flags_mf	ACK, URG, PUSH, RESET, SYN, FIN
	Statistic		
	Max	Min	Mean Std
	total_length, time_delta, ip.frag_offset, ip.ttl, tcp.window_size, tcp.window_size_scalefactor, tcp.window_size_value, dns.count.queries		

and bi-directional, respectively. Particularly, the duration, number of packets, and a series of flags (*e.g.,* $IP.flags\_df$ and $TCP.ACK$) will be counted as the respective sums, while other fields (*e.g.,* packet length and time delta) will be calculated the statistical features (*i.e.,* Max, Min, $Mean$, and Std). Overall, it generates 770-dimensional (*i.e.,* $n = 770$) features to characterize the traffic in terms of temporal, volumetric, and header-field distributions.

4.3. Fingerprint production

To produce the fingerprint of per-type DDoS, we first calculate the probability distribution histogram, then smooth by the kernel density estimation, and finally discretize with a differential process. Before performing the calculation, we unify the value range for sample set X by Max-Min normalization $X = \frac{x - \min(X)}{\max(X) - min(X)}$, where $x \in X$. For each type of DDoS attacks $\mathcal{A}_i$ ($i \in [1, \mu]$), the fingerprint production is solely based on its instances X_i.

Probability Distribution Histogram Calculation. The first step is to calculate the probability distribution histogram for each dimension feature. Considering one-dimension feature vector $z = (z_1, \cdots, z_m) \in \mathbb{R}^m$ from m instances $\{x_1, \cdots, x_m\}$ of same attack, the histogram generation function ϕ references attribute the value to b bins $\theta = (\theta_1, \cdots, \theta_b) \in \mathbb{R}^b$ such as

$$\phi(z; \theta_i) = \frac{1}{m} \sum_{j=1}^{m} [\![z_j \in \theta_i]\!] \tag{1}$$

is the value of the i-th bin corresponding to a portion of components of z falling to the interval θ_i, where $[\![\cdot]\!] = 1$ if the condition $(\cdot)$ is satisfied and 0 otherwise.

Kernel Density Estimation. Next, we would smooth the histogram bins by the kernel density estimation. The function for the k-*th* feature values is as follows

$$\begin{aligned} \hat{f}^k(x) &= \frac{1}{2h} \frac{1}{m} \sum_{i=1}^{m} [\![x_i \in [x-h, x+h)]\!] \\ &= \frac{1}{mh} \sum_{i=1}^{m} \frac{1}{2} \left[\!\!\left[\frac{|x - x_i|}{h} \leq 1 \right]\!\!\right] \end{aligned} \tag{2}$$

where the aggregation offset $h = 1/2b$ and $\int_0^1 \hat{f}^k(x) = 1$. The standard kernel density estimate can further be expressed as

$$\hat{f}^k(x) = \frac{1}{mh} \sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right) \tag{3}$$

where $K(\cdot)$ denotes the kernel function and we set it to Eq. (4).

$$K(x) = \begin{cases} 1/2 & \text{if } |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Therefore, features of all dimensions will be transformed into corresponding distribution curves, *i.e.,* $\{\hat{f}^1(x), \cdots, \hat{f}^n(x)\}$.

Differential Process. Finally, we perform a differentiation process to discretely sample and obtain a vector representation of the curve. To generate s-dimensional vectors, the normalization space will be evenly divided into $s-1$ parts, *i.e.,* the sampling points are $\hat{x} = (\hat{x}_1, \cdots, \hat{x}_s) = (0, \frac{1}{s-1}, \cdots, \frac{s-2}{s-1}, 1)$. Thus, the fingerprint $\mathcal{F}_i$ of attack category $\mathcal{A}_i$ can be represented as a $n \times s$ matrix

$$\mathcal{F}_i = \begin{pmatrix} \hat{f}^1(\hat{x}) \\ \vdots \\ \hat{f}^n(\hat{x}) \end{pmatrix} = \begin{pmatrix} \hat{f}^1(\hat{x}_1) & \hat{f}^1(\hat{x}_2) & \cdots & \hat{f}^1(\hat{x}_s) \\ & & \vdots & \\ \hat{f}^n(\hat{x}_1) & \hat{f}^n(\hat{x}_2) & \cdots & \hat{f}^n(\hat{x}_s) \end{pmatrix} \tag{5}$$

where $\hat{f}^i(\hat{x}_j)$ denotes the j-*th* quantified values in i-*th* dimension feature curve.

4.4. Family division

Our intention is to explore the similarity between attacks (*i.e.,* the edges' relationship in a graph) to find partitions, so we identify family division as a community detection problem.

Graph Generation. Based on the aforementioned attack fingerprints, we generate a graph to portray relationships between different types of DDoS. Specifically, the node v_i refers to the category $\mathcal{A}_i$, and the weight A_{ij} of the edge between the nodes v_i and v_j represents the similarity between the categories $\mathcal{A}_i$ and $\mathcal{A}_j$, *e.g.,* the ℓ_2 distance shown in Eq. (6).

Algorithm 1 Function Partition_Search($G, \mathcal{P}$).

Require: The graph G and the initial partition $\mathcal{P}$ ($\{\{v\}|v \in V(G)\}$)
Ensure: The result of partition $\mathcal{P}$
$G_{\text{ori}} = G$, $\mathcal{P} \leftarrow$ Node_Move($G, \mathcal{P}$)
while $|\mathcal{P}| \neq |V(G)|$ **do**
  $\mathcal{P}_{\text{ref}} \leftarrow$ Partition_Refine($G, \mathcal{P}$)
  $G \leftarrow$ Node_Aggregate($G, \mathcal{P}_{\text{ref}}$)
  $\mathcal{P} \leftarrow \{\{v|v \in V(G), v \subseteq C_1\}|C_1 \in \mathcal{P}\}$
  $\mathcal{P} \leftarrow$ Node_Move($G, \mathcal{P}$)
end while
return $\mathcal{P} \leftarrow \{\{v|v \in V(G_{\text{ori}}), v \subseteq C_1\}|C_1 \in \mathcal{P}\}$

$$\ell_2(\mathcal{F}_i, \mathcal{F}_j) = \left\|\mathcal{F}_i, \mathcal{F}_j\right\|_2 = \sqrt{\sum_{p,q} (\hat{f}_i^p(\hat{x}_q) - \hat{f}_j^p(\hat{x}_q))^2} \tag{6}$$

To focus on the significant relationships, we retain the tightly connected edges with top η% weights. Furthermore, this graph construction will be performed t times as cross execution in § 4.3. That is to say, we could obtain t heterogeneous graphs $(G_1, \cdots, G_t)$ based on the same node-set (*i.e.,* DDoS types).

Modularity Metric. Given without other prior knowledge between attacks, the modularity $\mathcal{H}$ is a suitable optimization metric that tries to maximize the difference between the actual number of edges in a community and the expected number of such edges. Consider a graph G, we use e_c to represent the actual number of edges in community c. The expected number of edges can be expressed as $\frac{K_c^2}{2m}$, where K_c is the sum of the degrees of the nodes in community c and $m = \frac{1}{2}\sum_{ij} A_{ij}$ is the total number of edges in the graph. We can calculate the modularity as

$$\mathcal{H} = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j) = \frac{1}{2m}\sum_c (e_c - \gamma\frac{K_c^2}{2m}) \tag{7}$$

where $k_i = \sum_j A_{ij}$ is the (weighted) degree of node v_i, c_i refers to its community, as well as $\delta(c_i, c_j) = 1$ if $c_i = c_j$ and 0 otherwise. Meanwhile, $\gamma > 0$ is a resolution parameter and the higher values lead to more communities.

In addition, the change of modularity by moving a node v to a community C_1 for the partition $\mathcal{P}$ can be represented as $\Delta\mathcal{H}_{\mathcal{P}(v\mapsto C_1)} = \mathcal{H}_{\mathcal{P}(v\mapsto C_1)} - \mathcal{H}_{\mathcal{P}}$. Particularly, when moving an isolated node v_i into a community C_1, its modularity variation can be calculated as

$$\begin{aligned}\Delta\mathcal{H}_{\mathcal{P}(v_i\mapsto C_1)} =& [\frac{\sum_{\text{in}}+2k_{i,\text{in}}}{2m} - (\frac{\sum_{\text{tot}}+k_i}{2m})^2] - [\frac{\sum_{\text{in}}}{2m} \\ & (\frac{\sum_{\text{tot}}}{2m})^2 - (\frac{k_i}{2m})^2] = \frac{k_{i,\text{in}}}{m} - \frac{k_i\sum_{\text{tot}}}{2m^2}\end{aligned} \tag{8}$$

where $\sum_{\text{in}}$ is the weighted sum of the links inside C_1, $\sum_{\text{tot}}$ is the weighted sum of the links incident to nodes in C_1, and $k_{i,\text{in}}$ is the weighted sum of the links with starting node v_i in C_1.

Partition Search. We leverage the Leiden (Traag et al., 2019) algorithm to conduct the partition search for the pre-constructed graphs. Specifically, it consists of three phases: (i) local moving of nodes, (ii) refinement of the partition and (iii) node aggregation for the refined partition. The iteration process for these three phases is described in Algorithm 1.

(i) The first stage is node moving (Algorithm 2), the default value of the initial partition is one node by one community (*i.e.,* $\{\{v\}|v \in V(G)\}$) if without specified. We enqueue all nodes in the graph, and successively move the head node in the queue to the community C_1 that maximizes the modularity increment. After moving the node, its neighbor nodes that do not belong to C_1 will be added to the tail of the queue Q. Until the queue is empty, the process of node moving ends.

