Security Data Collection and Data Analytics in the Internet: A Survey

Xuyang Jing, Zheng Yan, *Senior Member, IEEE*, and Witold Pedrycz, *Fellow, IEEE*

***Abstract*—Attacks over the Internet are becoming more and more complex and sophisticated. How to detect security threats and measure the security of the Internet arises a significant research topic. For detecting the Internet attacks and measuring its security, collecting different categories of data and employing methods of data analytics are essential. However, the literature still lacks a thorough review on security-related data collection and analytics on the Internet. Therefore, it becomes a necessity to review the current state of the art in order to gain a deep insight on what categories of data should be collected and which methods should be used to detect the Internet attacks and to measure its security. In this paper, we survey existing studies about security-related data collection and analytics for the purpose of measuring the Internet security. We first divide the data related to network security measurement into four categories: 1) packet-level data; 2) flow-level data; 3) connection-level data; and 4) host-level data. For each category of data, we provide a specific classification and discuss its advantages and disadvantages with regard to the Internet security threat detection. We also propose several additional requirements for security-related data analytics in order to make the analytics flexible and scalable. Based on the usage of data categories and the types of data analytic methods, we review current detection methods for distributed denial of service flooding and worm attacks by applying the proposed requirements to evaluate their performance. Finally, based on the completed review, a list of open issues is outlined and future research directions are identified.**

***Index Terms*—Security-related data, data collection, data analytics, DDoS flooding attacks, worm attacks, security measurement.**

I. Introduction

THE INTERNET has played an important role in our daily life. At the same time when the Internet brings us great convenience, it also raises a lot of security threats. Security threats have become an important factor that restricts the development of the Internet. Recently, security threats have been examined in several different fields. Zou *et al.* [1] first summarized security requirements of wireless networks and then presented a comprehensive overview of attacks encountered in wireless networks. Ahmad *et al.* [2] analyzed various security threats to application, control, and data planes of Software Defined Networking (SDN). Ali *et al.* [3] presented some security threats in cloud computing and provided a comparative analysis of attacks and countermeasures. AbdAllah *et al.* [4] identified five major aspects related to security in Information-Centric Networking (ICN) based on the ICN attacks. Many researchers have considered that the premise of guaranteeing security is to effectively resolve security threats, especially to prevent attacks.

Network attacks that are the main threats for security over the Internet have attracted special attention. The openness and interconnection of the network and the security vulnerabilities of protocols and software lead to multiple and multi-level network attacks. Distributed Denial of Service (DDoS) and worm attacks are two typical attacks over the Internet. DDoS attacks aim to prevent normal users from accessing specific network resources. The distribution of DDoS attacks causes serious damage. Worms, a kind of self-duplicating and self-propagating malicious codes, spread themselves across networks without any human interaction. They compromise hosts by exploiting vulnerabilities in operating systems or installed programs and employ the infected hosts to launch many kinds of attacks. Researchers have completed many surveys to offer an overview on how to effectively counter DDoS and worm attacks. In early work, Peng *et al.* [5] introduced defense mechanisms for DoS and DDoS attacks. They classified defense mechanisms into following categories: attack prevention, attack detection, attack source identification, and attack reaction. Zargar *et al.* [6] surveyed defense mechanisms for DDoS flooding attacks from the perspective of source-based, destination-based, network-based, and hybrid approaches. Yan *et al.* [7] presented a comprehensive survey on defense mechanisms against DDoS attacks using SDN in a cloud computing environment. Li *et al.* [8] presented a survey and compared the Internet worm detection and containment mechanisms from the perspective of activity characteristics of the worms. Kaur and Singh [9] reviewed signature-based worm detection methods based on the classifications of signature generation.

Manuscript received September 21, 2017; revised April 21, 2018; accepted July 29, 2018. Date of publication August 7, 2018; date of current version February 22, 2019. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800704, in part by NSFC under Grant 61672410 and Grant U1536202, in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2016ZDJC-06, in part by 111 Project under Grant B08038 and Grant B16037, and in part by the Academy of Finland under Grant 308087. *(Corresponding author: Zheng Yan.)*

X. Jing is with the State Key Laboratory on Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: xuyangjing91@163.com).

Z. Yan is with the State Key Laboratory on Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with the Department of Communications and Networking, Aalto University, 02150 Espoo, Finland (e-mail: zheng.yan@aalto.fi).

W. Pedrycz is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6R 2V4, Canada, also with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China, and also with the Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia (e-mail: wpedrycz@ualberta.ca).

Digital Object Identifier 10.1109/COMST.2018.2863942

TABLE I
COMPARISON OF OUR SURVEY WITH OTHER EXISTING SURVEYS

Covered Topics	**[5]**	**[6]**	**[7]**	**[8]**	**[9]**	**[10]**	**[11]**	**[12]**	**[13]**	**[14]**	**Our survey**
Summarize security data	N	N	N	N	N	N	N	N	N	N	Y
Review data analytic methods	Y	N	N	N	Y	Y	Y	Y	Y	N	Y
Propose additional requirements of data analytics	N	N	N	N	N	N	N	N	N	N	Y
Review DDoS attacks	Y	Y	Y	N	N	Y	N	N	N	Y	Y
Review Worm attacks	N	N	N	Y	Y	N	N	Y	N	N	Y
Establish the relationship between data categories and attack types	N	N	N	N	N	N	N	N	N	N	Y
Y: discussed; N: not discussed											

Different perspective of surveys offers different understanding of detecting and defending DDoS and worm attacks. However, based on our investigation, there is still an evident lack of a thorough survey from the perspective of summarizing detection methods for DDoS and worm attacks from the view of security-related data. Security-related data (security data, in short) refer to the data that can be used to detect security attacks, security threats or intrusions. It is the basic element in attack detection, no matter whether at a training or detecting stage. Many kinds of methods are employed to analyze security data to detect attacks. Before designing a detection method, we should select: (i) appropriate security data categories that capture sufficient information on malicious activities of attacks, (ii) effective analytic method that has the ability to accurately detect attacks by analyzing security data. By doing this, we can make a detection method accurate and efficient. In order to gain a deep insight into data categories and analytic methods, it becomes necessary to carefully survey current detection methods for network attacks from the perspective of security data.

In this paper, with the purposes discussed above, we firstly divide security data over the Internet into four categories: packet-level data, flow-level data, connection-level data, and host-level data. The packet-level data is defined as the information of packet header, packet payload and packet activities. It gives full information about network activities. Detection methods that use the packet-level data can detect most network attacks. But inspecting individual packets in high-speed network and encrypting packet payload in some circumstances make packet-level data-based methods inefficient. Flow is defined as a stream of packets that have one or more same attributes. It gives a more macroscopic view of network traffic. The flow-level data represents statistical information about the flow. Detection methods that use the flow-level data can be deployed over a high-speed network. They have the ability to solve the detection challenges related to encryption. A connection is defined as the aggregated traffic between two IP addresses. The connection-level data is the statistical description of connection. It provides global information of exchanged traffic between two IP addresses in a given time. We can use the connection-level data in conjunction with the packet-level data and the flow-level data to gain a more detailed insight into network traffic. The host-level data represents information about system events. It records host activities and uses it as a certain decision criterion. From the perspective of these security data categories, we survey current detection methods for DDoS flooding and worm attacks over the last decade. Then, we present a detailed overview on how they use different data categories and analytic methods to detect attacks. By summarizing attack detection methods, we discuss appropriate data categories and analytic methods that can be used to detect specific DDoS flooding and worm attacks. We also propose several additional requirements and use them as a measure to evaluate the performance of existing detection methods.

Although we can find a number of existing surveys about DDoS and worm attacks in [5]–[14], our survey has different focuses, provides deep review on the relationship between data categories and attack types and conducts literature evaluation based on additional requirements of data analytics. A tabulated comparison of our survey with other existing surveys of DDoS and worm attacks is presented in Table I. Through comparison, we can summarize the main contributions of this paper as below:

(i) We summarize commonly used security data over the Internet and divide them into four categories: packet-level data, flow-level data, connection-level data, and host-level data. For each data category, we provide a specific classification and discuss its advantages and disadvantages with regard to the Internet threat detection.

(ii) We propose several additional requirements for security-related data analytics in order to make the analytics flexible and scalable.

(iii) We thoroughly review current detection methods of Distributed Denial of Service (DDoS) flooding and worm attacks from the view of the usage of data categories and the types of data analytic methods by applying the proposed requirements to evaluate their performance. We also summarize what data categories and analytic methods can be used to detect what specific DDoS flooding and worm attacks.

(iv) We further figure out a number of open issues and propose future research directions to motivate network security research.

The rest of the paper is organized as follows. Section II introduces the main categories of security data. For each category of data, we give a specific classification and discuss its advantages and disadvantages regarding the Internet threat detection. In Section III, we discuss analytic methods of security data that are applied to detect attacks and propose a number of security requirements to evaluate the performance

TABLE II
ABBREVIATIONS IN THE PAPER

Abbreviation	Explanation
AL-DDoS	Application Layer Distributed Denial of Service
ANN	Artificial Neural Network
CUSUM	Cumulative Sum
C&C	Command and Control
DDoS	Distributed Denial of Service
DFC	Deal with Flash Crowds
DRDoS	Distributed Reflection Denial of Service
DNS	Domain Name System
DPI	Deep Packet Inspection
D-S	Dempster-Shafer
DT	Dynamic Threshold
EWMA	Exponential Weighted Moving Average
FTP	File Transfer Protocol
HCP	Hop Count Filter
HIDS	Host-based Intrusion Detection System
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
ICN	Information Centric Networking
IP	Internet Protocol
KNN	K-Nearest Neighbor
LDDoS	Low-rate Distributed Denial of Service
NTP	Network Time Protocol
N/T-L DDoS	Network/Transport Layer Distributed Denial of Service
PC	Personal Computer
PI	Protocol Independence
PSD	Power Spectral Density
RTT	Round Trip Time
SDN	Software Defined Networking
SD	Self-adaptive Detection
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSL	Secure Sockets Layer
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TTL	Time to Live
UDP	User Datagram Protocol
VoIP	Voice-over-IP

and quality of analytic methods. In Sections IV and V, respectively, we review current detection methods for DDoS flooding and worm attacks based on the security data categories and applied data analytic methods. Furthermore, we discuss some open research issues and present future research directions in Section VI. Finally, conclusions are presented in the last section. For convenience, the reader can refer to Table II for all abbreviations used throughout the paper.

II. Security Data

For any attacks, detection methods, whether used at the training, testing or detecting stage, are based on data. The data fundamentally affects the efficiency and accuracy of detection methods. Because different categories of data have different application scenarios, we should first consider what category of data could meet our needs when designing a detection method. In this section, for each category of data, we first briefly introduce its collection methods and then present its classification. Based on these findings, we can effectively and explicitly choose data categories to meet the needs of a detection method.

A. Packet-Level Data

Packets are generated when users' programs run such protocols as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP), etc. A packet consists of a packet header and a packet payload. Following the definition presented in the previous section, the packet-level data includes packet header information, packet payload information, and packet activity information.

1) Collection Method of Packet: Normally, we collect network packets at a physical interface by using a specific application programming interface called packet capture (pcap). Libpcap and Winpcap are two packet collecting software libraries for Unix and Windows, respectively. They have many network functions such as protocol analyzers, packet sniffers, network monitors, etc. There are many popular packet collecting tools freely available. For example, TCPdump provides the functions of collecting packets and making some statistical analysis based on tracing files [15]. Wireshark that adds a GUI to TCPdump and includes many traffic signatures can be used for application identification [16]. Snort is a tool for real-time traffic analysis, which is capable of performing content searching/matching and detecting many types of network security attacks [17]. Nmap uses raw IP packets to probe computer networks for host discovery, service and operating system detection [18]. Moreover, libtrace is an efficient packet collection and analysis library that supports multiple data formats [19].

One of the commonly used hardware-based packet collection methods is a mirroring mode. Packet forwarding devices mirror packets coming from one or more ports to another port, to which a capture device connects. This process is called port mirroring, port monitoring, or Switched Port Analyzer session [20]. Port mirroring can analyze both incoming and outgoing packets with a whole network view. But mirroring may induce packet delay, loss or reorder. Thus, having enough bandwidth is essential for a mirror port.