(ii) In the stage of partition refinement, the community could be further split into multiple sub-communities. As Algorithm 3 shows, $\mathcal{P}_{\text{ref}}$ is initially set to a singleton partition, and the single community node can be merged with other communities in $\mathcal{P}_{\text{ref}}$ to increase the modularity. Notably, mergers are performed only within each community of the partition $\mathcal{P}$. The prerequisite of merging is that the node and the community C_3 in $\mathcal{P}_{\text{ref}}$ is well connected to their belonging community C_1 in $\mathcal{P}$ (*i.e.,* the nodes in $C_3 \subseteq C_1$ have more than the expected edges $\mathcal{E}(C_3, C_1)$ with other nodes in C_1).

Algorithm 2 Function Node_Move($G, \mathcal{P}$).

Require: The graph G and the partition $\mathcal{P}$
Ensure: The result of partition $\mathcal{P}$
$Q \leftarrow$ Queue($V(G)$)
while $Q \neq \emptyset$ **do**
  $v \leftarrow Q$.pop(), $C_1' \leftarrow \arg\max_{C_1 \in \mathcal{P}\cup\emptyset} \Delta\mathcal{H}_{\mathcal{P}(v\mapsto C_1)}$
  if $\Delta\mathcal{H}_{\mathcal{P}(v\mapsto C_1')} > 0$ **then** (*c.f.,* Eq. (8))
    $\mathcal{P} \leftarrow \mathcal{P}(v \mapsto C_1')$
    Q.push($\{u|(u, v) \in E(G), u \notin C_1'\} - Q$)
  end if
end while
return $\mathcal{P}$

Algorithm 3 Function Partition_Refine($G, \mathcal{P}$).

Require: The graph G and the partition $\mathcal{P}$
Ensure: The result of refined partition $\mathcal{P}_{\text{ref}}$
$\mathcal{P}_{\text{ref}} = \{\{v\}|v \in V(G)\}$
for all $C_1 \in \mathcal{P}$ **do**
  for all $v \in \{v|v \in C_1, E(v, C_1 - v) \geq \mathcal{E}(v, C_1)\}$ **do** (*c.f.,* Eq. (9))
    if $v \in C_2 \in \mathcal{P}_{\text{ref}}$ and $|C_2| = 1$ **then**
      $\mathcal{T} \leftarrow \{C_3|C_3 \in \mathcal{P}_{\text{ref}} \cap C_1, E(C_3, C_1 - C_3) \geq \mathcal{E}(C_3, C_1)\}$
      Random select C_3' within $\mathcal{T}$ according to Eq. (10)
      $\mathcal{P}_{\text{ref}} \leftarrow \mathcal{P}_{\text{ref}}(v \mapsto C_3')$
    end if
  end for
end for
return $\mathcal{P}_{\text{ref}}$

$$\mathcal{E}(C_3, C_1) = \gamma n(C_3)\cdot n(C_1 - C_3) = \gamma n(C_3)\cdot(n(C_1) - n(C_3)) \tag{9}$$

where $n(\cdot)$ refers to the number of nodes in the community. As described in Eq. (10), the larger increment of the modularity, the community is more likely to be selected. The randomness degree of the community selection is determined by a parameter $\beta > 0$, which allows the partition space to be explored more broadly.

$$\Pr(\mathcal{P}_3' = \mathcal{P}_3) \sim \begin{cases} e^{\beta\Delta\mathcal{H}_{\mathcal{P}_{\text{ref}}(v\mapsto\mathcal{P}_3)}} & \text{if } \Delta\mathcal{H}_{\mathcal{P}_{\text{ref}}(v\mapsto\mathcal{P}_3)} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

(iii) In the last stage, the nodes connected to each other will be aggregated, and the edges are rebuilt based on the refined partition $\mathcal{P}_{\text{ref}}$ as designed in Algorithm 4. Note that the aggregate node's community is inherited from the original node in this stage. Ultimately, as described in Algorithm 1, these three stages are iteratively conducted until the number of partitions equals the number of aggregated nodes, *i.e.,* the partition results are no longer improved.

Algorithm 4 Function Node_Aggregate($G, \mathcal{P}$).

Require: The graph G and the partition $\mathcal{P}$
Ensure: The result of graph G
$V \leftarrow \mathcal{P}$
$E \leftarrow \{(C_1, C_2)|(u, v) \in E(G), u \in C_1 \in \mathcal{P}, v \in C_2 \in \mathcal{P}\}$
return $G = \text{Graph}(V, E)$

Membership Generation. As shown in Fig. 1(③), $t-1$ unweighted graphs $G = \{G_1, \cdots, G_{t-1}\}$ are used to obtain the membership $\mathcal{P}_{\text{mem}}$ by performing the above partition search process. Notably, the difference here is that we employ the modularity designed explicitly for the multiplex graphs, as shown in Eq. (11).

$$\mathcal{H}_{\text{multi}} = \frac{1}{2\mu}\sum_{ijlr}[(A_{ijl} - \gamma_s\frac{k_{il}k_{jl}}{2m_l}\delta_{lr}) + \delta_{ij}]\delta(c_{il}, c_{jr}) \tag{11}$$

where $A_{ijl} \in \{0, 1\}$ represents the connection between nodes v_i and v_j in the l-th graph, $k_{jl} = \sum_i A_{ijl}$, $\delta_{ij} = 1$ only if $i = j$, and c_{il} refers to the

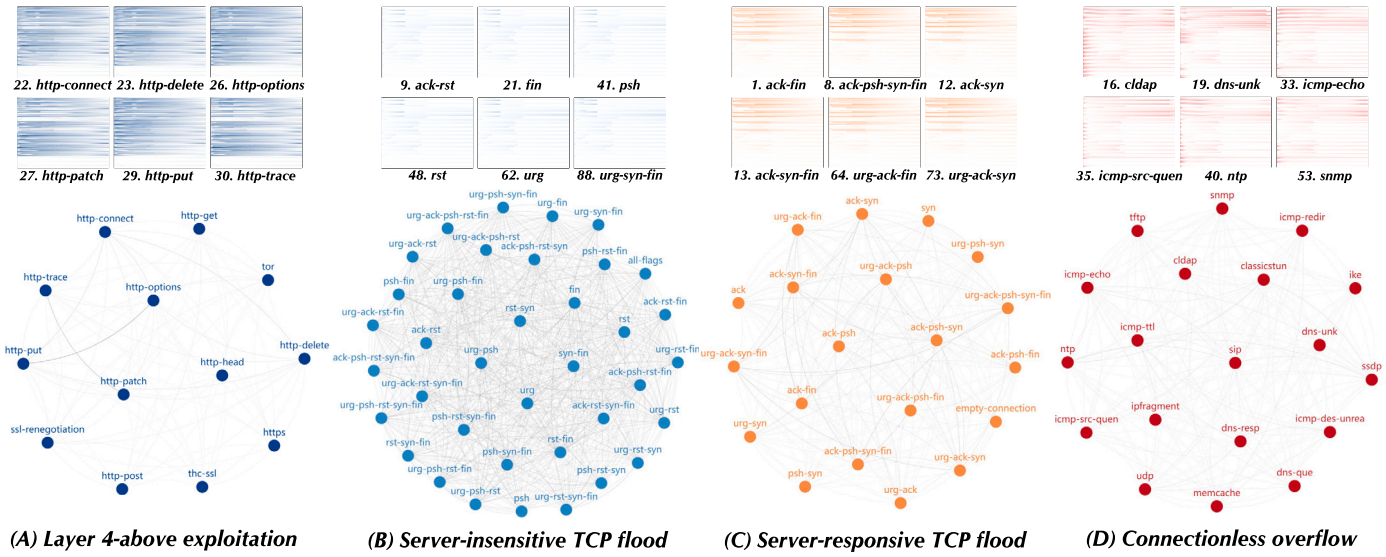


Fig. 2. The built four attack families are based on 89 types of DDoS.

community of the node v_i in the l-th graph. Also, the initial partition of the multiplex graphs adopts the default value.

Family Formation. We use the generated membership as the initial partition to guide the partition research process for the weighted graph. The edges' weights refer to $A_{ij} = 1/\ell_2(\mathcal{F}_i, \mathcal{F}_j)$. This design mainly stems from the consideration that multiple sampling can reduce the contingency of the family division results and enhance its reliability. By this step, we can obtain a family division result recorded as $\mathcal{R}_1$.

Cross-Executed Construction. To mitigate the effects of random errors, the above process will iterate t times, *i.e.,* the $G_i \in \{G_1, \cdots, G_t\}$ takes turns as the weighted graph and others as unweighted graphs to generate membership. We could get t group of results $\{\mathcal{R}_1, \cdots, \mathcal{R}_t\}$ in total, and the final family $\mathcal{R}_{\text{final}}$ is confirmed as the most frequently one from obtained partitions.

$$\mathcal{R}_{\text{final}} = \arg\max_{\mathcal{R}_i} n(\mathcal{R}_i), n(\mathcal{R}_i) = \sum\nolimits_j [\![\mathcal{R}_j = \mathcal{R}_i]\!] \tag{12}$$

5. Evaluation

Our evaluations revolve around the benefits based on the DDoS family, including deep insights for attacks (coarse-grained and fine-grained), reconciling two extremes of ML classification, and optimizing filter strategies. Finally, we conduct the practical evaluation of traffic scrubbing and compare our scheme with the SOTA DDoS mitigation system.