2) Classification of Packet-Level Data: Based on the definition of the packet-level data, we classify them into a number of types as below, which are usually used to detect the DDoS and worm attacks. Mahoney and Chan [21] introduced 33 header features that can be used to detect anomalies. Herein, we briefly discuss some commonly used header features for detecting DDoS and worm attacks.

a) Source/destination IP address: Source/Destination IP address is the fundamental part of data transmission in TCP/IP. It represents the address of a sender/receiver. The distribution and changing rate of source IP address are often utilized when attacks are launched by Botnet. The reason is that the bots usually have a more concentrated source IP address than legitimate users and they are usually strange for attack targets. Usually we apply information entropy to calculate the probability distribution of the source IP address. A high entropy value represents a high degree of randomness of IP addresses. In the same way, the distribution and changing rate of destination IP address can be used to infer the possibility of attacks in the case that a malicious host carries out target scanning.

b) Source/destination port: Source/Destination port exists in the TCP and UDP protocols. Different ports serve for different protocols. The distribution and changing rate of source/destination port are usually used to detect worm scanning. Worms scan a specific port on many destination hosts (horizontal scan) or scan several ports on a single destination host (vertical scan). These behaviors lead to drastic changes in the statistical information of the port.

c) Time to live: The main function of Time to Live (TTL) is to restrict the transmission distance (hop count) of packets. Due to the relative stability of the network, the hop count between a host and another host is located in a certain range. Based on this knowledge, Hop-Count-Filter (HCP) is applied to detect IP spoofing attack by matching the hop count of packets sent from their actual source with the hop count of packets sent from a claimed source [22]. The distribution of TTL values can also be used to detect IP spoofing. If distinct TTL values are observed in packets' headers that come from the same source address, we can infer with high probability that this address has been spoofed [23].

d) Timestamp: Timestamp represents a point-in-time of sending/receiving packets. Accurate packet timestamps are essential in many scenarios, e.g., they are used to compute inter-arrival time of packets, Round Trip Time (RTT) between two hosts and the delay of transmission routes. Timestamp can also be applied to check non-repudiation of packets [24].

e) Packet payload: Packet header features are useful to detect attacks that exploit vulnerabilities of a network stack or scan hosts for vulnerable services. Packet payload information can be used to detect attacks directed at vulnerable applications since the attackers imitate the network behaviors of normal users. For example, some malicious code of worm is carried by packet payload, therefore it cannot be detected by packet header attributes; some application layer DDoS attacks can be launched by using abnormal application contents with legitimate network/transport layer behaviors. Deep Packet Inspection (DPI) is a method that uses both packet header and payload information to determine whether a packet is an intrusion or not [25]. It is effective to detect attacks in application layer. The main challenges for payload-based detection methods are: (i) invasion of privacy, packet payload often encapsulates the data of higher level protocols such as HTTP and DNS, these data contain user private information, so that there is no right to directly check packet payload information, (ii) Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols encrypt application contents that are carried in payload, so it is difficult to detect malicious activities based on the analysis of packet payload.

f) Packet size: Packet size consists of header size and payload size expressed in bytes. Usually, the size of packet header is constant, but payload size is not fixed. It often depends on which system and application the packet is coming from. Some statistical information of packet size is widely used to detect attacks that are launched by Botnet, such as maximum packet size, minimum packet size, average packet size and standard deviation of packet size. The size of attack packets originated from one bot is similar to other packets that come from the same kind of bots. So attack traffic generated from the same botnet has similar or same packet size. Another usage of packet size is to inspect abnormal packets with arbitrary contents in payload. Moreover, we can determine the number of transmitting bytes by accumulating packet size in a certain time period.

g) The number of packets: The number of packets is another widely used data type in current detection methods, e.g., a drastical increase in the number of packets is an indicator of DDoS flooding attacks. According to the direction of packet transmission, this type can be divided into the number of incoming packets, the number of outgoing packets and the number of bidirectional packets. We also classify this data type based on the protocol types, e.g., the number of ICMP (request/reply) packets, the number of TCP packets with different flags (SYN, ACK, FIN, SYN-ACK, RST, etc.), the number of TCP-based protocol packets (HTTP, FTP, SMTP, etc.) and the number of UDP-based protocol packets (DNS, TFTP, NTP, etc.). These classifications can be freely combined with protocol header information or other information, such as the number of incoming TCP SYN packets from a source IP address, the number of incoming packets to a same destination port, the number of outgoing DNS query packets, and so on. Furthermore, we can use packet rate to represent the number of transmitting packets per unit time.

B. Flow-Level Data

In high-speed networks with rates up to hundreds of Gigabit per second (Gbps), collection of packet-level data requires expensive hardware. Moreover, due to the increasing usage of payload encryption and sophisticated obfuscation methods, traditional packet-based detection methods also exhibit poor performance. With the purposes of providing a macroscopic view of the network traffic and endeavoring to deal with the encrypted packets, the concept of flow has emerged. A flow is defined as a stream of packets that have one or more same attributes. These same attributes, usually called flow keys, commonly include packet header information, packet contents and meta-information [26]. For example, the flow keys of NetFlow, introduced in Cisco routers, are source IP address, destination IP address, source port, destination port, IP protocol, IP type of service and ingress interface [27]. Flow aggregation techniques utilize a tuple of predefined flow keys to aggregate packets. Different aggregated granularities on network traffic can be obtained by choosing different flow keys, according to the need of network administrators [28]. Flow has many applications [29]: network monitoring, application monitoring, host monitoring, network application classification and security awareness and intrusion detection. In this paper, we focus on the applications of flow in intrusion detection. Sperotto *et al.* [30] provided a detailed discussion on why flow-based intrusion detection is required and then presented an overview. Umer *et al.* [11] summarized current available flow-based datasets used for evaluation of intrusion detection methods and surveyed flow-based intrusion detection methods.

1) Collection Method of Flow: There are two flow collection strategies [29]: (i) depth-first, choosing specific flow

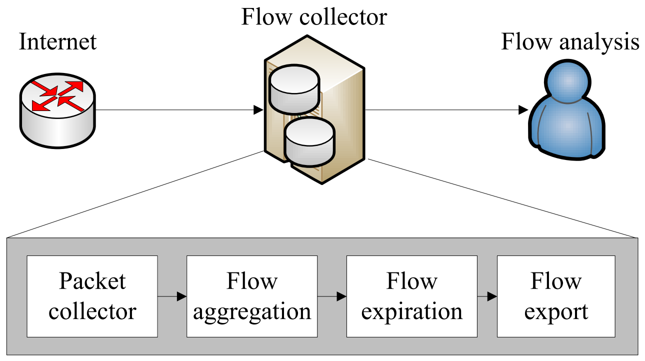


Fig. 1. The collection process of flow.

keys to aggregate packets in order to meet collection demands, (ii) breadth-first, collecting as much as possible information in order to have a global view on network traffic. We simplify the collection process of flow as illustrated in Fig. 1. The complete processes of flow collection and flow exportation are discussed in [26]. The first stage concerns packet collection from a monitor point. Then packets are aggregated into flows with predefined flow keys. During packet aggregation, statistical information of a flow is continuously updated in a flow cache. Once a flow record expires, it is sent to a flow collector for further analysis. A flow is considered expired when [26], [30]: (i) the flow was idle meaning that no packets have been transmitted for a long time exceeding a given threshold (idle timeout), (ii) the flow reaches the maximum allowed lifetime (active timeout), (iii) the FIN or RST flags have been seen in a TCP flow (natural expiration), (iv) automatic reduction of timeout parameters in case of resource constraints (resource constraints), (v) the flow cache memory becomes full (emergent expiration), (vi) all flow cache entries have to be expired in emergent events (cache flush).

2) Classification of Flow-Level Data: Flow-level data represents the statistical information of a flow. We classify flow-level data into the following types. Each type is extracted from flow records.

a) Flow count: Flow count is the number of different flows resulting from aggregating packets with the same flow keys. For example, at a host-side, we can aggregate incoming packets by source address, destination address, and protocol types. In the sequel, many flows with different three attributes can be obtained. Flow count is used to measure this quantity.

b) Flow type: When flow keys consist of port numbers or other protocol identifiers, we can acquire many kinds of flow such as HTTP flow, DNS flow, ICMP flow, TCP (SYN, SYN-ACK, FIN, etc.) flow and others. We can combine flow count with flow types to detect some evidently abnormal traffic. In a normal TCP connection building process, the number of SYN flow is equal to the number of SYN-ACK flow. In the same way, the number of DNS query flow is also equal to the number of DNS response flow. Any abrupt changes of these balanced relationships indicate the presence of attacks.

c) Flow size: A flow is made up of packets that share one or more same attributes. Therefore flow size is the number of packets in a flow. During DDoS flooding attacks, the size of attack flows is distinctly larger than legitimate flows.

d) Flow direction: From the perspective of a specific network, the transmission direction of packets is divided into incoming (inbound traffic) and outgoing (outbound traffic). Flow is generated by aggregating packets. Hence, flow can be classified into inflow and outflow. Inflow is determined by aggregating incoming packets with flow keys and outflow is specified by aggregating outgoing packets. For example, from the view of host-side, there have to be inflows and outflows during a normal TCP connection.

e) Flow duration: As we discussed above, a flow is expired according to six scenarios. So flow duration is the time duration from the first packet's arrival time to the expiration time of flow.

f) Flow rate: Flow rate is defined as the number of transmitting packets of a flow per unit time. In a given flow, the packets share one or more the same attributes. The flow rate has higher specificity than the packet rate but they have mostly similar usages. In the case of high-rate DDoS flooding attacks, the rate of attack flows is distinctly higher than the rate of normal flows.

C. Connection-Level Data

A connection is defined as the aggregated traffic between two IP addresses from the perspective of a specific network, where one address belongs to internal addresses and the other is an external address. If the rule of traffic collection is the same as the rule of flow expiration, a connection is made up of two flows (inflow and outflow). A connection will contain many flows if the collection rule is more extensive than the rule of flow expiration. For example, we define a connection as being composed of transmission traffic between two IP addresses in five minutes. We may get a number of flows (inflows and outflows) in this connection. Notably, the difference between flow and connection is that a flow does not have size restriction, that is to say, the flow is generated even if a single packet has been exchanged. But a connection is generated by at least two packets. Moreover, the notion of connection in this paper is the granularity of network traffic, which is different from the notion of "connection-oriented" in TCP protocol.

Connection-level data describes the traffic statistical information (includes inbound traffic and outbound traffic) exchanged between two IP addresses. It has a higher granularity of network traffic than the flow-level data because it provides global information of exchanged traffic between two IP addresses in a given time. The connection-level data can be divided into the following types.

1) Connection Size: Connection size can be divided into the size of packet-level and the size of flow-level. The size of packet-level is the total number of packets in a connection, while the size of flow-level is the count of flows.

2) Connection Duration: Connection duration is the time duration from connection generation to connection termination. It measures the communication time between two hosts.

3) Connection Count: Connection count is the number of connections in a certain period of time. It offers the information about the number of hosts that a given host connects within that period. A drastic change of connection count is also a signal of attack occurrence.

4) Connection Type: Connection type is determined by the type of collected traffic, such as TCP connection, UDP connection and ICMP connection.

The collection process of connection-level data is actually the process of collecting packet-level or flow-level data. Usually, we make use of the connection-level data together with the packet-level data and the flow-level data to acquire more detailed information about network traffic. By tracing a connection status and applying it together with the first two data categories, we can distinguish attack traffic from normal traffic based on the difference between them. But a detection method using the connection-level data requires keeping track of host and connection information, which implies that such a method requires more resources than other methods.

D. Host-Level Data

The host-level data is collected from a local host while the former three categories are collected from network devices. Various open-source tools can be used to collect host-level data, such as Collectl in Linux [31], Loadrunner in Windows [32], etc. The host-level data provide comprehensive knowledge of system events as it records host activities, host changes, host resource consumption, etc. Each type of attack is intended to have an impact on host performance. The host-level data can describe any internal changes in the host. They are widely used in Host-based Intrusion Detection System (HIDS). A HIDS monitors the internals of a computing system by analyzing host-level data. It is capable of detecting internal abnormal activities such as modification of file systems, privilege escalation, unauthorized logging and unauthorized access. Herein, we discuss some commonly used types of host-level data in attack detection.

1) CPU and Memory Usage: Monitoring CPU usage can deliver useful load information about running software programs. We can measure CPU usage at system level and user level. If a host is suffering from a network/transport layer DDoS flooding attack, the CPU usage of system level will drastically increase, but user level usage will hold steady. However, under application DDoS flooding attacks, such as HTTP requests and asymmetric attacks, both system level and user level CPU usage will simultaneously increase. Memory usage reveals the information about data exchange. It also drastically increases during application DDoS flooding attacks because applications need to process massive data.

2) Operation Log: Operation logs can be classified into equipment operation logs and application operation logs. The equipment operation logs collect the running events of equipment that connects with host, such as keyboard and mouse click events, cursor changes, writable objects, etc. The application operation logs represent user-related activities when using a specific application, e.g., local port creations or destruction events, the number of login events, software usage events, system calls, etc.

E. Summarization and Comparison

Table III shows the summarization and comparison of four data categories. From the table, we can observe that each data category has its own advantages and disadvantages with regard to security threat detection. Packet-level data that has full information about packet payload and packet header can offer detailed records of network activities, making real-time detection and payload matching possible. But having the information of packet payload is double-edged as it could intrude user privacy. Moreover, by considering the need of inspecting individual packets, existing collection methods of packet-level data are not suitable for high-speed networks with a rate up to hundreds of Gbps. Current collection methods of flow-level data provide several advantages compared to packet-level data, such as suitable for high-speed networks and widely deployed. The main merit of collecting flow-level data is to reduce the amount of network traffic to be analyzed. But the flow-level data has no information about packet payload, thus it is not useful in payload-based detection methods. The connection-level data records global information exchanged between two hosts so that it gives a higher granularity of network traffic than the flow-level data. We can measure the communication situation of a given host by analyzing the connection-level data. And when making use of the connection-level data together with the packet-level data or the flow-level data, we can obtain detailed information about network activities. However, collecting the connection-level data needs to keep track of the status of each connection, which requires additional resources. The host-level data differs from the first three data categories as it offers comprehensive information of system events rather than the information of network traffic and records any internal changes of a host. The host-level data are rarely analyzed alone to detect attacks due to the high false positive caused by some normal user activities that lead to abnormal system performance.

Through the above discussion, we can conclude that the selection of data categories is determined by the needs of a detection method and a network environment. We can take the advantages of each data category and use them in conjunction. For example, by considering the connection-level data together with the packet-level or the flow-level data, we can detect application layer related attacks. By analyzing the host-level data with network status, we can accurately infer whether a host suffers attacks.

III. Methods and Requirements of Security Data Analytics

In this section, we briefly introduce the widely used analytic methods of security data and discuss their advantages and disadvantages. In order to make the analytic methods of security data flexible and scalable, we further propose four additional requirements in terms of security data analytics.

TABLE III
SUMMARIZATION AND COMPARISON OF DATA CATEGORIES

Data category	Classification		Advantages	Disadvantages
	Type	Explanation		
Packet-level data	Source address	The IP address of source network interface	1. Access to raw packet data so that make real-time detection possible; 2. Allow pattern matching in payload content; 3. Have full information about network activities.	1. Collection methods are not suitable for high-speed networks; 2. Touch private and sensitive information.
	Destination address	The IP address of destination network interface		
	Source Port	The end-point of source network interface		
	Destination Port	The end-point of destination network interface		
	TTL	The Maximum hop count		
	Timestamp	The point-in-time of sending or receiving packets		
	Packet payload	The content that packet carries		
	Packet size	The size of a packet in bytes		
	The number of transmitting bytes	The number of bytes accumulated in a certain time period		
	The number of packets	The packet count accumulated in a certain time period		
	Packet rate	The number of transmitting packets per unit time		
Flow-level data	Flow count	The number of flows	1. Independent of encrypted payload; 2. Collection methods are suitable for high-speed networks; 3. Collection methods are widely deployed and well understood.	1. Collection methods cause some inevitable delay; 2. Have no information about packet payload.
	Flow type	The protocol information of a flow		
	Flow size	The number of packets in a flow		
	Flow direction	The transmission direction of a flow		
	Flow duration	The time duration of a flow		
	Flow rate	The number of transmitting packets per unit time of a flow		
Connection-level data	Connection size	The number of packets or flows in a connection	1. Provide global information of exchanged traffic between two IP addresses in a given time; 2. Support a good detection performance when being used with other categories of data.	1. Collection methods need to keep track of each connection status; 2. Have poor performance when being used separately from other categories of data.
	Connection duration	The time duration of a connection		
	Connection count	The number of connections		
	Connection type	The protocol information of a connection		
Host-level data	CPU usage	The load information about running software programs	1. Have full information about system performance and behaviors; 2. Record any internal changes of a host system.	1. Have a high false alarm rate when being used separately from other categories of data; 2. Collection methods take up host-side resources.
	Memory usage	The information about data exchange		
	Equipment operation logs	The running records of equipment that connects with a host		
	Application operation logs	The running records of an application		

A. *Methods of Security Data Analytics*

The methods of security data analytics with the purpose of detecting network attacks can be classified into three main categories: statistical methods, machine learning, and knowledge-based methods, as shown in Fig. 2. This classification is based on the nature of data analytic methods. It is not straightforward to propose a classification for security data analytic methods, because there is substantial overlap among the methods used in various classes.

1) Statistical Methods: In statistical methods, the network traffic activity occurring in normal conditions is captured and a profile representing its normal behaviors is generated. The profile is created based on metrics such as packet-level data and flow-level data. In the sequel, a statistical inference is applied to calculate an anomaly score (usually a distance to the profile), which is generated based on currently observed traffic and the normal profile. If the score passes a certain threshold, an alarm of anomaly will be generated. Univariate models, multivariate models and time series models have been applied in statistical methods. The univariate models need prior knowledge of the underlying distribution of data and estimate the parameters from given data (e.g., mean and standard deviation). The multivariate models consider correlations between two or more metrics and do not need prior knowledge of an underlying distribution. The time series models, which use an interval time combining with an event counter or a resource measure, consider the order and the inter-arrival times of observations as well as their values.

There are several widely used examples of statistical methods in attack detection. The first is information entropy. Entropy summarizes the traffic distribution by capturing the important characteristic of traffic features. The traffic distribution is used to detect abnormal behaviors through comparing with a predefined distribution. An entropy-based method is suitable for detecting attacks launched by Botnet based on anomalous patterns in networks [33]. In [34], several measures, such as entropy, conditional entropy, relative entropy, information gain, etc., were used to explain the characteristics of a dataset. Another statistical method is Cumulative Sum (CUSUM) algorithm, which is a sequential analysis technique

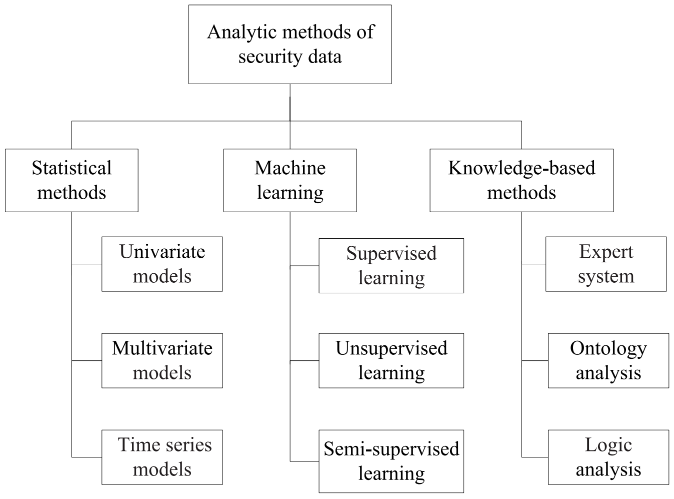


Fig. 2. Classification of analytic methods of security data.

used to detect irregular changes in traffic traces [35]. It is based on the fact that if a change occurs, the probability distribution of random sequence will change from negative to positive. Other examples are Exponentially Weighted Moving Average (EWMA) algorithm [36], Holt-Winter algorithm [37], etc.

The statistical methods have a number of distinct advantages:

- They do not require prior knowledge of network attacks. It means that they have the capability of detecting zero-day attacks.
- They can accurately detect the attacks that cause abruptly changes in network traffic, when setting an appropriate threshold.
- They use few features to characterize the network traffic so that their time and space complexity is small.

But, the statistical methods also have some disadvantages:

- They are susceptible to being trained by an attacker.
- The pure normal behavior model of a network is difficult to establish.
- An appropriate threshold is difficult to set in order to successfully balance false positives and false negatives.
- Most statistical methods rely on an assumption of a quasi-stationary process [38], which is not always realistic.

2) Machine Learning: Machine learning aims to establish an explicit or implicit model of analyzed patterns. The machine learning is mainly divided into three categories [39]–[41]: (i) supervised learning, (ii) unsupervised learning, and (iii) semi-supervised learning. In the supervised learning, the algorithm learns knowledge from labeled data and uses the obtained knowledge to classify the unknown data. Several supervised learning algorithms (e.g., K-Nearest Neighbor (KNN), Support Vector Machines (SVM), and Artificial Neural Network (ANN), Decision Trees, etc.) have been widely applied to detect network attacks [42]. In the unsupervised learning, the algorithm finds the underlying structure of the data without any labels. The unsupervised learning methods that mainly work based on similarity or distance computation are divided into partitioning methods (e.g., K-means, K-medoids, etc.), hierarchical methods (e.g., BIRCH, Chameleon, etc.), density-based methods (e.g., DBSCAN, OPTICS, etc.), and grid-based methods (e.g., STING, CLIQUE, etc.). In semi-supervised learning, a portion of labeled data is mixed into a large amount of unlabeled data to generate the training dataset for unsupervised learning. Here, the labeled data is used to obtain a mapping from a large number of clusters to several classes. Some detailed discussions about how to employ the machine learning to detect network attacks are presented in [39] and [40].

The machine learning has a number of advantages:

- It has high detection rate.
- It is capable of updating their execution processes in response to new traffic.

But, it also has some disadvantages:

- The supervised learning cannot detect unknown attacks until relevant information is fed for retraining.
- The unsupervised learning needs prior knowledge to determine the number of clusters and is under the assumption that the large clusters are normal and small clusters are abnormal.
- The machine learning consumes more resources in both training and updating processes than other two types of methods.

3) Knowledge-Based Methods: In the knowledge-based methods, network or host events are matched with predefined attack rules or patterns or signatures to examine them for the presence of known attack instances. Expert system is the most widely used knowledge-based methods. The expert system extracts the specific features from the training data and builds a rule in order to classify new coming data. Another type of the knowledge-based methods is ontology analysis. It expresses the relationships between collected data and uses these relationships to infer particular attack types. Furthermore, logic analysis models attack patterns in an expressive logic structure and uses this structure to determine whether network events are legal.

The knowledge-based methods exhibit the following advantages:

- They are simple, robust, and flexible.
- They have a high detection rate under the circumstance that attack rules or patterns or signatures are accurately established.

Some disadvantages of the knowledge-based methods are identified as below:

- They cannot detect unknown attacks.
- They need high-quality prior knowledge.
- These methods may trigger some false alarms due to non-availability of biased normal and attack datasets.

B. Requirements of Security Data Analytics

Various theories and methods can be applied to analyze security data so as to detect network attacks. Traditional requirements (such as real-time, high accuracy, less consumption, widely deployment, etc.) have been usually considered when designing attack detection methods. But the Internet attacks are growing day-by-day, novel detection methods should be developed. As a result, additional requirements on detection methods should be further considered for the purpose of detecting all potential security threats and attacks and applying into network security measurement. With the purpose of making detection methods more flexible and scalable, we propose four additional requirements. By considering these four requirements, we believe future detection methods will show advanced performance.

1) Self-Adaptive Detection (SD): The anomaly-based detection methods typically build normal profile for network behaviors, and detect intrusion based on the deviation between normal profile and current network profile [43]. They have the ability to detect zero-day attacks while misuse-based detection methods can only detect known attacks by using signatures. But an attacker can elaborately launch attacks by intentionally training detection methods to make it gradually accept abnormal network behaviors as normal. Facing such an attack, it is hard to define what types of network behaviors are normal. If the patterns of normal network behaviors are wrong or incomplete, anomaly-based detection methods will have a high false positive rate. Therefore the detection methods should adaptive to the changes of current network behaviors instead of mainly depending on a static profile, that is to say the detection methods should be capable of updating the detection strategies with traffic changes.

2) Dynamic Threshold (DT): Most detection methods set a threshold during the measurement of deviation in anomaly-based technologies or make it act as an upper bound for measuring the number of anomalies. Actually, it is difficult to choose a proper value as the threshold that can be applied in all kinds of networking scenarios. Setting a fixed threshold value could greatly impact detection accuracy and efficiency. Thus, it is preferred that the threshold should dynamically accuracy and efficiency. Such a dynamically changed threshold could greatly balance between false positive rate and false negative rate.

3) Protocol Independence (PI): There are many kinds of DDoS flooding and worm attacks in the context of various networking protocols. We cannot predict when and which type of attacks will occur. A qualified detection method should be protocol independent for detecting DDoS flooding and worm attacks. Both direct and indirect DDoS flooding attacks have their own attack characteristics and could exploit multiple protocols to launch attacks. So a protocol-independent method is expected to detect a category of DDoS flooding attacks instead of detecting a specific protocol-related attack. For worm attacks, we prefer efficient early detection at scanning and propagating phases. Although worms have many scanning and propagating schemes, they have a similar purpose. A protocol independent method can detect both known and unknown worms based on the scanning or propagating schemes. In particular and from the view of network security measurement, a generic and pervasive method is highly expected for evaluating the Internet security as a whole.

4) Deal With Flash Crowds (DFC): Flash crowds that massive legitimate users send request packets to a server in a short period of time, have very similar properties to DDoS flooding and worm attacks. But they are legitimate network traffic. A comprehensive comparsion between DDoS flooding attacks and flash crowds is shown in Table IV [5]. Moreover, with the development of the Internet, flash crowds occur more and more frequently. An attacker usually launches malicious activities as soon as flash crowds occur. It can hide its activities under flash crowds and achieves its malicious goals. The detection methods that have the ability of dealing with flash crowds and distinguishing attacks accompanying with flash crowds are urgently needed.

TABLE IV
COMPARISON BETWEEN DDOS FLOODING ATTACKS AND FLASH CROWDS

	DDoS flooding attacks	**Flash Crowds**
Network impact	Congested	Congested
Server impact	Overloaded	Overloaded
Traffic	Malicious	Genuine
Response to traffic control	Unresponsive	Responsive
Traffic type	Any	Mostly Web
Traffic amount	Any	Large amount of traffic
Predictability	Unpredictable	Mostly predictable

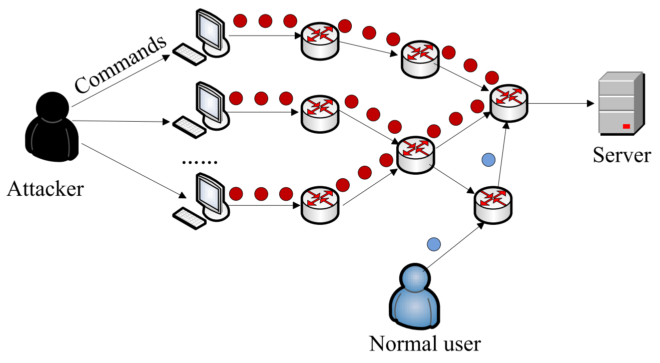


Fig. 3. DDoS flooding attacks.

IV. SECURITY DATA ANALYTICS FOR DDOS FLOODING ATTACK DETECTION

Distribution and cooperation are the main characteristics of DDoS attacks. There are two methods to launch DDoS attacks over the Internet. One is called protocol attack (also called vulnerability attack and killer packet attack). It exploits vulnerability of protocols or applications and sends some malformed packets to the victim. The victim will crash when processing these malformed packets. For example, Ping of Death attack, Winnuke attack and Teardrop attack fall into this category. Another attack is called DDoS flooding attack, which is the most common one, as shown in Fig. 3. This type of attack floods a victim and occupies the victim's resources so that it cannot provide normal services for legitimate users.