Dataset. In the experiments, we use the real-world DDoS attacks traffic from security vendors and the local Internet Service Provider (ISP). We collect 72 types of DDoS (and the additional 18 types of attacks used for unknown attack detection and family extensibility) from the knowledge base of MAZEBOLT[4] (MAZEBOLT, 2016) such as UDP, ACK-FIN, HTTP-GET floods. Furthermore, we also work with the local ISP[5] to capture DDoS traffic including 17 kinds of attacks, *e.g.,* DNS query, NTP reflection, and ICMP redirect. More details of all 89 types of DDoS could be found in our online repository.

5.1. Entire-spectrum family construction

We first develop the attack family construction over the entire-spectrum DDoS fingerprint. After the partition search, all 89 types of DDoS are assigned to 4 families, as shown in Fig. 2. We term the 4 families as "A: layer-4 above exploitation", "B: server-insensitive TCP flood", "C: server-responsive TCP flood", "D: connectionless overflow". Specifically, family A contains 13 types of DDoS which are all the exploitations above the transport layer. For example, the SSL/TLS supports secure encryption for the application layer and the layer-7 HTTP protocol. Noteworthy, the "tor" (Tor's Hammer Attack) is also a layer-7 DDoS against web servers. The family D involves 19 kinds of attacks that belong to the connectionless overflow, such as ICMP unreachable and SNMP reflection. These attacks can be directly launched without establishing a connection like the TCP handshake. We will further explore families A and D in § 5.2 by analyzing subset attributes.

As for families B and C, they are TCP-based floods with different flag settings. Through family characterization, we find a significant difference between them with respect to the server response pattern. In family B (top part in Fig. 3), the victim has little feedback for the aggressive behavior, *e.g.,* almost no backward packets in the $\mathcal{A}_{42}$: psh-fin flood (the session is disconnected) and $\mathcal{A}_{83}$: urg-rst flood (the session is reset). However, the attacks from family C (bottom part in Fig. 3) will induce the server response, such as the backward RST packet in $\mathcal{A}_7$ and the handshake check (SYN & ACK) in $\mathcal{A}_{56}$. This inspires us to combine backward packets to enhance DDoS defense, we discuss it in § 5.4. Interestingly, although these TCP Flag-related attacks are all Layer-4 attacks in MAZEBOLT's knowledge base, their traffic behaviors are very different.

To compare different family division methods, we chose Leiden and Louvain from community detection, as well as K-means and DBSCAN from unsupervised clustering to conduct experiments. We select modularity as the metric given that it can consider the connection structure between the DDoS types. We set $n_{\text{clusters}} = 4$ and $min_{\text{samples}} = 1$ for K-means and DBSCAN, and the results (after exponential transformation) are summarized in Table 2. It displays that Leiden and Louvain outperform K-means and DBSCAN. Meanwhile, Leiden is slightly better than Louvain. For the two clustering methods, they would be affected by the hyperparameters. K-means achieve the highest modularity

[4] Using MAZEBOLT since it covers most of the popular DDoS attacks, and the knowledge base is relatively comprehensive. Moreover, the knowledge base is derived from actual scenarios and is constantly updated as business changes and attack variants emerge.

[5] We deploy servers in the provincial network gateway and the passed traffic will be mirrored to our servers without affecting/interfering with user services in the backbone path. Traffic collection lasts approximately two years. During this period, all traffic will go through the filters from about 20 security vendors and finally vote to determine its label. This ISP dataset is mainly to expand the types of attacks to supplement MAZEBOLT.

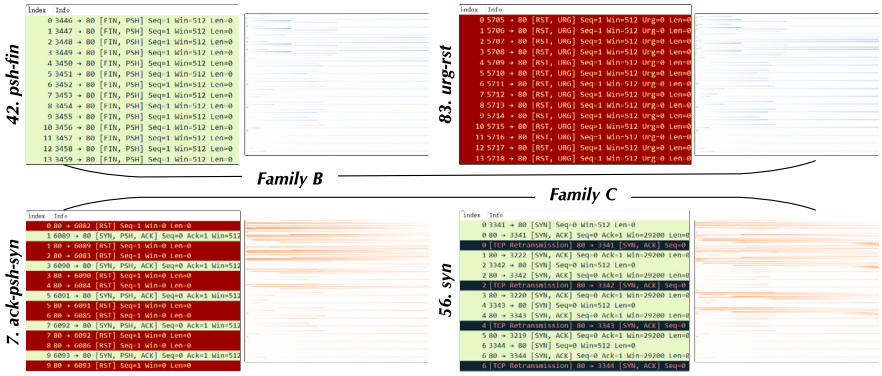


Fig. 3. The instances from families B and C.

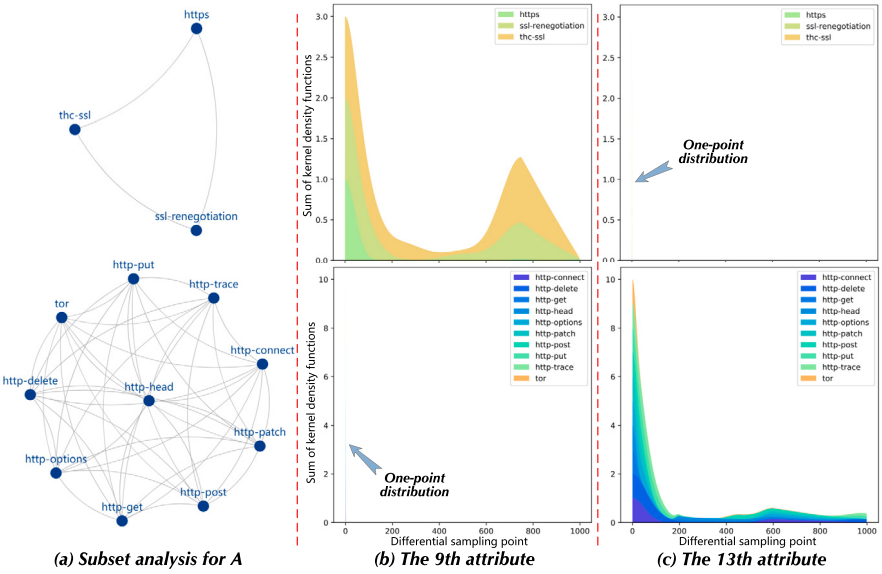


Fig. 4. Further analysis of A based on 9-*th* and 13-*th* attributes.

Table 2
The modularity in different methods.

Modularity ($y = e^x$)			
Leiden	0.99291	**Louvain**	0.99285
K-means ($n_{\text{clusters}} = 4$)			
$P_{tra} = 25\%$	$P_{tra} = 50\%$	$P_{tra} = 75\%$	$P_{tra} = 100\%$
0.98925	0.99215	0.99200	0.99200
DBSCAN ($min_{\text{samples}} = 1$)			
$eps = 2$	$eps = 4$	$eps = 6$	$eps = 8$
0.98883	0.99088	0.99094	0.99204

with the ratio $P_{tra} = 50\%$ of the training set, and remain stable when $P_{tra} = 75\%, 100\%$. Yet the DBSCAN performance improves with a larger cluster radius, reaching 0.99204 when $eps = 8$. Overall, the unsupervised clustering tends to be less stable given the hyperparameters are non-trivial to determine. While the community detection algorithms are relatively suitable to perform the partition search tasks for DDoS family construction.

5.2. Subset attributes analysis

Besides analyzing the full spectrum, we also investigate the fine-grained partitions based on some subset attributes. We explore the {request & response} for family A and the {packets v.s. temporal} distribution for family D. Fig. 4 displays that family A is further divided into two groups based on the 9-th: *tls.handshake* and 13-th: *http.response.code* attributes. It is clear that the kernel density curve of the three attacks, "https", "thc-ssl", "ssl-renegotiation", present similar trends in the 9-th attribute. While the other 10 DDoS types are grouped together given they hold the same distribution in the 13-th attribute. For the remaining two subfigures, the 13-th attribute for SSL/TLS and the 9-th attribute for the HTTP protocol, both show the one-point distribution at x-axis $= 0$. Interestingly, we find "tor" could be characterized by the 13-th dimension in our experiments, but the 12-th dimension (*http.request.method*) can not. It can be attributed to Tor's Hammer Attack disassembling its request as invisible by the TCP segment. This again shows that backward packets can be combined to enhance defenses, as we stated in § 5.1.

In Fig. 5, we depict the fine-grained partitions for family D based on the ratio between the attack time and duration in each snippet. For example, consider the kernel density curve $\hat{f}^{141}(x)$ of 141-th: *timesum_1_0*, the time ratio refers to the area under the curve after normalization, *i.e.*, $R = \sum_{i=1}^{s} x_i \hat{f}^{141}(x_i)$. From the bottom part of Fig. 5, we can find that three groups show different curve trends. Subgroup 1 is relatively stable compared with subgroup 2 which present fluctuates violently, especially when $t \in [0, 2]$. Particularly, subgroup 2 shows a higher peak value $< 100ms$, which indicates we should intercept these flows earlier (corresponds to early disconnection in § 5.4). Subgroup 3 is significantly different from the others, manifested in that the ratios return to

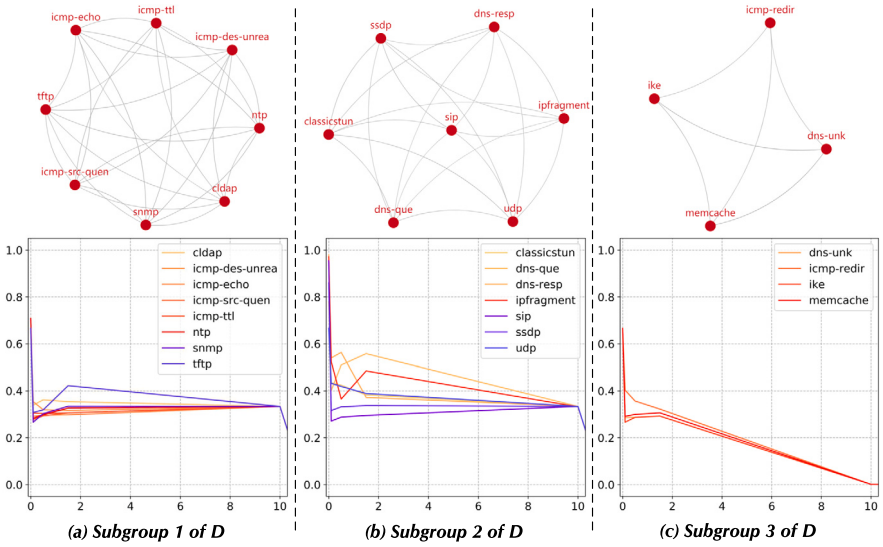


Fig. 5. The {packets v.s. temporal} distribution of family D.

Table 3
The evaluations for guidelines of the supervised classification and unknown class detection. g_4 (★) refers to our DDoS family.

Classification	$g_1(n_c=90)$		$g_2(n_c=5)$		$g_3(n_c=5)$		$g_4(n_c=5)$ ★		$g_5(n_c=2)$	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
LR	62.33	58.57	69.30	64.62	68.21	62.04	95.52	94.78	97.32	97.31
SVM	59.26	51.81	67.46	62.21	68.04	61.91	83.50	81.46	85.89	82.36
DT	78.96	75.02	89.45	88.98	89.57	89.15	99.91	99.89	99.91	99.90
RF	85.33	82.91	89.70	89.45	90.80	91.60	99.92	99.91	99.92	99.92
XGB	82.19	79.03	89.03	88.30	88.90	88.27	99.58	99.23	99.68	99.37
CNN	84.50	82.21	88.16	86.92	89.87	86.32	94.54	93.83	95.03	95.01
CMD	89.15	89.06	93.97	93.68	94.34	93.82	99.73	99.55	99.80	99.76
FS-Net	88.62	84.46	93.86	93.00	93.83	92.90	99.62	99.46	99.64	99.63
ERNN	88.81	84.69	93.21	92.87	93.88	93.02	99.68	99.49	99.71	99.65

Anomaly detection	$g_1(n_c=91)$		$g_2(n_c=6)$		$g_3(n_c=6)$		$g_4(n_c=6)$ ★		$g_5(n_c=3)$	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Kitsune	76.25	73.02	78.23	76.16	80.07	79.92	97.66	96.25	98.31	97.95
Whisper	79.33	77.61	83.81	82.32	84.59	82.41	97.98	97.64	98.30	97.93
HyperVision	80.04	78.19	84.77	82.71	84.93	82.66	98.06	97.82	98.63	98.07

0 at $t = 10$. It means that "ike", "icmp-redir", "dns-unk", and "mamcache" four attacks are mainly concentrated in the first 10s.

5.3. Model guidelines

We perform evaluations for model guidelines, to show DDoS families could guide the granularity of classification/unknown class detection. As mentioned in § 3, both binary and attack-level multi-classification methods tend to be unsatisfactory. The binary classification approaches focus on differentiating "benign" and "attack", the "malicious" label could not be directly used to take countermeasures by network operators. While model enabling identifying each class inevitably performs some accuracy loss when the number of classes increases. Our attack family is promising to realize the trade-off between accuracy and result availability.

5.3.1. Supervised classification guidelines

We conduct supervised model classification tasks using 9 common and well-known ML models, including logistic regression (LR), support vector machines (SVM), decision tree (DT), random forest (RF), xgboost (XGB), CNN (Ma et al., 2020), CMD (Zhao et al., 2023a), FS-Net (Liu et al., 2019), and ERNN (Zhao et al., 2023b). A total of 5 groups of experiments are performed, denoted $g_1 \sim g_5$ respectively. Among them, g_1 conducts the multi-classification task with the number of classes $n_c = 90$ and g_5 develops the binary-classification task *i.e.*, $n_c = 2$. For $g_2 \sim g_4$, all are the multi-classification with $n_c = 5$. Specifically, g_4 performs the family-level classification and $g_2 \sim g_3$ are the controlled experiments that randomly assign 89 attacks into four labels.

The results are summarized in Table 3, the overall model performance, is $g_1 \ll g_2, g_3 \ll g_4, g_5$. We find that all models performed poorly in g_1 due to the difficulty of massive (similar) categories. As we desired, g_5 achieves the best accuracy and F1 score because it only needs binary classification. Although all $g_2 \sim g_4$ are 5-classification tasks, g_4 is fundamentally different from the other two groups. The model effect in g_4 clearly outperforms g_2, g_3, and tends to close to g_5. For instance, the ACC of FS-Net is 93.86%, 99.62%, and 99.64% in g_2, g_4, and g_5 respectively. This means that family-level classification can indeed reconcile the two extremes, to achieve a trade-off between accuracy and practicality (granularity).

5.3.2. Unknown class detection guidelines

We also evaluate three unknown class detection models, including Kitsune (Mirsky et al., 2018), Whisper (Fu et al., 2021), and HyperVision (Fu et al., 2023). Among them, Kitsune designs AutoEncoder to perform unknown class detection, while Whisper and HyperVision

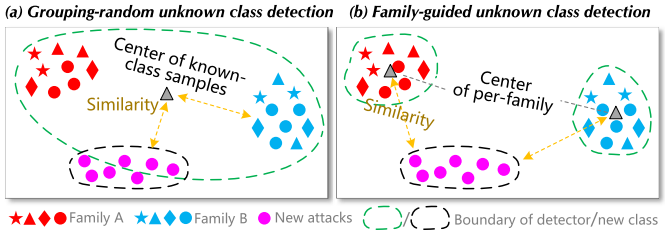


Fig. 6. Illustration of the family-guided unknown class detection.

are both clustering-based, achieving feature extraction with frequency domain and flow interaction graphs respectively. We conduct known class classification as well as anomaly detection against unknown attacks. The known classes refer to benign and 89 types of DDoS attacks, while the unknown classes reference the 18 new types of extended attacks. Particularly, with the update of MAZEBOLT (MAZEBOLT, 2016), we further collect 18 new types of DDoS attacks, including brust-mhddos, cfb-mhddos, get-mhddos, head-mhddos, even-mhddos, gsb-mhddos, cookie-mhddos, cloudscraper http/s-get, cloudscraper http/s-put, cloudscraper http/s-empty-post, cloudscraper http/s-head, cloudscraper http/s-delete, cloudscraper http/s-options, overload, urg-ack-psh-syn, urg-ack-rst-syn, ack-rst-syn, ack-psh-rst. These new DDoS (unknown attacks or variants of known attacks) are used here for unknown class detection evaluation, as well as for exploring the extensibility of the DDoS family in § 6.3. As the bottom of Table 3 shows (perform known class identification and unknown class detection), the model performance trends are consistent with supervised classification, *i.e.,* $g_1 \ll g_2, g_3 \ll g_4, g_5$. This is expected, and we intuitively explain the unknown class detection guiding role of DDoS families in Fig. 6. In subfigure (b), each family forms a separate cluster so that new attacks can be easily distinguished. In contrast, random grouping causes the boundaries of known classes to be expanded in subgraph (a), which will make it difficult to identify unknown-class instances as outliers. Overall, our proposed DDoS family indeed provides model guidelines in terms of supervised classification or unknown class detection, to realize high-accuracy and family-level attack identification.

5.4. Defense strategy optimization

The deploy countermeasures could be more practical for network operators, so we introduce here using attack families to facilitate to optimize defense strategies.

Backward Packet Statistics. Based on the above analysis, we observe that those DDoS attacks of family C will induce server responses and generate backward packets. This motivates us to develop a unified template to achieve defense against multiple attacks. As Fig. 7 (a) shows, we can count the backward packets that match the attacked responses, and use the destination IP (attacker) to update the rule table. This is very different from traditional filtering rules, which typically count forward packets. Particularly, such backward packet statistics have two benefits. (i) We can defend against more DDoS attacks using fewer rules because of the homogeneity of attacks' backward packets. (ii) Backward packet statistics are more robust and enable coping with inapparent/potential attacks that disassemble requests into multiple segments. We explain these statements in detail next. For the 20 types of attacks of family C, the typical approach requires constructing rules for each attack one by one.[6] Based on our template of backward packet statistics, we can use only two rules to realize defense against 20 types of DDoS attacks. In Fig. 7 (b), one rule counts backward RST

[6] For example, to defend against ACK-PSH-SYN flood, the typical rule could refer to counting the forward packets that carry the ACK, PSH, and SYN simultaneously. Furthermore, source IPs whose count results exceed the threshold are added to the blacklist and corresponding packets will be filtered.

```
for_count = count(pkt.tcp.flag == {XXX}, [ip.src])
back_count = count(pkt.tcp.flag == {XXX}, [ip.dst])
if for_count([pkt.ip.src]) > T or back_count([pkt.ip.dst]) > T:
      drop
else:
      pass
```
(a) Defense template of backward packet statistics

```
back_count = count(pkt.tcp.flag == {RST}, [ip.dst])
if back_count([pkt.ip.dst]) > T:
      drop
else:
      pass
```

```
back_count = count(pkt.tcp.flag == {SYN,ACK}, [ip.dst])
if back_count([pkt.ip.dst]) > T:
      drop
else:
      pass
```
(b) Defense family C by using only two rules

```
back_rst = count(pkt.tcp.flag == {RST}, [ip.dst])
back_syn_ack = count(pkt.tcp.flag == {SYN,ACK}, [ip.dst])
if (back_rst + back_syn_ack)([pkt.ip.dst]) > T_Aggregation:
      drop
else:
      pass
```
(c) Defense with threshold aggregation