Usually attackers launch DDoS attacks with massive hosts that have installed malware programs. A botnet is made up of compromised hosts (zombies) and controlled by an attacker (botmaster). The botmaster utilizes command and control system (C&C system) to remotely control and issue attack commands to zombies. Botnets have been studied extensively. Khattak *et al.* [44] presented three comprehensive taxonomies of botnet features. The first taxonomy categorizes botnet behaviors as those concerning propagation, rallying, C&C,

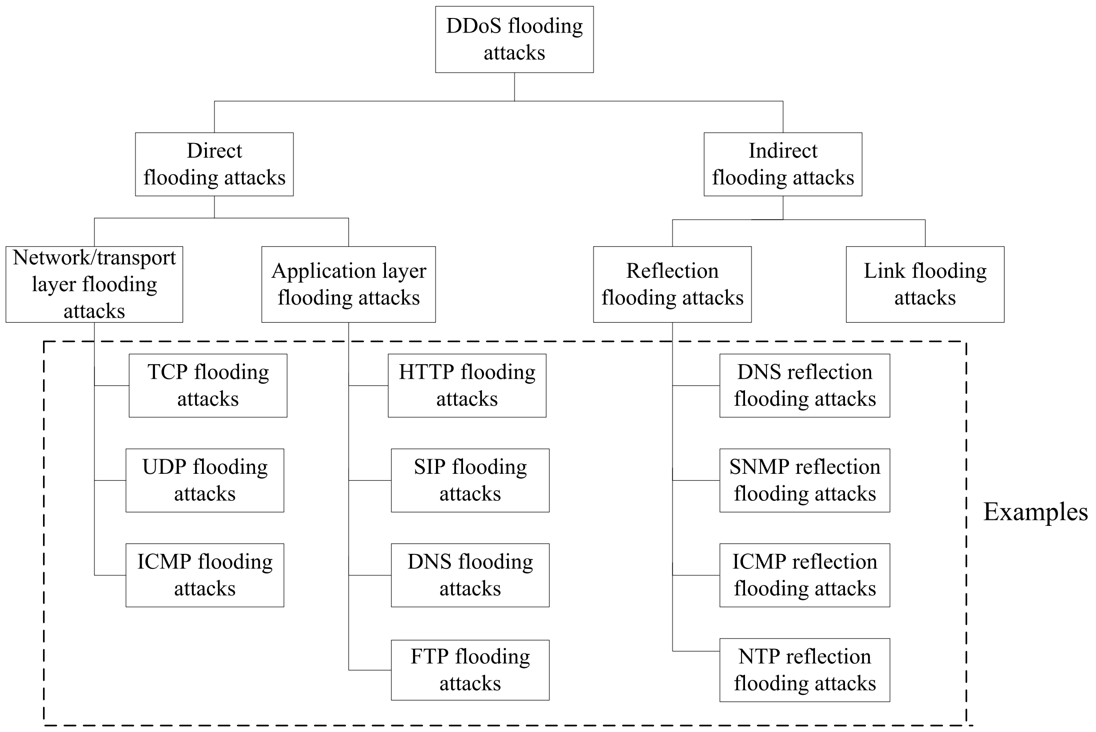

DDoS flooding attacks
Direct flooding attacks
Indirect flooding attacks
Network/transport layer flooding attacks
Application layer flooding attacks
Reflection flooding attacks
Link flooding attacks
TCP flooding attacks
UDP flooding attacks
ICMP flooding attacks
HTTP flooding attacks
SIP flooding attacks
DNS flooding attacks
FTP flooding attacks
DNS reflection flooding attacks
SNMP reflection flooding attacks
ICMP reflection flooding attacks
NTP reflection flooding attacks
Examples

Fig. 4. The classification of DDoS flooding attacks.

purpose, topology, and evasion. The second taxonomy classifies botnet detection methods into bot detection, C&C system detection and botmaster detection. The third taxonomy classifies botnet defense mechanisms into preventive and remedial. Hoque *et al.* [10] classified botnet into stationary botnets and mobile botnets. They not only introduced current development trends of botnets, but also pointed out challenges when designing botnets.

To prevent DDoS attacks, there are two different ways: (i) direct attack detection that exploits the characteristics of DDoS attacks, (ii) botnet detection that detects bots, botmaster and C&C system. In this paper, we focus on the detection methods for DDoS flooding attacks.

DDoS flooding attacks can be generated in two ways: direct flooding attacks and indirect flooding attacks, as shown in Fig. 4 (we only list some specific examples, there still are many examples based on different protocols). In the direct flooding attacks, attackers usually spoof source IP address of attack packets and send them to a victim directly. In the case of indirect flooding attacks, attackers use many innocent intermediates to flood the victim indirectly. Based on this classification, we surveyed current detection methods for DDoS flooding attacks based on the data categories and the methods of data analytics they used. We discussed the current work based on some data categories in each data analytic method because other categories had not been found in the usage of corresponding attack detection in the literature as best as we know. Notably, due to the detection problem of Low-rate DDoS flooding attacks that we will explain in the following, we separately discuss this type of attacks. Moreover, a discussion in each subsection is given with the purpose of identifying what categories of data should be collected and which analytic method should be applied for each type of attack.

A. Detection Methods Against Network/Transport Layer DDoS Flooding Attacks

The main purpose of Network/Transport Layer DDoS (N/T-L DDoS) flooding attacks is to consume network bandwidth and overwhelm network infrastructures by sending a large volume of attack traffic. These attacks often exploit weaknesses of network/transport protocols.

SYN flooding attacks are one of the most frequently happening N/T-L DDoS flooding attacks which exploit the weakness of TCP three-way handshake. An attacker sends massive SYN request packets to a victim without subsequent ACK reply packets. The victim is busy in processing these request packets with a half-open connection so that it has few resources to process normal user's request packets. In this part, the detection methods of SYN flooding attacks are mainly discussed, because UDP protocol is currently used to launch reflection flooding attacks and ICMP flooding attacks can be easily defended by firewall.

1) Statistical Methods: In this part, we discuss statistical analytic methods for detecting N/T-L DDoS flooding attacks using packet-level data, flow-level data and connection-level data.

a) Use packet-level data: Bellaiche and Gregoire [45] used the numbers of SYN, ACK, SYN-ACK and RST packets that drastically change in unusual TCP handshakes to detect attacks. Entropy is calculated based on the fraction of these packet counts occurring under normal traffic. During SYN flooding attacks, massive unusual handshakes cause the drastic change of the numbers of the above four types of packets. SYN flooding attacks can be detected when the variation of entropy value exceeds a certain threshold. The detection method is independent from the traffic volume and does not produce false alarms during flash crowds. Thus, this method can satisfy the requirement of DFC, but cannot achieve the goals of SD, DT, and PI.

Based on the similar principle, Sengar *et al.* [46] used normalized frequencies of SYN, SYN-ACK, FIN and RST packets during non-attack traffic to build normal probability distributions. During detection, current probability distribution of packet counts is built. Hellinger distance is employed to measure the distance between normal probability distributions and current probability distributions. A dynamic threshold was computed by employing a stochastic gradient algorithm based on the Hellinger distance observed during a training period. An alarm will be raised when the Hellinger distance is higher than an estimated threshold. Their method has the ability of detecting the attacks that violate the normal communication steps of protocols. Therefore, the method can support DT and PI, but the requirements of SD and DFC cannot be satisfied.

Boro *et al.* [47] considered two detection features for UDP flooding attacks targeted to a particular destination: (i) the count of total destination port changes for non-spoofed source addresses with random destination ports, and (ii) the count of source addresses changes for randomly spoofed source addresses. A KD-Tree that each node represents a unique source address is used to record information of incoming traffic. In each time interval, they performed pre-order traversal of the KD-Tree to calculate the entropies of the count of destination port changes and the count of source address changes. An alarm will be raised to indicate UDP flooding attacks when the two entropies beyond a preset threshold. However, this method cannot support all additional requirements.

Kim *et al.* [48] proposed a statistics-based malicious packet detection and filtering scheme named PacketScore. PacketScore utilizes the notion of "Conditional Legitimate Probability" (CLP) based on Bayesian theorem to judge whether a packet is legitimate. The CLP of a packet is calculated by comparing its attribute values with the values in a baseline profile. The baseline profile contains the distribution of six attributes namely packet size, TTL value, source address prefix, TCP flag and server port number, and some joint distribution, such as <packet size and protocol-type> , <server port number and protocol-types>, etc. Once the score of every packet is computed, PacketScore prioritizes packets based on their scores and performs selective packet filtering by comparing the scores with a dynamic threshold that is determined by a recent snapshot of CLP values. The method can support DT and PI, but cannot satisfy the requirements of SD and DFC. Some extended schemes of PacketScore were described in [49] and [50].

b) Use flow-level data: Zhou *et al.* [51] utilized source address and destination address to aggregate packets into flows. Based on the knowledge that attack packets that originated from the same botnet have the similar size, they calculated entropy of the distribution of packet size in each flow. The more concentrated the packet size is, the lower the entropy is. They detected DDoS flooding attacks by measuring the entropy deviation between normal and current traffic. Their method is protocol independent because they only used the packet size in each flow. Thus, the method can support the requirement of PI. But it cannot achieve the goals of SD, DT, and DFC.

c) Use connection-level data: David and Thomas [52] monitored traffic on connection-level and aggregated traffic into flows with 5 tuples (viz. source and destination addresses, source and destination ports, and protocol type). The entropy of flow counts was calculated for each connection in each time interval. Moreover, an dynamic threshold was calculated based on the traffic pattern of network activities and user behaviors. DDoS flooding attacks are detected when the variation between the entropy of flow counts at each instant and the mean value of entropy in that time interval are greater than their corresponding thresholds. The method can achieve the requirement of DT, but cannot support SD, DT and DFC.

2) Machine Learning: In this part, we discuss machine learning for detecting N/T-L DDoS flooding attacks using packet-level data, flow-level data and connection-level data.

a) Use packet-level data: Saied *et al.* [53] applied an Artificial Neural Network (ANN) to separate attack traffic from genuine traffic based on specific features. They compared the features of legal packets, which are generated by normal applications with the features of illegal packets that are generated by DDoS attacking tools. Then, they used distinct features as input variables to train TCP/UDP/ICMP topological ANN structures, including source IP address, source and destination port number, packet size, the number of packets, etc. The trained ANN can detect known and unknown DDoS flooding attacks with high accuracy. Thus, the method can support the requirement of PI, but cannot achieve the goals of SD, DT and DFC.

Vijayasarathy *et al.* [54] proposed a real-time detection method using a Naive Bayes classifier. They used windowing to split input traffic into traffic subsets in order to obtain reasonable estimation and control over the reaction time of the system for attacks and have better models from big training datasets. In the phase of training, the system takes traffic statistics namely the amount of source and destination addresses, source and destination ports, the number of packets, TCP flags, packet size and packet arrival time as input and trains them using the Naive Bayes algorithm. 10-fold cross validation was used to evaluate the accuracy of attack. But this method cannot support all additional requirements.

Su [55] proposed a detection method by combining with a weighted k-nearest neighbor (KNN) classifier. The author derived 35 features from headers, including IP, TCP, UDP, ICMP, ARP, and IGMP. A weight value is calculated for each feature and an optimal vector of weighted features is used for attack classification. This method achieves 97.42% detection accuracy for known attacks and 78% accuracy for unknown attacks. The method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

The traffic of DDoS flooding attacks launched by a Botnet is quite different from flow crowds. Kong *et al.* [56] identified some statistical features to discriminate DDoS flooding attacks from flash crowds, such as the number of unique source addresses in each interval, the number of increased source addresses in adjacent interval, the average of the number of packets sent by source addresses in each interval, and the standard of the number of packets sent by source addresses in each interval. With these features, traffic is classified by employing some supervised methods. The method can satisfy DFC, but cannot support SD, DT and PI.

Lee *et al.* [57] proposed a proactive detection method for DDoS flooding attacks by exploiting an architecture that consists of the selection of victims that help attack, the communication between an attacker and victims, and the process of launching an attack. The entropy of source and destination addresses, the entropy of source and destination ports, the entropy of packet types, the occurrence of packet types (TCP, UDP, and ICMP), and the number of packets are computed to build feature vectors based on a sample of consecutive packets. A hierarchical clustering algorithm was adopted to form clusters using the feature vectors. The normal traffic and each phase of DDoS flooding attacks are partitioned. The method can support the requirement of PI, but cannot achieve the goals of SD, DT and DFC. An extended detection method combined with a feature ranking algorithm was proposed in [58].

b) Use flow-level data: Data centers provide a various of services and applications such as Web, FTP, DNS, Hadoop, etc. Many services containing a data center can easily lead to corresponding DDoS attacks. Traditional packet-based DDoS attack detection methods seem impractical. Xiao *et al.* [59] proposed an efficient detection method deployed at data centers by applying CKNN (K-nearest neighbors traffic classification with correlation analysis) on flow-level data. They aggregated packets that come from the data center network with identical 5 tuples (viz. source and destination addresses, source and destination ports, and protocol type). Each flow can be represented by a set of statistical features, such as flow duration, flow size, etc. Due to the high correlation among the flows, which were generated by the same application, the method can detect the existing attacks by examining flow features with CKNN classification and correlation analysis. The detection method that makes full use of the similarity among flows instead of pre-building a profile is self-adaptive and protocol independent. Thus, this method can support SD and PI, but cannot satisfy the requirements of DT and DFC.

Wagner *et al.* [60] designed a kernel function based a one-class SVM classifier to detect attacks. A special kernel function considers properties of Netflow records and projects data points into a higher dimension before classification. Such a one-class SVM classifier is capable of identifying outliers and anomalies. They tested the detection method with some types of flooding attacks and obtained high detection accuracy. Thus, the method can satisfy the requirement of PI, but cannot achieve the goals of SD, DT and DFC.

Qin *et al.* [61] aggregated packets into flows with 5 tuples (viz. source and destination addresses, source and destination ports, and protocol type). For each flow, they recorded packet size, source address, destination address, destination port, and flow duration. Then they constructed entropy vectors of these five features and modeled normal profiles using a K-means algorithm. By comparing current traffic profiles with the normal traffic profiles, the deviations of entropy vectors are calculated and compared with a threshold to figure out potential attacks. The method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

c) Use connection-level data: Kumar and Selvakumar [62] proposed a classification algorithm called RBPBoost by combining ensemble of neural classifiers and Neyman Pearson cost minimization strategy. Several features were extracted from traffic during a specified time window, such as the number of connections to the same host, the number of connections having the same packet length, the number of connections that have SYN errors using the same service, the number of UDP echo packets to a specified port, etc. In training phase, an ensemble of neural classifiers was trained using the above features for each individual dataset and the results were combined. In testing phase, the Neyman Pearson Structural Risk Minimization was applied to make a final classification decision. But the method cannot support all additional requirements.