Fig. 7. The strategy optimization against family C.

```
response_count = count(pkt.http.response == Bad Request, [ip.dst])
if response_count([pkt.ip.dst]) >= T:
      puzzle
else:
      pass
```
(a) Defense tor's hammer with backward packet statistics

```
request_count = count(pkt.http.request, [ip.src])
response_count = count(pkt.http.response, [ip.dst])
if request_count([pkt.ip.src]) >= T_Aggregation or
response_count([pkt.ip.dst]) >= T_Aggregation:
      drop
else:
      pass
```
(b) Defense with threshold aggregation family A

Fig. 8. The strategy optimization against family A.

packets, and the other counts the packets that carry SYN and ACK simultaneously. The former can filter 15 attacks including ack, ack-fin, ack-psh, ack-psh-fin, ack-psh-syn, ack-psh-syn-fin, ack-syn, ack-syn-fin, urg-ack, urg-ack-fin, urg-ack-psh, urg-ack-psh-fin, urg-ack-psh-syn-fin, urg-ack-syn, and urg-ack-syn-fin. And the latter could filter 5 attacks, *i.e.,* empty-connection, psh-syn, syn, urg-psh-syn, and urg-syn.

Moreover, backward packet statistics are robust. For the tor's hammer attack of family A, this attack disassembles TCP packets by setting segments and makes the request invisible/inapparent. Nonetheless, it will induce the "Bad Request" in backward packets (response), so that backward packet statistics could be more robust to perform detection, as shown in Fig. 8 (a).

Threshold Aggregation. As stated in § 1, the problem of slow reaction in traffic scrubbing can be attributed to attack packets from various types but few volumes. To cope with this problem, it will be an effective solution to aggregate the filter threshold based on our DDoS families. In practice, the adaptive adversary could launch multiple kinds of attacks but keep the per-type attack volume small to avoid triggering the filters. An efficient way is to aggregate the statistics of similar attacks in the family so that monitor whether the sum of malicious packets reaches the threshold. As described in Fig. 7 (c), we provide the countermeasure instances for 20 types of attacks in family C. Similarly, we can also aggregate thresholds of HTTP exploitation requests from family A in Fig. 8 (b). By performing threshold aggregation for same-family attacks, it can realize the quick reaction in traffic scrubbing (evaluations in § 5.5 will demonstrate this).

Early Disconnection. For family D, many overflow attacks concentrate a big fraction of traffic in the early stage. For instance, the packets of "dns-response", "ipfragment", "sip" attacks occupy > 90% duration

```
udp_freq = count(pkt.udp, [ip.src]) / duration
icmp_freq = count(pkt.icmp, [ip.src]) / duration
ipfrag_freq = count(pkt.ip.fragment, [ip.src]) / duration
if udp_freq([pkt.ip.src]) > F or icmp_freq([pkt.ip.src]) > F or
ipfrag_freq([pkt.ip.src]) > F:
    drop
else:
    pass
```

Fig. 9. The strategy for early disconnection.

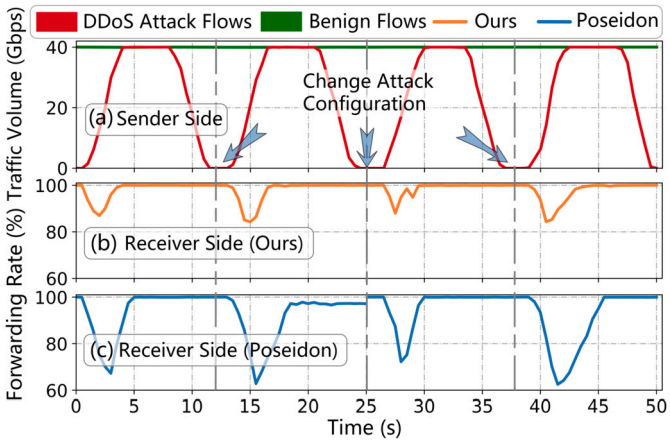


Fig. 10. Compares traffic scrubbing effect of optimized strategy with Poseidon.

in $0 \sim 100ms$.[7] To this end, we could set the strategy like Fig. 9 to reject the high-frequency packets thereby preventing adversaries from quickly overflowing bandwidth in a short period (*e.g.,* ~100 ms).

5.5. Traffic scrubbing evaluation

To further explore the traffic scrubbing effect, we carry out the integration test with the DPDK (Intel, 2014) data plane. As stated in § 5.4, we compare the optimized defense strategies based on our attack families with the recent SOTA method named Poseidon (Zhang et al., 2020). The experiment develops four attack scenes. **Scene 1:** during $t \in [0, 12]$, 15 types of DDoS involving backward RST packets from family A are launched, and Poseidon sets the corresponding strategies, yet we only install one rule to count the backward RST packets. This scenario is designed to verify the insights that only a single rule can protect against multiple types of attacks after strategy optimization. **Scene 2:** during $t \in [12, 25]$, the adversary exploits various HTTP-based attacks from family A including HTTP GET, Tor's Hammer, etc. Poseidon routinely configures typical defense rules, while we propose to statistics backward HTTP response packets. **Scene 3:** during $t \in [25, 38]$, a series of DDoS traffic from family D is generated to attack the victim. Note that the rule is based on the relative threshold within a sliding window, we set $0.1s$ and 1s for ours and Poseidon (reference the early disconnection in § 5.4). **Scene 4:** during $t \in [38, 50]$, the attacker mixes TCP-based and HTTP-based DDoS, and this setting evaluates the effect of threshold aggregation. More details about filter rules can be found in the online repository.

Fig. 10 displays the traffic scrubbing effect in this multi-scene experiment. Subfigure (a) shows the sender traffic, while subfigures (b) and (c) refer to the forwarding ratio of benign traffic in our defense and Poseidon. In scene 1, Poseidon starts to enforce discarding when the number of per-IP DDoS packets reaches the threshold, while our one rule is able to trigger the filter to defense. Meanwhile, our bandwidth curve presents faster restoration due to the count value in the single rule shared for these attacks. In scene 2, since Tor's Hammer Attack

[7] This case displays that attacks from different protocol layers in MAZEBOLT may also have similar behaviors.

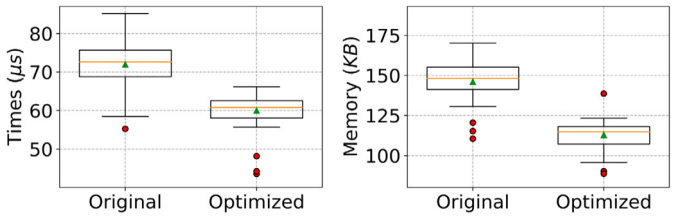


Fig. 11. The time and memory overhead of rules for traffic scrubbing.

disassembles the requests by TCP segment, Poseidon can not identify this attack to present a obvious bandwidth preemption (*i.e.,* can't revert to 100%). By contrast, our strategies perform more robust results, which illustrates the statistics of the backward packets is effective for defending against these attacks. In scene 3, we find that it is more practical to employ a short sliding window as the relative threshold, especially for attacks that are concentrated in the early stage. In scene 4, this scenario is similar to scene 1, the difference is that more types of attacks are launched. Based on the threshold aggregation, our traffic scrubbing performance is still superior with respect to worst bandwidth congestion (84.32% and 62.5% in ours and Poseidon) and restoration time (~5.1 s and ~6.6 s in ours and Poseidon). Overall, the optimized strategies based on attack family insights could realize significant improvement for the DDoS mitigation effect.

Furthermore, we measure the introduced latency and memory overhead for the original rule set and the optimized one. We first run the DPDK layer-3 forwarding program without any rules, then deploy the rule set and perform subtraction calculations to obtain the additional overhead introduced by the defense strategies. In Fig. 11, the original rule set imposes ~73 μs while the optimized only induces ~60 μs. For the memory overhead, our optimized strategy also reduces ~20% compared to the original. Therefore, in addition to speeding up the reaction and improving the effectiveness of traffic scrubbing, strategy optimization also facilitates simplifying the rule set and thereby reduces overhead.

6. Deep insights into family division

In this section, we intend to provide a series of deep insights into family construction with respect to stability, robustness, and extensibility.

6.1. Stability

Our proposed DDoS family construction roadmap involves extracting traffic features, generating attack fingerprints, and executing community division. To explore the stability of the family division process, we develop here evaluation with imbalanced/balanced data and various hyperparameters.

Imbalanced/Balanced Data. In previous evaluations of § 5, we directly perform family division based on the native data proportion, referring to Fig. 12 (a), the class with top-5 most samples are https, ack, ack-psh, ack-psh-fin, and ack-rst. For comparison, we also perform data balancing in Fig. 12 (b) to perform family construction. Whether it is the original data proportion or the balanced (Fig. 12), the inter-class weight (similarity) matrices have almost no difference, thereby generating the isomorphic family partitions. This is because the fingerprint generation of each class mainly includes probability distribution histogram calculation, kernel density estimation, and differential process (refer § 4.3). These steps are not affected by inter-class sample imbalance. Then, subsequent graph generation and partition search will not be affected (refer § 4.4).