3) Knowledge-Based Methods: In this part, we discuss knowledge-based analytic methods for detecting N/T-L DDoS flooding attacks using packet-level data, flow-level data and connection-level data.

a) Use packet-level data: Sun *et al.* [63] discussed five kinds of skillful SYN flooding attacks and proposed an accurate and fast detection method, named SACK^2. Under a normal TCP three-way handshake, the number of SYN-ACK packets that a server sends should be almost equal to the number of ACK packets that a client sends. However during the SYN flooding attack, this balanced relationship will be broken with a large deviation as there are no corresponding ACK packets. SACK^2 utilizes counting bloom filters to observe the difference of SYN-ACK packets and ACK packets in number. Each bucket in bloom filters is hash value that is calculated by using client's and server's IP addresses, client and server's ports, and client and server's initial sequence numbers of TCP packets. SACK^2 does not leave any chance for an attacker to evade the detection, so it can detect all kinds of SYN flooding attacks and distinguish attacks from flash crowds by adjusting a threshold. Thus, this method can satisfy the requirement of DFC, but cannot achieve the goals of SD, DT and PI.

The PING command of ICMP protocol is used to test whether a host is reachable. When a source host sends a PING request to a living host, that host must answer with a PING reply. A PING flooding attack exploits this weakness to overwhelm a host with mass PING requests. Yadav *et al.* [64] proposed a distributed defense approach to mitigate the PING flooding attack by inspecting whether the number of PING request packets crosses a threshold or whether the size of PING request packets is bigger than the size of normal PING packet. When a router, which connects with a victim, has detected a PING flooding attack, it will issue an alert message to intermediate routers. Those routers will start dropping the PING packets whose destination address is the victim's IP address. In daily Internet activities, the PING command is often blocked by firewall. However, the method cannot support all additional requirements.

b) Use flow-level data: Sanmorino and Yazid [65] proposed a protocol independent detection method that uses flow patterns. All incoming packets with identical 5 tuple (viz. source and destination addresses, source and destination ports, and protocol type) are aggregated into flows. They used flow size and flow count to calculate the average flow size in a time interval. If the average flow size is very small while flow count is large, then a flooding attack is happening at that time. For example, during SYN flooding attacks, an attacker employs massive bots to overwhelm a victim. So the flow count will drastically increase due to the increase of the number of new source IP address. The method can satisfy the requirement of PI, but cannot achieve the goals of SD, DT and DFC.

Miao *et al.* [66] introduced eight scenarios of SYN flooding attacks according to the positions of attackers, victims and attacking addresses. Then they used Netflow data to detect Internet-wide SYN flooding attacks based on the symmetry relationship between SYN flows and SYN-ACK flows. But their method cannot support all additional requirements.

c) Use connection-level data: Rahmani *et al.* [67] utilized total variation distance to measure the similarity between inflow size and outflow size in each connection. During normal TCP/UDP/ICMP connections, inflow size and outflow size retain a similar shape. Any abrupt disproportion corresponds to a legitimate or an illegitimate anomaly. Distance measurement is an effective way to discriminate DDoS flooding attacks from flash crowds because the similarity among attack traffic is much higher than the similarity among legitimated traffic of flash crowds. Thus, the method can satisfy the requirements of SD, PI and DFC, but cannot support DT.

4) Discussion: Table V gives a summary of detection methods of N/T-L DDoS flooding attacks. We also provide comparison results of existing detection methods with regard to N/T-L DDoS flooding attacks in terms of the following criteria:

- Self-adaptive Detection:
  -*Yes*: The detection method detects attacks adaptively to the changes of current network behaviors.
  -*No*: The detection method employs a static profile to detect attacks.
- Dynamic Threshold:
  -*Yes*: The detection method applies a dynamic threshold to detect attacks.
  -*No*: The detection method uses a static threshold to detect attacks.
- Protocol Independent:
  -*Yes*: The detection method has the ability to detect a category of attacks independently from a concrete protocol.
  -*No*: The detection method can only detect a specific protocol-related attack.
- Deal with Flash Crowds:
  -*Yes*: The problem of flash crowds was considered and solved in the paper.
  -*No*: The problem of flash crowds was not considered or solved in the paper.

From the table, we can observe that the number of packets and flow count are widely used to detect N/T-L DDoS flooding attacks, especially in knowledge-based detection methods [63]–[67]. This is inspired by the nature of the flooding attacks that flood a target host with massive packets, which is obvious in the network/transport layer. Current DDoS flooding attacks are often launched by a botnet. Each bot, which is infected by the same malicious program, generates attack packets in the same format. The attack packets share many similar characteristics, such as destination address, source port, destination port, packet size, packet rate. These identical characteristics at the packet-level lead to similarities on flow-level (such as flow rate, flow size, flow duration) and connection-level (such as connection size, connection duration). Therefore, generating a profile that measures the distribution of traffic features (such as source and destination addresses, source and destination ports, packet size, packet rate, flow duration, etc.) in statistical methods is quite necessary [45]–[48], [51], [52]. Machine learning-based detection methods often select some important characteristics of traffic features that can reflect that the traffic is generated by a botnet, to classify attack traffic [53], [57], [59], [62].

Moreover, the properties of Self-adaptive Detection and Dynamic Threshold are not widely applied in the current literature. Most of the methods do not take the problem of flash crowds into account. This leads to high false rate when flash crowds occur.

B. Detection Methods Against Application Layer DDoS Flooding Attacks

Application Layer DDoS (AL-DDoS) flooding attacks generally focus on exhausting server resources such as sockets, CPU, memory, disk/database bandwidth and I/O bandwidth [68]. The attacker usually customizes them to disrupt a particular Web server. Unlike N/T-L DDoS flooding attacks, AL-DDoS flooding attacks can be launched by using legitimate traffic and the volume of traffic is also not too large. Moreover, during flash crowds, attackers can imitate legitimate user's requests and attack a targeted server. The AL-DDoS flooding attacks become more and more undetectable due to the above obscure activities.

Several famous AL-DDoS flooding attacks are discussed below.

(i) HTTP flooding attacks: The HTTP flooding attacks are one of the most famous AL-DDoS flooding attacks that aim to Web servers. They can be mainly classified into the following categories [6], [68]: (a) HTTP request flooding attacks that send high-rate requests packets, such as HTTP GET/POST flooding attacks. (b) Slow HTTP request attacks (idle attacks) in which an attacker keeps HTTP connection opening in an idle state without actually sending a complete HTTP request, such as slowloris attack, (c) HTTP asymmetric attacks that send high-workload requests. Jiang *et al.* [69] provided valuable insight on the impacts of AL-DDoS flooding attacks on HTTP/1.1 and HTTP/2.0.

(ii) SIP flooding attacks: These attacks can be easily launched by employing open SIP traffic generators. They have some attack forms: (a) SIP INVITE flooding attacks in which a large number of INVITE messages are generated and

TABLE V
SUMMARY AND COMPARISON OF DETECTION METHODS OF N/L-T DDOS FLOODING ATTACKS

References	Collected Security Data	Analysis Method	DA	FA	SD	DT	PI	DFC	Remarks
[45]	The number of packets	Entropy variation	100%	0%	N	N	N	Y	Detect all kinds of TCP SYN flooding attacks
[46]	The number of packets	Hellinger distance variation	93.33%-100%	0%	N	Y	Y	N	An online anomaly detection method
[47]	Source address and destination port	Entropy variation	N.A.	N.A.	N	N	N	N	Using KD-Tree makes detection method scalable
[48]	Packet size, TTL values, source address, TCP flags, server port, etc.	Conditional Legitimate Probability	N.A.	0.03%-5.15%	N	Y	Y	N	Effective in filtering several different attack types
[51]	Packet size in each flow	Entropy variation	N.A.	N.A.	N	N	Y	N	Detection method is stable
[52]	Connection size on flow-level	Entropy variation	N.A.	N.A.	N	Y	N	N	Adaptive threshold makes detection method accurate
[53]	Source address, packet size, the number of packets, etc.	ANN	100%	N.A.	N	N	Y	N	Detect known and unknown attacks
[54]	The number of packets, source address, packet size, packet arrival time etc.	Naive Bayesian classifier	97%-99.5%	0%-2.3%	N	N	N	N	A real-time system to detect DDoS flooding attacks
[55]	Packet size, TCP flags, source port, etc.	Weighted KNN classifier	97.42%	0.33%	N	N	Y	N	Detect known and unknown DDoS attacks in real-time
[56]	The number of packets, source addresses	Classification algorithm	100%	0%	N	N	N	Y	Tested with several classification algorithm
[57]	The number of packets, source address, source port, packet type, etc.	Hierarchical clustering algorithm	N.A.	N.A.	N	N	Y	N	Analyze each phase of DDoS flooding attacks
[59]	Flow size, flow duration, etc.	KNN with correlation analysis	88.4%-96.7%	N.A.	Y	N	Y	N	A grid-based method is used to reduce computational burden
[60]	Netflow records	One-class SVM	89.6%-93.8%	0%-3.3%	N	N	Y	N	A kernel function makes detection method universal
[61]	Flow duration, packet size, source address, etc.	K-means algorithm	100%	N.A.	N	N	Y	N	Applying dynamic threshold makes detection more effective
[62]	Connection count, the number of packets, etc.	Neural classifier	90%-98.5%	2.9%-4%	N	N	N	N	Detection method is suitable for a real time environment
[63]	The number of packets	Number variation	100%	N.A.	N	N	N	Y	Detect all kinds of SYN flooding attacks
[64]	The number of packets, packet size	Match attack rule	N.A.	N.A.	N	N	N	N	A distributed detection method
[65]	Flow size, flow count	Match attack rule	95%	N.A.	N	N	Y	N	A pattern-based method reduces the cost of infrastructure
[66]	Flow count	Number variation	N.A.	N.A.	N	N	N	N	Detection method is real-time
[67]	Connection size on packet-level	Number variation	90%	N.A.	Y	N	Y	Y	Need less parameters than other methods

DA: Detection Accuracy; FA: False Alarm; N.A.: Not Available;
SD: Self-adaptive Detection; DT: Dynamic Threshold; PI: Protocol Independence; DFC: Deal with Flash Crowds; Y: Yes; N: No

transmitted to a victim. (b) SIP BYE flooding attacks in which a large number of BYE messages are sent to a SIP server. (c) SIP REGISTER flooding attacks in which a large number of user agent REGISTER requests are sent to a SIP registrar. (d) Multi-attribute flooding attacks in which a large number of all four SIP messages (INVITE, BYE, RINGING, and OK) are sent to the SIP server.

(iii) DNS flooding attacks: During DNS flooding attacks, some attacker commands bots to send a large volume of malformed DNS queries to a DNS server in order to exhaust its resources. There are two types of DNS flooding attacks [70]: (a) Water Torture in which bots send a large number of DNS queries by adding random subdomains to a prefixed domain. (b) NXDOMAIN in which bots send a large number of queries to non-existent domain names. Another DNS-based DDoS flooding attack is amplification flooding attacks. We will discuss them in Section IV-D.

1) Statistical Methods: In this part, we discuss statistical analytic methods for detecting AL-DDoS flooding attacks using packet-level data, flow-level data and connection-level data.

a) Use packet-level data: Based on the principle of HTTP protocol, Alenezi and Reed [71] proposed a detection method for HTTP flooding attacks. During normal network communications, congestion window (cwnd) value is changing due to congestion control and the change is with an reasonable range. But when a HTTP flooding attack occurs, the victim controls traffic volume by setting the cwnd value at a low value. Instead of building a static normal profile, they employed a Cumulative Sum (CUSUM) algorithm to detect the variation of cwnd values. If the threshold value is exceeded, an alarm is triggered. The method can support SD, but cannot satisfy the requirements of DT, PI and DFC.

Zhou *et al.* [72] proposed an efficient method that can be deployed in network backbone to distinguish AL-DDoS flooding attacks from flash crowds. Their method has three modules: (i) abnormal traffic detection module that issues a specific signal of 'ATTENTION' if abrupt changes in the number of HTTP GET request packets exceed a presumed threshold, (ii) AL-DDoS flooding attack detection module that first derives traffic model from Web traffic by using the number of HTTP GET request packets and the packet arrival time of each incoming source address, and then compares the current traffic model with a normal traffic model to get attack probability, (iii) filter module that drops the network traffic coming from a malicious source address that is determined by the second module. Thus, the method can satisfy the requirement of DFC, but cannot achieve the goals of SD, DT and PI.

Sengar *et al.* [46] applied the same detection method to detect SIP flooding attacks by using the probability distributions of the number of SIP INVITE, SIP 200 OK, SIP ACK and SIP BYE packets. But the above detection method becomes ineffective if the four types of packets are proportionally flooded simultaneously. Tang *et al.* [73] developed a versatile detection method for the SIP flooding attacks. They designed a three-dimensional sketch data structure to separately summarize the number of SIP INVITE, SIP 200 OK, SIP ACK and SIP BYE packets. Based on sketch data structure, a probability distribution is established for each SIP attribute independently. A SIP flooding attack can be detected with a high probability by comparing the Hellinger distance among data distributions in sketches with a dynamic threshold that is calculated with Exponential Weighted Moving Average (EWMA) algorithm. Thus, their method can satisfy the requirement of DT, but cannot achieve the goals of SD, DT and DFC.

Wang *et al.* [74] proposed an effective detection and defense system for AL-DDOS flooding attacks, named SkyShield, which monitors the divergence of distribution of packet amount recorded by sketch. SkyShield employs three sketches (S_1, S_2, S_3) and two Bloom filters (B_1, B_2). In each detection cycle, S_1 is used to record information of all incoming requests with source addresses as input keys. S_2 stores the results of S_1 in the last normal detection cycle. At the end of each cycle, the divergence between S_1 and S_2 is calculated and compared with a threshold that is computed by a Multiple Independent EWMA algorithm. If an anomaly is detected, S_2 will not be updated anymore that ensures the detection method is self-adaptive and S_3 will be updated by S_1. In mitigation phase, all incoming traffic will be checked by B_1 that is a whitelist and B_2 that represents a blacklist. The remaining requests are inspected based on abnormal buckets of S_3. SkyShield achieves high performance in attack detection and mitigation even when the attacks occur during flash crowds. Thus, SkyShield can satisfy all additional requirements of SD, DT, PI and DFC.

Thapngam *et al.* [75] used packet arrival rate as input to extract pattern behavior of attack packets. They observed that the transmission rate of attack packets has the property of repeatability and can be considered as a pattern in a short period of time. Oppositely, the packet arrival rate of normal traffic is non-predictable because the users may take time to skim and respond. They employed Pearson's correlation coefficient to judge the similarity among traffic to discriminate attack traffic from the traffic generated by real users like flash crowds. Thus, their method can achieve the goals of SD, PI and DFC, but cannot support DT.

Yu *et al.* [76] first acted as attackers to mimic browsing behaviors of a legitimate Web viewer in order to study the characteristics of mimicking attacks. They found that mimicking attacks can be successfully launched if the attacker has a sufficient number of active bots. However, sufficient number of bots is hard to fully gather in some time, which gives possible to detect the flash crowd mimicking attacks. They established a map of the number of page request for a 24-hour period and based on it, a mapping of the variation of fine correntropy that is a tool for second order similarity measurement of page requests was built. The flash crowd mimicking attacks can be accurately detected by comparing current mappings with the pre-established mappings. The method can satisfy the requirement of DFC, but cannot achieve the goals of SD, DT and PI.

Bhatia [77] presented an ensemble-based DDoS flooding attack detection method that has the ability of separating flash crowd traffic. They combined packet-level data and host-level data per interval to construct traffic feature vector, namely the number of packets, the number of new source addresses, the number of source addresses, the number of packets per IP, CPU utilization, CPU load, memory utilization. Exponentially, EWMA algorithm was employed to analyze traffic feature vector in two adjacent time intervals to detect attacks. Thus, the method can support SD and DFC, but cannot satisfy the requirements of DT and PI.

Alonso *et al.* [70] proposed a novel notion called DNS social structure by abstracting recursive DNS traffic from recursive DNS servers. They observed a phenomenon that DNS usage gives rise to a social structure through mining the interactions of IP address to domain names. A normal DNS social structure will be built at DNS server-side based on the number of distinct source (agent) IP addresses, the number of DNS query packets and other calculated features. During DNS flooding attacks, the normal DNS social structure is drastically changed, indicating the presence of attack. But their method cannot support all additional requirements.

b) Use flow-level data: Yu *et al.* [78] applied a similarity metric, called flow correlation coefficient to discriminate attack flows among suspicious flows (such as flow crowds). In their method, each flow is made up of incoming packets that share the same destination address. They formulated the flow correlation coefficient by using the basic elements of flow size. When a possible attack alarm goes off, the flow correlation coefficient between suspicious flows is calculated. If the correlation coefficient exceeds a threshold, this pair of flows is determined as attack. The detection method is self-adaptive and protocol independent because it makes full use of the similarity among flows. Thus, their method can satisfy the requirements of SD, PI and DFC, but cannot support DT. Saravanan *et al.* [79] also proposed using flow similarity to detect AL-DDoS flooding attacks during flash crowd.

Giralte *et al.* [80] described a normal user behavior in a statistical way. They aggregated packets with the same source address and protocol type. Then the flow count, flow size and flow rate of HTTP protocol are used to compute statistics for each user (each source address). If a high deviation between a user value and the average value is reported, this user is labeled as suspicious. However, their method cannot support all additional requirements.

Zolotukhin *et al.* [81] defined that a conversation is the combination of two flows such that the source socket of one of these flows is equal to the destination socket of another flow and vice versa. They recorded conversation duration (that is flow duration) and extracted following features from inflow and outflow in each TCP conversation: packet rate (per second); the number of transmit bytes per second; maximal, minimal and average of packet size; maximal, minimal and average of TCP window values; maximal, minimal and average of TTL values; and percentage of packets with different TCP flags. During offline training, a normal user behavior model is built with the help of fuzzy clustering based on these features. Then, they applied an online training algorithm to rebuild the model every time when a new portion of network traffic is available for the analysis. Reconstruction error between the old model and the new one is calculated. If the error value exceeds a predefined threshold, this conversation is considered to be an intrusion. However, this method cannot support all additional requirements.

c) Use connection-level data: Beitollahi and Deconinck [82] measured various statistical attributes for users in non-attack conditions, including the distribution of source IP addresses, request packet rates, the number of transmit bytes per second (downloading rat), connection duration, browsing behaviors that extracted from Web operation logs, arrival distribution rate of users (source addresses), and well-known confirmed users. They used these features to build a normal user profile. For each connection, a score is assigned based on normal user profile. The connections that get lower scores are more possible to be intrusions. The method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

2) Machine Learning: In this part, we discuss applying machine learning to detect AL-DDoS flooding attacks using packet-level data and connection-level data.

a) Use packet-level data: Adi *et al.* [83] designed a stealthy DDoS flooding attack that has the ability of directly attacking HTTP/2 Web servers. They showed this type of attack can degrade the performance of machine learning analysis. To prevent this novel attack, the authors presented a detection scheme that combines feature selection techniques and supervised learning methods. Three features are extracted from legitimate and malicious traffic, namely the number of packets grouped by packet type, the total number of bytes grouped by packet type and the packet arrival time. These features are ranked and classified by supervised learning methods. But the method cannot support all additional requirements.

Ramamoorthi *et al.* [84] first built normal user's behavior profile using HTTP request rate, page viewing time, the number of TCP and UDP and ICMP packets, etc. These features of behavior profile are used as training samples for an Enhanced Support Vector Machines (ESVMs) with string kernels. The experimental results show that this ESVM with string kernels has a higher accuracy than one class SVMs, Binary SVMs and SVMs with string kernels. But their method cannot support all additional requirements.

She *et al.* [85] introduced a clustering-based detection method for HTTP flooding attacks. They first defined that if the time interval between two packets that arrive continuously less than a pre-set threshold, then these two packets are in the same session. According to this definition, they extracted following features from a session: the total number of requests in a session, the total size of all request packets in a session, the request packet rate of a session and the average frequency of the request packets in a session. The K-means algorithm was used to cluster traffic and a normal user's behavior model was obtained from clusters. A new session is whether normal or not is determined by its distance from the normal model. But the method cannot support all additional requirements.

b) Use connection-level data: Singh *et al.* [86] analyzed the standard Environmental Protection Agency-Hypertext Transfer Protocol (EPA-HTTP) dataset and extracted some essential features, namely the entropy of the number of HTTP GET request packets per connection, the number of HTTP GET request packets for a particular IP address, and the variance of the entropy per IP address. These features were considered as the input to a multilayer perceptron classification algorithm with a genetic algorithm for differentiating attacks from a normal profile. Their method not only shows that the entropy values of the number of HTTP GET request packets and IP addresses decrease in case of an attack, but also proves that the nature of packet transmission from an attack source is almost identical. Their method cannot support all additional requirements.

3) Knowledge-Based Methods: In this part, we discuss knowledge-based analytic methods for detecting AL-DDoS flooding attacks using packet-level data and flow-level data.

a) Use packet-level data: Geneiatakis *et al.* [87] utilized the one-to-one mapping relationship among INVITE-responses-ACKs to detect SIP flooding attacks. Three bloom filters with counters are respectively employed to keep track of the INVITE requests and the corresponding responses and the final ACKs. A new metric calculated based on the counters, named "session distance", is introduced as a detection measurement against such attacks. The smaller the value of session distance, the more legitimate traffic is. They calculated a judging dynamic threshold by considering the average value of the session distance, network delay and user response time. Experimental results showed that detection time is negligible and the rate of false alarms is very low if applying such a detection method. The method can satisfy the requirements of SD and DT, but cannot support PI and DFC.

Liao *et al.* [88] analyzed the differentiation between Web user's behaviors and then extracted the following two feature sequences to represent browsing behaviors: the sequence of request frequency and the sequence of request interval. The detection architecture consists of two parts: (i) Request interval sequence part. This part filters normal users based on the assumption that they take more time in Web browsing than malicious users that have a short time interval between two adjacent request packets. (ii) Request frequency sequence part. This part further filters normal users from malicious

TABLE VI
SUMMARY AND COMPARISON OF DETECTION METHODS OF AL-DDOS FLOODING ATTACKS

References	Collected Security Data	Analysis Method	DA	FA	SD	DT	PI	DFC	Remarks
[71]	TCP window size	CUSUM algorithm	N.A.	0%	Y	N	N	N	Select a novel traffic feature
[72]	The number of packets, packet arrival time	Model similarity mining	N.A.	N.A.	N	N	N	Y	The method can be deployed for handling heavy backbone traffic
[73]	The number of packets	Sketch with Hellinger distance metric	100%	1.67%/ 7.81%	N	Y	N	N	Sketch technology makes detection method scalable
[74]	The number of packets	Bloom filter and Sketch Hellinger distance metric	100%	1.77%- 3.67%	Y	Y	Y	Y	Test the impacts of all parameters
[75]	Packet arrival time	The similarity of packet rate	85%/ 94%	0%	Y	N	Y	Y	Test the performance of detection method with three datasets
[76]	The number of packets	Distance metric	N.A.	N.A.	N	N	N	Y	Imitate attackers to study the characteristics of attacks
[77]	The number of packets, source address, CPU and memory utilization, etc.	EWMA change detection	N.A.	N.A.	Y	N	N	Y	Combine packet-level data and host -level data to detect attacks
[70]	The number of packets, source address	DNS social structure	N.A.	0.2%	N	N	N	N	The first work of detecting DNS flooding attacks on a recursive level
[78]	Flow size	Flow similarity	N.A.	N.A.	Y	N	Y	Y	Find a new feature to discriminate DDoS attacks
[80]	Flow size, flow count, flow rate	Analysis of model change	N.A.	N.A.	N	N	N	N	Combine several technologies to model web-server behaviors
[81]	Flow duration, packet rate, the number of packets, TTL, etc.	Analysis of model change	100%	Around 0.5%	N	N	N	N	Have the ability to detect attacks that utilize encrypted protocols
[82]	Connection duration, source address, packet rate, etc.	ConnectionScore algorithm	N.A.	0%	N	N	Y	N	Effectively detect common and meek attacks
[83]	The number of packets, the number of transmitting bytes, packet arrival time	Classification algorithm	N.A.	0%- 0.27%	N	N	N	N	Present a novel DDoS attack for HTTP/2 services
[84]	The number of packets, packet rate, packet arrival time, etc.	Enhanced SVM algorithm	99.32%	N.A.	N	N	N	N	Enhanced SVM with string kernels has high detection accuracy
[85]	The number of packets, packet size, packet rate	K-means algorithm	75.6%- 98.37%	0.19%- 5.17%	N	N	N	N	A simple but robust method
[86]	The number of packets, source address	Multilayer Perceptron	Around 98.32%	Around 0.5%	N	N	N	N	Analyze the impacts of DDoS attacks
[87]	The number of packets	Bloom filter with counters	N.A.	0%	Y	Y	N	N	Evaluate through different scenarios
[88]	The number of packets, packet arrival time	Match attack rule	Around 90.11%	Around 3.5%	N	N	N	N	Employ a stepwise refinement method to filter traffic
[89]	Meta-information	D-S evidential theory	N.A.	4.5%	Y	N	N	N	Use meta-information to aggregate packets
DA: Detection Accuracy; FA: False Alarm; N.A.: Not Available; SD: Self-adaptive Detection; DT: Dynamic Threshold; PI: Protocol Independence; DFC: Deal with Flash Crowds; Y: Yes; N: No									

users by using a decomposition and rhythm matching algorithm. The remaining sequences are inspected based on a predefined threshold. But the method cannot support all additional requirements.

b) Use flow-level data: Zhang *et al.* [89] proposed a novel packets aggregation method that uses meta-information and employs resulting flows to detect AL-DDoS flooding attacks. They used two surface features from packets of a single user (source address): average scan time and sequence of page requests. Average scan time is, in fact, the average of inter-arrival time of request packets from the same source address. Sequence of page requests represents the access times of a specific page namely the number of request packets of that page. Then they aggregated users' packets with similar average scan time and sequence of page requests to one flow. By adding up all users' sequence of page requests and calculating page's access frequency to generate a normalized frequency vector, hot-spot access is extracted from flows. Finally, they applied Dempster-Shafer (D-S) evidence theory to analyze flow rate to evaluate a flow and assess whether it is an attack flow or not. Thus, their method can support SD, but cannot satisfy the requirements of DT, PI and DFC.

4) Discussion: Having the same format as in Table V, we summarize and compare detection methods of AL-DDoS flooding attacks in Table VI.

From the table, we can see that the number of packets is still an important traffic feature to detect AL-DDoS flooding attacks. In statistical methods, measure the similarity among traffic [72], [75], [78] and monitor the changes of user behaviors [70], [71], [77], [80]–[82] are widely employed to detect AL-DDoS flooding attacks, rather than using information entropy to reveal the distribution of traffic features.

This is due to the obscure characteristics of AL-DDoS flooding attacks. The detection methods that combine with traffic compression and fusion technique [73], [74] for AL-DDoS flooding attacks obtain high performance, which are capable of dealing with large-scale network traffic. The usages of machine learning and knowledge-based methods in detecting AL-DDoS flooding attacks are as similar as the usages in detecting N/T-L flooding attacks. Although both flash crowds and AL-DDoS flooding attacks are gusty and have a high volume in network traffic. There still exist some key differences between them, such as the distribution of source addresses, the speed of traffic, the volume of traffic, traffic arrival time, etc. All these features can be used to discriminate AL-DDoS flooding attacks from flash crowds by measuring the similarity among the traffic [72], [74]–[78]. We also find that the requirements on dynamic threshold and protocol independence are not widely considered in the current detection methods.