Hyperparameter Settings. Next, we construct DDoS family partitions based on Leiden, Louvain, K-means, and DBSCAN with various parameter settings. Specifically, the *initial_membership* settings in Leiden

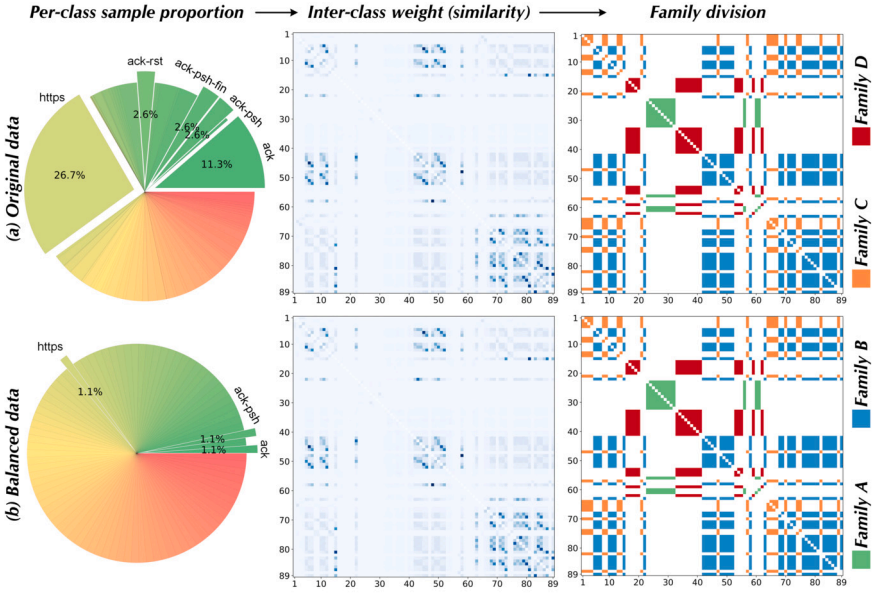


Fig. 12. The family division with different data proportions.

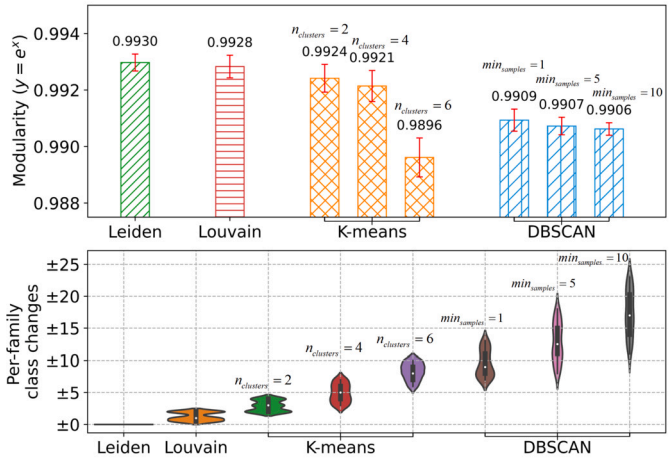


Fig. 13. Family division with different methods based on various hyperparameter settings.

and Louvain both refer to the built membership from unweighted multiplex graphs. Given the Leiden algorithm involves the *n_iterations* parameter, we set it as {2, 3, 5, 10} respectively. For K-means, we set $n_{clusters}$ = {2, 4, 6} and per-group involves P_{tra} = {25%, 50%, 75%, 100%}. For DBSCAN, we set $min_{samples}$ = {1, 5, 10} and per-group involves *eps* = {2, 4, 6, 8}. The random seeds of all methods are set as {1234, 2345, 3456, 4567, 5678}. Multiple sets of experiments are performed, and the average and standard deviation of the modularity results are shown at the top of Fig. 13. Leiden and Louvain achieve better results compared to the clustering algorithms. For K-means and DBSCAN, the former is greatly affected by the $n_{clusters}$ parameter, while the latter is always unsatisfactory (manifested as the low modularity). Meanwhile, the bottom of Fig. 13 displays the class changes of each family, we observe that the trends of class changes are homogeneous with the modularity results. Among them, the family division results of the Leiden method are not affected even if change parameter settings, since the three-phase partition search and cross-executed construction process (refer § 4.4). Louvain method produces a few changes in per-family classes given it is sensitive to initial conditions and not very stable. Regarding K-means and DBSCAN, their results were very volatile and highly correlated with the hyperparameters. Overall, clustering algorithms are severely affected by hyperparameters, and fine-tuning hyperparameters is challenging in practice. However, community detection algorithms are relatively more stable, which is one of the motivations (refer § 3) for leveraging community detection in the family construction roadmap.

Table 4
The family division results against evasion/insertion packets with diverse ratios.

Evasion	Insertion	Family A	Family B	Family C	Family D
5%	0%	±0	±0	±0	±0
10%	0%	±0	±4	±4	±0
20%	0%	±0	±5	±5	±0
0%	5%	±0	±0	±0	±0
0%	10%	±0	±3	±3	±0
0%	20%	±0	±5	±5	±0
5%	5%	±0	±0	±0	±0
10%	10%	±0	±4	±4	±0
20%	20%	±0	±5	±5	±0

6.2. Robustness

Furthermore, due to the rise of adversarial example techniques, we consider tampered traffic and evaluate its impact on family construction. Particularly, unlike image pixels (Zhao et al., 2023c) which can be arbitrarily tampered with, traffic samples need to comply with protocol standards and guarantee attack effectiveness. Recent researches (Wang et al., 2020c, 2021) show that using Selective Symbolic Execution (S2E) can automatically mine the available adversarial instances, including evasion and insertion packets. So-called *evasion* and *insertion* packets refer to the different packet sequences between the middlebox and end hosts due to discrepancies in protocol implementations, *e.g.,* DPI middleboxes usually implement simplified state machines. Therefore, we consider *evasion* and *insertion* attacks since they are practical/representative. In Table 4, we conduct 9 groups of experiments with the diverse ratios of insertion and evasion, *i.e.,* {5%, 10%, 20%}. Intuitively, the larger the proportion of attack packets, the greater the impact on the family division results. Nonetheless, families A and D are not affected in any way even with 20% attack packets. Further, we carefully analyze the affected parts of families B and C, to find the following phenomena. (i) For the urg-ack attack from family C, if the ack flag is removed, it will

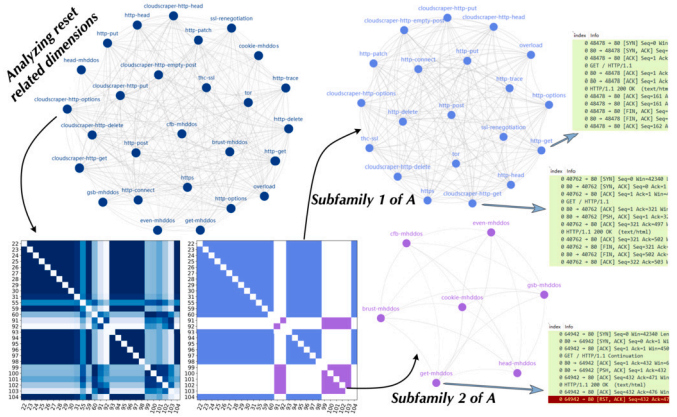


Fig. 14. Extending the family A and analyzing the TCP.flags.reset related dimensions.

become urg attack and fall into family B. (ii) For the ack-psh-rst-fin attack from family B, if the rst flag is removed, it will become ack-psh-fin attack and fall into family C. (iii) For the urg-ack-psh attack from family C, if the rst flag is added, it will become urg-ack-psh-rst attack and fall into family B. (iv) For the urg-syn attack from family C, if the fin flag is added, it will become urg-syn-fin attack and fall into family B. (v) For the psh-syn attack from family C, if the fin flag is added, it will become psh-syn-fin attack and fall into family B. These five situations correspond to the changes of five classes between families B and C in the Table 4. We admit that adversarial traffic samples indeed affect the results of the family division, yet we would like to argue this impact is not very serious. Given the changes usually involve subtle changes between similar attacks, the benefits of the DDoS family such as model guidelines and defense strategy optimization are not fundamentally influenced.

6.3. Extensibility

In the real world, the emergence of new DDoS attacks is inevitable, and so we explore here the extensibility of the DDoS family. There are 18 new DDoS attacks collected for unknown attack detection evaluation in § 5.3.2, and we use them for family extension research. First, we extract features and produce fingerprints based on the traffic of 18 types of attacks. This process is incremental, *i.e.,* does not need to recalculate previously existing attack fingerprints. Then, we update the similarity matrix, which involves calculating the distance of new classes from previous classes. This update is also incremental and only needs to be expanded based on the original matrix. Subsequently, the partition search is performed to achieve family division. This step is not incremental and requires searching partitions for all types of attacks. However, the algorithm execution time takes only about a few seconds, this is acceptable especially compared with expert analysis.