C. Detection Methods Against Low-Rate DDoS Flooding Attacks

Low-rate DDoS (LDDoS) flooding attacks send a large number of packets within a specific time interval to decline the performance of a victim's services. They follow the form of periodic pulse (ON/OFF pattern), as shown in Fig. 5. LDDoS flooding attacks are totally different from traditional high-rate DDoS flooding attacks. First, they exploit many vulnerabilities of a target system, e.g., shrew attacks use TCP Retransmission Time-Out (RTO) mechanism [90], NewShrew attacks use both TCP RTO mechanism and slow start mechanism [91], low-rate DoS attacks against concurrent servers (LoRDAS) use the service response mechanism of application layer [92], and Reduction of Quality (RoQ) attacks exploit a common adaptation mechanism [93]. Second, LDDoS flooding attacks send attack packets only in a specific interval and do nothing in other time-slots. This characteristic of LDDoS flooding attacks hides abnormal traffic as legitimate traffic by keeping a low average rate. Obviously, it is difficult to use the detection methods for high-rate DDoS flooding attacks to detect the LDDoS flooding attacks.

1) Statistical Methods: In this part, we discuss statistical analytic methods for detecting LDDoS flooding attacks using packet-level data and flow-level data.

a) Use packet-level data: Hoque *et al.* [94] used the entropy of source addresses, the change rate of IP addresses and packet rate to build a normal profile under non-attack traffic. A dissimilarity value is calculated based on the deviation between normal profile and current profile. If the dissimilarity value is greater than a predefined threshold, an alarm is generated. The detection method is protocol independent because it exploits the basic characteristics of LDDoS. They also proposed a method to classify low-rate and high-rate DDoS flooding attacks. Their method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

Xiang *et al.* [95] proposed a detection method for LDDoS flooding attacks by using two new information metrics namely generalized entropy metric and information distance metric. The generalized information entropy as a generalization of Shannon entropy is one of a family of functions for quantifying either the diversity uncertainty or randomness of a system. In their method, the distribution of source IP addresses or the distribution of IP packet sizes can be used. The information distance they designed can measure the deviation of probability distributions between attack traffic and normal traffic. By adjusting the value of parameters of the generalized entropy and information distance metrics, the method can increase the information distance between LDDoS flooding attack traffic and normal traffic. The detection method is protocol independent because it exploits the basic characteristics of LDDoS. It can support PI, but cannot satisfy the requirements of SD, DT and DFC.

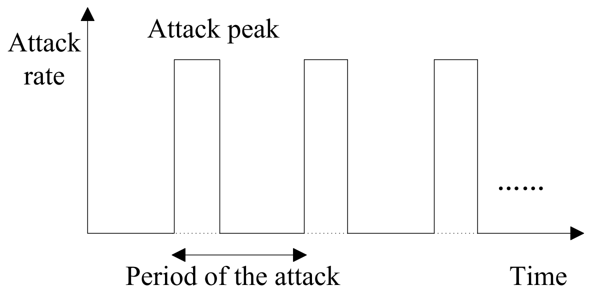


Fig. 5. LDDoS flooding attacks.

Network traffic exhibits self-similarity over a large time scale while presenting multifractal characteristics over a small time scale. Based on this idea, Wu *et al.* [96] proposed a novel detection method for LDDoS flooding attacks using a multifractal technology. When LDDoS flooding attacks are launched, a large number of UDP packets will appear in the network because the LDDoS flooding attacks usually use UDP protocol. This status makes the Lipschitz-Holder exponent (Holder exponent) that is used to characterize the bursty of network traffic, fall quickly. The smaller the Holder exponent is, the more bursty the network traffic. Therefore, by monitoring the abrupt change of the Holder exponent in adjacent time periods through wavelet analysis, LDDoS flooding attacks can be detected and considered to exist if the deviation is larger than a detection threshold. Their method can satisfy the requirement of SD, but cannot achieve the goals of DT, PI and DFC. Based on the multifractal characteristics exist in network traffic, another LDDoS attack detection work based on wavelet transform and neural network is described in [97].

b) Use flow-level data: It is obvious that a periodic pulse is difficult to be detected by using existing methods of analyzing network traffic in time domain, because its average share of bandwidth is not big enough. Wu *et al.* [98] used the theory of Digital Signal Processing (DSP) based on a small signal model to detect TCP-based LDDoS flooding attacks. They aggregated packets into flows by protocols and considered the variation of flow size in adjacent time periods as the sign of attack. The method can support SD, but cannot satisfy the requirements of DT, PI and DFC.

Zhou *et al.* [99] proposed an Expectation of Packet Size (EPS)-based method to distinguish LDDoS attacks from legitimate traffic. They classified packets that share the same

TABLE VII
SUMMARY AND COMPARISON OF DETECTION METHODS OF LDDOS FLOODING ATTACKS

References	Collected Security Data	Analysis Method	DA	FA	SD	DT	PI	DFC	Remarks
[94]	Source address, packet rate	Distance variation	100%	0%	N	N	Y	N	The methods can detect both low-rate and high-rate DDoS attacks
[95]	Source address, packet size	Entropy and distance metric	N.A.	N.A.	N	N	Y	N	The methods has high traceback accuracy
[96]	The number of packets	Multifractal detrended fluctuation analysis	91%/ 92%	9%/ 10%	Y	N	N	N	Exploit the changes in multifractal characteristics of network traffic
[98]	Flow size	Small signal model	100%	0%	Y	N	N	N	Find out the attributes of LDDoS attacks in frequency domain
[99]	Flow size and packet size in each flow	Expectation variation	98.7%	N.A.	N	N	Y	N	Simple but effective method
[100]	Flow rate	Match attack rule	N.A.	N.A.	Y	N	Y	N	Ignore the characteristics of LDDoS attacks
[101]	Flow size	Congestion Participation Rate	100%	1.625%	Y	N	Y	N	Simple calculation and wide deployment

DA: Detection Accuracy; FA: False Alarm; N.A.: Not Available;
SD: Self-adaptive Detection; DT: Dynamic Threshold; PI: Protocol Independence; DFC: Deal with Flash Crowds; Y: Yes; N: No

destination address into flows and calculated EPS value of each flow at different time. The EPS value of attack traffic is smaller than that of normal traffic and varies within a narrow limit. Motivated by this difference, attack flows can be distinguished by comparing the EPS values among flows. This detection method is protocol independent since it exploits the basic characteristics of LDDoS. Thus, the method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

2) Knowledge-Based Methods: In this part, we discuss knowledge-based analytic methods for detecting LDDoS flooding attacks using flow-level data.

Use flow-level data: During the congestion time caused by LDDoS flooding attacks, the rate of attack flows will drastically increase while normal flow rate will decrease due to the congestion control mechanism. But in a non-attack period, the rate of attack flows is close to zero and normal flow rate will increase. Wu *et al.* [100] utilized the difference between the rate of normal flows and attack flows in pulse periods to fight against LDDoS flooding attacks. They aggregated incoming packets by source and destination addresses and recorded each flow rate in a time interval. When packet loss rate is abnormal, the average rate of each flow is calculated. If flow rate is beyond a predefine threshold, this flow is treated as a malicious flow. This detection method exploits the basic characteristics of LDDoS and does not need to prebuild a normal profile. Thus, it can satisfy the requirements of SD and PI, but cannot support DT and DFC.

Zhang *et al.* [101] proposed a novel metric called Congestion Participation Rate (CPR) deployed in a router to detect LDDoS flooding attacks. All incoming packets with identical five-tuple keys (viz. source and destination addresses, source and destination ports and protocol type) are aggregated into flows. The CPR of a given flow is defined as the ratio of the flow size in congestion time to the flow size in normal time. During attacks, the CPR values of attack flows substantially increase and become higher than a threshold. Through monitoring the difference of CPR values of network flows, their method can effectively identify LDDoS flooding attack flows. The detection method exploits the basic characteristics of LDDoS and does not need to prebuild a normal profile. Thus, this method can support SD and PI, but cannot achieve the goals of DT and DFC.

3) Discussion: Using the same format as in Table V, we summarize and compare detection methods of LDDoS flooding attacks in Table VII.

From the table, we can observe that detection methods for LDDoS flooding attacks are different from the detection methods for the first two types of attacks. As opposed to the high-rate DDoS flooding attacks, LDDoS flooding attacks inject a short burst of traffic periodically to occupy the resources of victim. These attacks are difficult to detect and defense, as most of the detection methods for DDoS flooding attacks are triggered by high-rate and high-volume traffic. The most efficient way of detecting LDDoS flooding attacks is to find the periodicity and abruptness of the changes in packet amount, which is analyzed in frequency domain [96], [98]. Another way is to find the similar characteristics (such as packet size, packet rate, etc.) of packets generated by the same botnet to detect LDDoS flooding attacks [94], [95], [99]. Also, utilizing the distinct behaviors of LDDoS flooding attacks is an effective detection method [100], [101]. Obviously, most of the detection methods for LDDoS flooding attacks are self-adaptive to the changes of traffic [96], [98], [100], [101] and protocol independent [94], [95], [99]–[101]. These methods use the basic characteristics of the attacks and are very robust. However, the adoption of dynamic threshold and the problem of flash crowds are not considered in the above reviewed works.

D. Detection Methods Against Reflection Amplification DDoS Flooding Attacks

In Distributed Reflection DoS (DRDoS) flooding attacks, as shown in Fig. 6, an attacker employs bots to send massive request packets with a spoofed source address (a victim's address) to reflectors. The reflectors reply corresponding response packets to the victim. As a result, the victim receives a lot of response packets and its system resources will be drastically consumed. Usually, the size/number of response packets is many times larger than the size/number of request packets, so we call this type of DRDoS flooding attacks as reflection amplification DDoS flooding attacks (referred to as amplification DDoS flooding attacks). There are two reasons for an attacker to launch DRDoS flooding attacks: (i) anonymity: an attacker can hide its location with a spoofed source IP address; (ii) amplification: an attacker can amplify the impact of attacks by exploiting botnet and unsymmetrical size of response packets.

The impact of DRDoS flooding attacks can be measured by two amplification factors: Packet Amplification Factor (PAF) which is the ratio of the number of response packets to the number of request packets and Bandwidth Amplification Factor (BAF) which is the ratio of the payload size of response packets to the payload size of request packets. Rossow [102] has evaluated 14 UDP-based protocols, which are vulnerable to be exploited to launch amplification attacks. They showed that UDP-based amplification flooding attacks usually have much higher amplification factors. Moreover, Kuhrer *et al.* [103] demonstrated that TCP three-way handshake also has amplification potential and the amplification factors are around 20.

Detection methods of amplification DDoS flooding attacks can be classified into two methods: reflector-end and victim-end. At the reflector-end, the key techniques against amplification flooding attacks are the detection of spoofed address. But there are two drawbacks of reflector-end detection methods: (i) there exist many potential reflectors, and (ii) illegitimate incoming requests might look the same as legitimate requests in reflectors. At the victim-end, typical detection methods depend on analyzing the unbalance relationship between outgoing request packets and incoming response packets or utilizing attack attributes of DDoS flooding attacks. Ryba *et al.* [104] discussed that the victim-end detection methods may be ineffective. The reason is that after attacks have occurred, the victim's bandwidth may already have become saturated or the volume of traffic is too high for the victim to process. In the next two subsections, we review current detection methods of DRDoS flooding attacks from the perspective of security data.

1) Statistical Methods: In this part, we discuss statistical analytic methods for detecting DRDoS flooding attacks using flow-level data.

Use flow-level data: Wei *et al.* [105] found that the rate of responsive flows from reflectors has linear relationship with each other because they are simulated by the same attacking traffic. They defined all packets to a victim through one router as a flow. Spearman's rank correlation coefficient was employed to calculate the flow correlation coefficient between flow pairs to differentiate attack traffic from legitimate traffic. Their method exploits the similar characteristics of DRDoS flooding attack traffic. Thus, the method can support SD and PI, but cannot satisfy the requirements of DT and DFC. Similar work was proposed in [106].

2) Machine Learning: In this part, we discuss machine learning for detecting DRDoS flooding attacks using packet-level data.

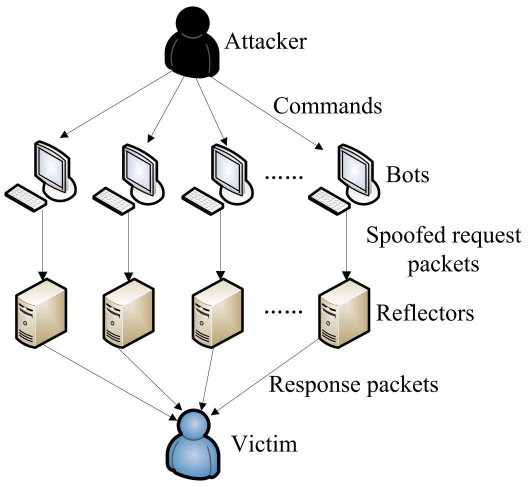


Fig. 6. DRDoS flooding attacks.

Use packet-level data: Gao *et al.* [107] found that the following five features largely change in a time unit during DRDoS flooding attacks: (i) the number of packets that only contain IP header without TCP or UDP header, (ii) the sum sizes of the UDP packets sent to the victim, (iii) the number of packets sent to the victim, (iv) quantity variance of packets sent to and received by the victim, and (v) the maximum number of the packets sent to the victim among all ports. Based on the variation of these features, a normalized polynomial kernel based SVM algorithm is applied to decide whether the data in transiting is attack traffic. The method cannot satisfy all additional requirements.

Meitei *et al.* [108] selected several features from DNS query traffic, such as the mean of inter packet arrival time from same IP, probability of occurrence of an IP per 15 seconds, minimum, average and maximum packet size, etc. They classified the DNS traffic into normal and abnormal by employing Decision Tree, Multilayer Perceptron, Naive Bayes and SVM. Attribute selection algorithms such as Information Gain, Gain Ratio and Chi Square are used to reduce the redundant features. The experimental results show that Decision Tree achieves the highest accuracy of 99.3%. However, their method cannot support all additional requirements.