After the above pipeline, among the 18 types of attacks, urg-ack-psh-syn is assigned to family C; urg-ack-rst-syn, ack-rst-syn, and ack-psh-rst are assigned to family B; the other 14 attacks are all assigned to family A. We continue to analyze family A, specifically, we select the TCP.flags.reset related dimensions and calculate ℓ_2 distance to generate similarity matrix. As shown in Fig. 14, those attacks of family A could be divided into subfamilies 1 and 2. The subfamily 2 includes 7 types of mhddos-initiated attacks, *i.e.,* brust-mhddos, cfb-mhddos, get-mhddos, head-mhddos, even-mhddos, gsb-mhddos, and cookie-mhddos. The attacks of subfamily 2 all involve RST flag in traffic content, this is because the MHDDoS (MHDDoS, 2023) tool is designed to forcibly cut off the connection to the victim server to mimic normal connection termination. Therefore, compared Cloudscraper-related attacks (Cloudscraper, 2019), MHDDoS-generated DDoS presents greater changes from the original family A. Overall, family expansion is feasible

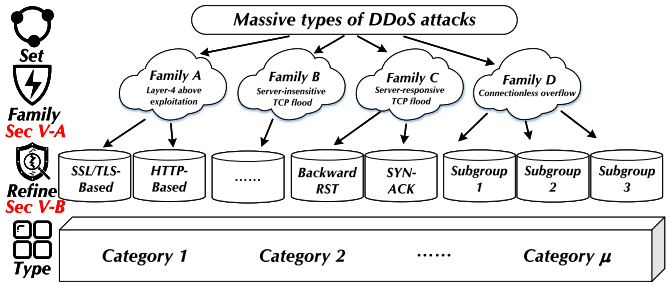


Fig. 15. The hierarchical construction of the attack family.

and practical, and most steps are incremental. A clear characteristic is that different requirements will determine the granularity of the family division, thus it is promising to develop customized family construction, we discuss them in § 7.

7. Discussion

Customized Granularity in Hierarchical Construction. In § 5.1 and § 5.2, we display that the proposed DDoS family partition can perform at different granularities with different feature dimensions. To achieve greater versatility in real-world scenarios, we could develop a hierarchical construction process to adapt to the various requirements, as shown in Fig. 15. The top tends to portray the traffic pattern, while the bottom focuses more on subtle features. For example, the attacks of family A are all layer 4-above exploitations, while they can be further divided into SSL/TLS-based and HTTP-based DDoS according to different field representations. The service-centric users who aim to improve DDoS mitigation may only need coarse-grained family information, yet the security vendors concerned about attack investigation could prefer fine-grained reports. Therefore, building a customized scheme for family divisions benefits widespread promotions to various users.

Real-World Deployment. Based on insights from the DDoS family, there are multiple deployment models for defense strategy optimization in practice. For instance, the Cloud Security Service Providers (CSSPs, *e.g.,* Cloudflare, Arbor, Akamai) play a vital role in securing today's Internet from DDoS attacks. Considering the customized requirements of different users, the corresponding rule configuration of the family division could be deployed at the CSSPs or the victim's ISP. This deployment model facilitates the market management (*e.g.,* add constraints to the output of the family partition results) and rapid expansion (*e.g.,* adapt to emerging attacks) of security services.

Limitations and Future Works. Our work has a few limitations. First, the customized granularity productions depend on the needs of large-scale users, the future work could consider a requirement collection and feedback mechanism. Then, with emerging more categories, the computational complexity may increase such as fingerprint generation and partition construction. Therefore, we would explore which calculation processes could run in parallel to maximize efficiency to cope with more DDoS types. Finally, minimizing training datasets based on constructed families is also a promising direction given it is unrealistic to collect samples of all classes. As part of future work, we will extract the representative categories to characterize their family thereby lightweight the dataset scale.

8. Conclusion

In this paper, we re-examine the current DDoS landscape and pioneer a new direction of attack family to cope with massive types. Our proposed roadmap includes traffic pattern characterization, attack fingerprint production, cross-executed family division, etc. Based on the built family, we discover a series of deep insights into attack behavior. For instance, we find that counting the backward packets could realize a more robust defense effect, which is very different from previous

solutions. Meanwhile, the attack family benefits the ML task by reconciling the two extremes between binary classification and attack-level multi-classification. Furthermore, we perform the defense strategy optimization after inspiring by family partitions. With the practice evaluation on the testbed, we demonstrate our traffic scrubbing performance significantly outperforms the SOTA DDoS mitigation system in terms of worst bandwidth congestion and restoration time. Particularly, only using one rule could defend against 15 types of attacks. Finally, we discuss hierarchical-manner family construction to achieve customized granularity for widespread applications.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

We are grateful to the reviewers for their very constructive feedback and insightful comments. This work was supported in part by National Key R&D Program of China (2023YFB3106800), by National Natural Science Foundation of China (62072398), by SUTD-ZJU IDEA Grant for visiting professors (SUTD-ZJUVP201901), by State Key Laboratory of Mathematical Engineering and Advanced Computing, and by Key Laboratory of Cyberspace Situation Awareness of Henan Province (HNTS2022001), by the Natural Science Foundation of Jiangsu Province (BK20220075), and by the Fok Ying-Tung Education Foundation for Young Teachers in the Higher Education Institutions of China (No. 20193218210004).

References

Ahn, Y.Y., Bagrow, J.P., et al., 2010. Link communities reveal multiscale complexity in networks. Nature 466, 761–764.
Amin, M.T.A., et al., 2017. Unveiling polarization in social networks: a matrix factorization approach. In: INFOCOM. IEEE, pp. 1–9.
Andersen, D.G., 2003. Mayday: distributed filtering for Internet services. In: USNIX USITS.
Andersen, D.G., Balakrishnan, H., Feamster, N., et al., 2008. Accountable Internet protocol (AIP). In: ACM SIGCOMM.
Antonakakis, M., et al., 2017. Understanding the mirai botnet. In: USENIX Security Symposium. USENIX Association, pp. 1093–1110.
Argyraki, K.J., et al., 2005. Active Internet traffic filtering: real-time response to denial-of-service attacks. In: USENIX ATC.
Baldesi, L., Butts, C.T., Markopoulou, A., 2018. Spectral graph forge: graph generation targeting modularity. In: INFOCOM. IEEE, pp. 1727–1735.
Barradas, D., Santos, N., et al., 2021. Flowlens: enabling efficient flow classification for ml-based network security applications. In: NDSS. The Internet Society.
Bartos, K., Sofka, M., et al., 2016. Optimized invariant representation of network traffic for detecting unseen malware variants. In: USENIX Security Symposium. USENIX Association, pp. 807–822.
Caselli, M., Zambon, E., et al., 2016. Specification mining for intrusion detection in networked control systems. In: USENIX Security Symposium. USENIX Association, pp. 791–806.
Cho, K., Shin, K.G., 2016. Fingerprinting electronic control units for vehicle intrusion detection. In: USENIX Security Symposium. USENIX Association, pp. 911–927.
Ciucu, F., Poloczek, F., Schmitt, J.B., 2014. Sharp per-flow delay bounds for bursty arrivals: the case of fifo, sp, and EDF scheduling. In: INFOCOM. IEEE, pp. 1896–1904.
Cloudscraper, 2019. https://github.com/VeNoMouS/cloudscraper.
Dixon, C., Anderson, T.E., Krishnamurthy, A., 2008. Phalanx: withstanding multimillion-node botnets. In: USENIX NSDI.
van Ede, T., Bortolameotti, R., Continella, A., Ren, J., Dubois, D.J., Lindorfer, M., Choffnes, D.R., van Steen, M., Peter, A., 2020. Flowprint: semi-supervised mobile-app fingerprinting on encrypted network traffic. In: NDSS. The Internet Society.
Ester, M., Kriegel, H., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD. AAAI Press, pp. 226–231.
Fayaz, S.K., Tobioka, Y., et al., 2015. Bohatei: flexible and elastic ddos defense. In: USENIX Security Symposium. USENIX Association, pp. 817–832.
Fogla, P., Sharif, M.I., Perdisci, R., Kolesnikov, O.M., Lee, W., 2006. Polymorphic blending attacks. In: USENIX Security Symposium. USENIX Association.
Fu, C., Li, Q., Xu, K., 2023. Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis. In: NDSS. The Internet Society.
Fu, C., Li, Q., et al., 2021. Realtime robust malicious traffic detection via frequency domain analysis. In: CCS. ACM, pp. 3431–3446.
Gilad, Y., Herzberg, A., Sudkovitch, M., et al., 2016. CDN-on-Demand: an Affordable DDoS Defense via Untrusted Clouds. NDSS.
Gong, D., Tran, M., et al., 2019. Practical verifiable in-network filtering for DDoS defense. In: ICDCS. IEEE, pp. 1161–1174.
Guimera, R., Nunes Amaral, L.A., 2005. Functional cartography of complex metabolic networks. Nature 433, 895–900.
Hartigan, J.A., Wong, M.A., 1979. A k-means clustering algorithm. J. R. Stat. Soc., Ser. C, Appl. Stat. 28, 100–108.
Herwig, S., et al., 2019. Measurement and analysis of hajime, a peer-to-peer iot botnet. In: NDSS. The Internet Society.
Holland, J., Schmitt, P., Feamster, N., Mittal, P., 2021. New directions in automated traffic analysis. In: CCS. ACM, pp. 3366–3383.
Intel, 2014. Data Plane Development Kit. http://www.dpdk.org.
Ioannidis, J., Bellovin, S.M., 2002. Implementing pushback: router-based defense against DDoS attacks. In: USENIX NSDI.
Keromytis, A.D., Misra, V., Rubenstein, D., 2002. SOS: secure overlay services. In: ACM SIGCOMM.
Korczynski, M., Duda, A., 2014. Markov chain fingerprinting to classify encrypted traffic. In: INFOCOM. IEEE, pp. 781–789.
Lambiotte, R., Rosvall, M., Scholtes, I., 2019. From networks to optimal higher-order models of complex systems. Nat. Phys. 15, 313–320.
Li, Z., Wang, C., et al., 2014. Improving data forwarding in mobile social networks with infrastructure support: a space-crossing community approach. In: INFOCOM. IEEE, pp. 1941–1949.
Liang, J., Guo, W., Luo, T., et al., 2021. FARE: enabling fine-grained attack categorization under low-quality labeled data. In: NDSS. The Internet Society.
Lin, X., et al., 2022. ET-BERT: a contextualized datagram representation with pre-training transformers for encrypted traffic classification. In: WWW. ACM, pp. 633–642.
Liu, C., et al., 2019. Fs-net: a flow sequence network for encrypted traffic classification. In: INFOCOM. IEEE, pp. 1171–1179.
Liu, X., Yang, X., Xia, Y., 2010. NetFence: preventing Internet denial of service from inside out. In: ACM SIGCOMM.
Liu, X., et al., 2008. To filter or to authorize: network-layer DoS defense against multimillion-node botnets. In: ACM SIGCOMM.
Liu, Z., Jin, H., Hu, Y.C., Bailey, M., 2016. MiddlePolice: toward enforcing destination-defined policies in the middle of the Internet. In: ACM CCS.
Liu, Z., Namkung, H., et al., 2021. Jaqen: a high-performance switch-native approach for detecting and mitigating volumetric ddos attacks with programmable switches. In: USENIX Security Symposium. USENIX Association, pp. 3829–3846.
Ma, X., et al., 2020. Pinpointing hidden iot devices via spatial-temporal traffic fingerprinting. In: INFOCOM. IEEE, pp. 894–903.
Mahajan, R., Bellovin, S.M., et al., 2002. Controlling high bandwidth aggregates in the network. In: ACM SIGCOMM.
Mahimkar, A., Dange, J., et al., 2007. dfence: transparent network-based denial of service mitigation. In: NSDI. USENIX.
MAZEBOLT, 2016. Mazebolt knowledge base. https://kb.mazebolt.com/.
MHDDoS, 2023. https://github.com/MatrixTM/MHDDoS.
Mirsky, Y., Doitshman, T., Elovici, Y., et al., 2018. Kitsune: an ensemble of autoencoders for online network intrusion detection. In: NDSS. The Internet Society.
Naylor, D., et al., 2014. XIA: Architecting a More Trustworthy and Evolvable Internet. ACM SIGCOMM CCR.
Rossow, C., 2014. Amplification hell: revisiting network protocols for ddos abuse. In: NDSS. The Internet Society.
Ruehrup, S., Urbano, P., et al., 2013. Botnet detection revisited: theory and practice of finding malicious P2P networks via Internet connection graphs. In: INFOCOM. IEEE, pp. 3393–3398.
Saha, A., Ganguly, N., Chakraborty, S., De, A., 2019. Learning network traffic dynamics using temporal point process. In: INFOCOM. IEEE, pp. 1927–1935.
Savage, S., Wetherall, D., et al., 2000. Practical network support for IP traceback. In: ACM SIGCOMM.
Shen, M., Wei, M., et al., 2017. Classification of encrypted traffic with second-order Markov chains and application attribute bigrams. IEEE Trans. Inf. Forensics Secur. 12, 1830–1843.
Sisodia, D., Li, J., Jiao, L., 2020. In-network filtering of distributed denial-of-service traffic with near-optimal rule selection. In: AsiaCCS. ACM, pp. 153–164.
Sommer, R., Paxson, V., 2010. Outside the closed world: on using machine learning for network intrusion detection. In: IEEE Symposium on Security and Privacy. IEEE Computer Society, pp. 305–316.
Song, D.X., Perrig, A., 2001. Advanced and authenticated marking schemes for IP traceback. In: IEEE INFOCOM.
Song, Z., et al., 2023. I2RNN: an incremental and interpretable recurrent neural network for encrypted traffic classification. IEEE Trans. Dependable Secure Comput.
Tang, R., Yang, Z., et al., 2020. Zerowall: detecting zero-day web attacks through encoder-decoder recurrent neural networks. In: INFOCOM. IEEE, pp. 2479–2488.

Tong, G., Cui, L., et al., 2016. Terminal-set-enhanced community detection in social networks. In: INFOCOM. IEEE, pp. 1–9.
Traag, V.A., et al., 2019. From Louvain to Leiden: guaranteeing well-connected communities. Sci. Rep. 9, 1–12.
Walfish, M., Vutukuru, M., Balakrishnan, H., et al., 2006. DDoS defense by offense. In: ACM SIGCOMM.
Wang, A., Chang, W., Chen, S., Mohaisen, A., 2020a. A data-driven study of DDoS attacks and their dynamics. IEEE Trans. Dependable Secure Comput. 17, 648–661.
Wang, Q., et al., 2020b. You are what you do: hunting stealthy malware via data provenance analysis. In: NDSS. The Internet Society.
Wang, Z., Zhu, S., Cao, Y., Qian, Z., Song, C., Krishnamurthy, S.V., Chan, K.S., Braun, T.D., 2020c. Symtcp: eluding stateful deep packet inspection with automated discrepancy discovery. In: NDSS. The Internet Society.
Wang, Z., Zhu, S., Man, K., Zhu, P., Hao, Y., Qian, Z., Krishnamurthy, S.V., Porta, T.L., Lucia, M.J.D., 2021. Themis: ambiguity-aware network intrusion detection based on symbolic model comparison. In: CCS. ACM, pp. 3384–3399.
Wu, Y., et al., 2014. Diagnosing missing events in distributed systems with negative provenance. In: SIGCOMM. ACM, pp. 383–394.
Xie, G., Li, Q., et al., 2022. Mousika: enable general in-network intelligence in programmable switches by knowledge distillation. In: INFOCOM. IEEE, pp. 1938–1947.
Xu, C., Shen, J., Du, X., 2020. A method of few-shot network intrusion detection based on meta-learning framework. IEEE Trans. Inf. Forensics Secur. 15, 3540–3552.
Yaar, A., Perrig, A., Song, D., 2004. SIFF: a stateless Internet flow filter to mitigate DDoS flooding attacks. In: IEEE S&P.
Yang, X., Wetherall, D., Anderson, T., 2005. A DoS-limiting network architecture. In: ACM SIGCOMM.
Zhang, M., et al., 2020. Poseidon: mitigating volumetric DDoS attacks with programmable switches. In: NDSS. The Internet Society.
Zhang, X., Hsiao, H.C., et al., 2011. SCION: scalability, control, and isolation on next-generation networks. In: IEEE S&P.
Zhao, Z., et al., 2023a. CMD: co-analyzed IoT malware detection and forensics via network and hardware domains. IEEE Trans. Mob. Comput.
Zhao, Z., et al., 2023b. ERNN: error-resilient RNN for encrypted traffic detection towards network-induced phenomena. IEEE Trans. Dependable Secure Comput.
Zhao, Z., et al., 2023c. SAGE: steering the adversarial generation of examples with accelerations. IEEE Trans. Inf. Forensics Secur. 18, 789–803.

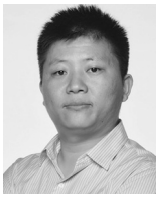
Ziming Zhao is a Ph.D. student in Zhejiang University, Hangzhou, China. He has published more than 10 papers in international journals and conference proceedings, including TIFS, TDSC, TMC, TSE, ESE, CCS, RTSS, and AAAI. His research interests include machine learning, traffic identification, and privacy-preserving.

Zhaoxuan Li is a Ph.D. student in State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering (IIE), Chinese Academy of Sciences (CAS), Beijing, China. He has published more than 10 papers in international journals and conference proceedings, including TIFS, TDSC, TSE, COMNETS, ESE, and ICWS. His research interests include traffic identification, blockchain security, formal methods, and privacy-preserving.

Zhihao Zhou received his B.E. degree in Information Security from Zhejiang University, China, in 2022. He is currently a full time research assistant in Zhejiang University, China. His current research focuses on machine intelligence and cyberspace security.

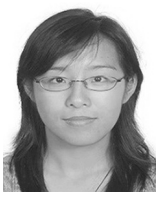
Jiongchi Yu received his bachelor's degree from Zhejiang University, China. He is pursuing the Ph.D. degree at the School of Computing and Information Systems, Singapore Management University (SMU), Singapore. His research focuses on traditional software testing and security issues of cloud native infrastructures.

Zhuoxue Song received the B.S. degree in computer science and technology from Hangzhou Dianzi University, in 2021. He is currently working toward the M.S. degree in the security of cyberspace at Zhejiang University, Hangzhou, China. His main research interests include encrypted traffic classification and intrusion detection.



Xiaofei Xie received the B.E., M.E., and Ph.D. degrees from Tianjin University. He is currently an Assistant Professor with Singapore Management University, Singapore. His research mainly focuses on program analysis, traditional software testing, and quality assurance analysis of artificial intelligence. He was a recipient of the three ACM SIGSOFT Distinguished Paper Awards in FSE'16, ASE'19, and ISSTA'22.

Fan Zhang received his Ph.D. degree from the Department of Computer Science and Engineering, University of Connecticut, CT, USA, in 2011. He is currently a Full Professor with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, and also with the Alibaba–Zhejiang University Joint Institute of Frontier Technologies, Hangzhou. His research interests include system security, hardware security, network security, cryptography, and computer architecture.



Rui Zhang is an associate researcher with the SKLOIS, Institute of Information Engineering, CAS, China. She was a postdoctor in Institute of Software, Chinses Academy of Sciences. She was a visiting scholar in Georgia Institute of Technology from 2009 to 2010 and 2018 to 2019. She has published more than 40 technical papers in international journals and conference proceedings. Her research interests include blockchain security, security protocol and applied cryptography.