Attack packets that are generated by the same botnet have a high similarity on packet-level data. Cai *et al.* [109] proposed a reflector-end detection method for DNS amplification flooding attacks. They extracted three features per unit time, including the number of DNS request packets, the ratio of DNS response packet size to DNS request packet size and the ratio of the number of DNS response packets to the number of DNS request packets. Then, they used k-means algorithm to classify normal and abnormal clusters to make a detection pattern and calculate reference points. Based on the detection pattern, new data is whether normal or not is determined by its distance from the reference point of each group. But the method cannot support all additional requirements.

3) Knowledge-Based Methods: In this part, we discuss knowledge-based analytic methods for detecting DRDoS flooding attacks using packet-level data and flow-level data.

TABLE VIII
SUMMARY AND COMPARISON OF DETECTION METHODS OF DRDOS FLOODING ATTACKS

References	Collected Security Data	Analysis Method	DA	FA	SD	DT	PI	DFC	Remarks
[105]	Flow rate	The similarity of packet rate	N.A.	0.1%	Y	N	Y	N	Detect two typical scenarios of DRDoS attacks
[107]	The number of packets, packet size	SVM algorithm	89.74%-100%	0%-8.33%	N	N	N	N	Detect DRDoS attacks under a certain assumed conditions
[108]	Packet size, packet arrival time, IP occurrence rate, etc.	Classification algorithm	94.7%-98.9%	N.A.	N	N	N	N	Analyze only the DNS query traffic.
[109]	The number of packets, packet size	K-means algorithm	N.A.	N.A.	N	N	N	N	Specify attack pattern using three features
[110]	Packet size, TTL values, and the number of packets	Match attack rule	N.A.	N.A.	N	N	Y	N	Use some additional features to detect DRDoS attacks
[111]	The number of request and response packets	Confirmation mechanism	Around 91%	Around 2.6%	Y	N	Y	N	A simple but robust detection method
[112]	Source and destination address, and source and destination port	Match attack rule	N.A.	N.A.	Y	N	N	N	Detect DNS amplification attacks at a local DNS server
[113]	The number of packets	Match attack rule	96%	N.A.	N	N	N	N	Analyze several scenarios of DRDoS attacks

DA: Detection Accuracy; FA: False Alarm; N.A.: Not Available;
SD: Self-adaptive Detection; DT: Dynamic Threshold; PI: Protocol Independence; DFC: Deal with Flash Crowds; Y: Yes; N: No

a) Use packet-level data: If reflectors are able to detect whether the source address of request packets has been spoofed, it would be very easy to detect and prevent amplification flooding attacks. Through analyzing two large datasets, Rudman and Irwin [23] showed that the variation of source address and TTL values are useful for detecting NTP amplification attack. If the packets from the same source address have an obvious gap among TTL values, this source address is an attack target. The reason is attack sources are commonly distributed in different subnets, so the final TTL values of packets sent to the same targeted address must be different.

Böttger *et al.* [110] identified following additional features to characterize DRDoS flooding attacks and used them to distinguish legitimate requests from spoofed attack requests: request and response packet size and payload similarity; the number of ICMP port unreachable messages; TTL values. They showed that a protocol-agnostic approach makes detection process effective to defend not only the attacks on static port numbers, but also novel DRDoS flooding attacks. The method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

Tsunoda *et al.* [111] proposed a simple but robust method to detect DRDoS. They summarized the types of response packets of TCP/IP protocols and their corresponding request types. Hence, the proposed method used a confirmation mechanism to confirm the validity of received response packets based on a request-response relationship. The method can support SD and PI, but cannot achieve the goals of DT and DFC.

Kambourakis *et al.* [112] extracted source/destination IP address and source/destination port number from DNS request packets to build a DNS request table with four columns. A DNS responses table that has the same structure as the request table is built based on the features extracted from DNS response packets that a client receives. Through comparing these two tables, a response packet is marked as suspicious if the features of the response packet do not match the request packets previously sent in a given period. As soon as the number of suspicious packets exceeds a given threshold, an alert is generated. This method can only satisfy the requirement of SD.

b) Use flow-level data: Huistra [113] stated that DNS amplification flooding attacks can be detected by observing the size and the number of DNS packets at reflector-end. He aggregated packets with Netflow standard. For each DNS request inflow, flow size is compared with a predefined threshold to determine whether this flow is suspicious. For each DNS response outflow, flow size is dealt with the same process and furthermore the average size of the response packets is also taken into consideration as a second metric. The method cannot satisfy all additional requirements.

4) Discussion: Having the same format as in Table V, we summarize and compare detection methods of DRDoS flooding attacks in Table VIII.

For DRDoS flooding attacks, existing methods usually use legitimate request packets to deceive reflectors so as not to generate anomalies in network traffic. But it is also efficient to detect them by utilizing similar characteristics of attack traffic [105], [107]–[109]. Another way for defending DRDoS flooding attacks is to monitor the deviation between the statistical information of outgoing traffic (request traffic) and the statistical information of incoming traffic (response traffic) [110]–[113]. However, the application of dynamic threshold and the problem of flash crowds are not considered in these methods.

E. Detection Methods Against Link Flooding Attacks

Link flooding attacks (also called crossfire attacks [114]) are launched by a botnet to cut off specific links in the Internet and to make a specific region disconnected from others. Fig. 7 shows an attack model of link flooding attacks [115]. First, an attacker elaborately selects a target area and decoy servers. The decoy servers must be located around the target area. A map of network topology about the paths from the bots to target servers is created by employing bots to send traceroute commands to target servers. When all bots complete the work of

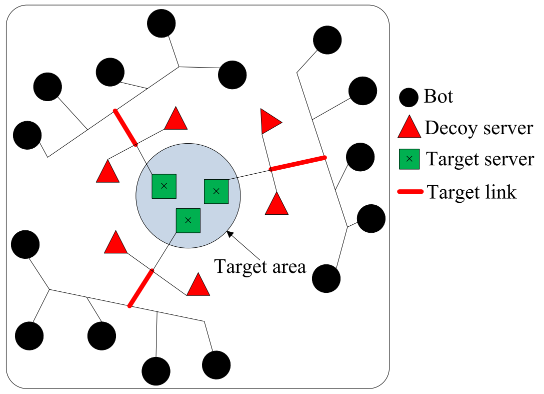


Fig. 7. Overview of the link flooding attack.

finding the paths, the attacker aggregates all traceroute results and finds target links that frequently appear in the results. Then, the attacker selects bot-decoy pairs. A suitable bot-decoy satisfies the condition that traceroute commands from the bot to the decoy server contain the target links. Once target links and bot-decoy pairs are selected, the attacker blocks target links by commanding bots to send low-rate traffic to decoy servers. As a result, the target area is isolated from other regions. Herein, the purpose of introducing link flooding attacks is to arouse researchers' attention. This type of attack has serious attack potential although it needs an elaborate design.

Through analyzing the generation process of link flooding attacks, Hirayama *et al.* [115] found that the number of traceroute packets that are used to find target links between bots and target/decoy servers increases before the attack occurs. They employed a detection server to count the number of traceroute packets at routers. When the cumulative sum is higher than a predefined threshold, an alarm will be triggered. Hirayama's detection method is the first study that can detect link flooding attacks before link congestion occurs.

Xue *et al.* [116] proposed a novel system called LinkScope to detect link flooding attacks and locate target links or area. As link flooding attacks congest the links around a target area, following anomalies could happen on these links: (i) packet loss rate, connection failure rate and Round-Trip Time (RTT) could drastically increase, (ii) available bandwidth could decrease, and (iii) packet reordering could increase on related routers. LinkScope contains three types of probing patterns to measure these anomalies caused by link flooding attacks. The first is Round Trip Probing (RTP), mainly used to measure packet loss rate, RTT and packet reordering. The second is Extended Two Way Probing (eTWP) that is used to measure available bandwidth. The third one is Modified Recursive Packet Train (mRPT) that is used to locate the target links or the target area by combining hop-by-hop and end-to-end measurement. Finally, LinkScope correlates the traceroute data and the measurement data to infer the target links or the target area.

Rerouting traffic is a feasible method to detect link flooding attacks. Liaskos *et al.* [117] formulated a detection method of link flooding attacks via relational algebra. It represents the association of bots to potential targets. They continuously rerouted traffic in a manner so that bots are forced to persistent participation in link flooding events. Thus, the bots exhibit suspicious behaviors and reveal their presence. Lee *et al.* [118] proposed a collaborative re-routing system that reroutes legitimate traffic to uncongested links while limits bandwidth available to attack traffic at congested links.

Actually, it is difficult to detect link flooding attacks by using current detection methods of DDoS flooding attacks. There are three reasons behind this [114], [115]: (i) attack traffic does not reach the target region directly, (ii) bots disguise as legitimate users with valid addresses to communicate with decoy servers, (iii) botnets can flood target links without sending unwanted traffic, e.g., they can send useful packets to each other.

V. Security Data Analytics for Worm Attack Detection

Worms are a kind of self-duplicating and self-propagating malicious codes that spread themselves across networks without any human interaction. They compromise systems, steal sensitive information, congest network and launch many kinds of attacks. A worm's life consists of following phases [8], [12]: (i) target finding, worm uses scanning technologies to search vulnerable hosts that can be compromised easily, (ii) transferring, worm sends its duplication to victims after the victims are discovered, (iii) activation, malicious activities of worms can be triggered based on a specific data or under a certain condition, (iv) infection, once worm has successfully infected the host, the infected host will exhibit malicious behaviors. During the first two phases, the worm is active over the Internet and causes some network anomalies, making it possible to be detected by exploiting its special behaviors.

The categories of worms are classified by the different characteristics in target finding and transferring phases [8], as shown in Fig. 8. According to the manners in the target finding phase, we can classify worms into scan-based worms, topology-based worms and passive worms. The scan-based worms search vulnerable hosts by probing IP addresses, e.g., hit-list scanning worms, routable scanning worms, and blind scanning worms. The topology-based worms use the information contained in a victim machine to search new targets, e.g., email worms and social network worms. The passive worms wait for a host request and reply with worm duplication, e.g., CRClean worm. According to the ways in propagation, worms can be divided into self-carried worms, second channel worms and embedded worms. The self-carried worms straightforwardly transfer malicious codes embedded in payload by itself. The second channel worms obtain malicious codes by visiting specific public servers or infected machines through a backdoor. The embedded worms mix its malicious codes in legitimate traffic to hide themselves. Based on the ways in transmission, there are TCP-based worms that transmit malicious codes with TCP protocol and UDP-based worms that transmit malicious codes with UDP protocol. The difference between these two transmission schemes is that the TCP-based worms are latency-limited and the UDP-based worms

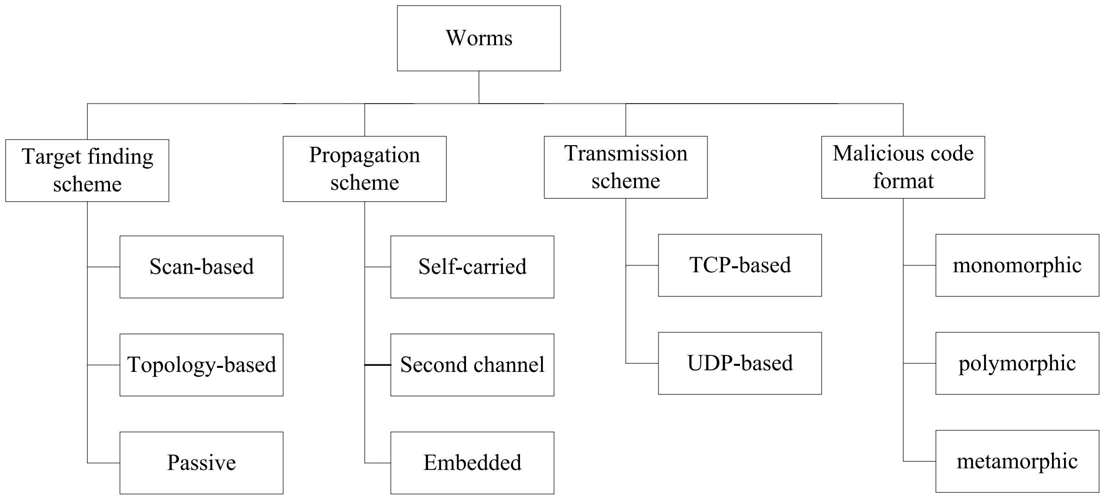


Fig. 8. The classification of worm characteristics.

are bandwidth-limited. According to the format of malicious codes that worms carry, worms can be classified into three types. The first one is monomorphic worms that do not change the sequence of malicious codes and always send the same content to targets. The second one is polymorphic worms that mutate malicious codes by using some encryption or semantics-preserving code manipulation techniques [119]. The polymorphic worms are the mostly prevalent over the Internet. The last one is metamorphic worms that perform different behaviors under different environments.

Comparing with the DDoS flooding attacks, worms are more difficult to detect. Herein, we discuss some typical detection methods for worm attacks based on the methods of data analytics and the data categories they used. Most methods detect worms by exploiting its special behaviors in target finding and transferring phases. Therefore, packet-level data and flow-level data are mainly used, while connection-level data (such as connection count) is analyzed by knowledge-based methods [8] and the host-level data is rarely analyzed alone to detect such kind of attacks. Moreover, we give a summary and comparison of detection methods of worm attacks in the last subsection of Section V.

A. Statistical Methods Against Worm Attacks

In statistical methods, behavior-based techniques are widely employed to capture essential characteristics of worms that reflect the unique behaviors in target finding and transferring phases. They inspect the headers of packets instead of checking the payload information. In the following, we discuss statistical analytic methods for detecting worms using packet-level data and flow-level data.

1) Use Packet-Level Data: Yang *et al.* [120] proposed a method for detecting local worms by analyzing the characteristics of the traffic generated by TCP-based worms. The TCP-based worms often send SYN packets to scan destination addresses and ports that are randomly selected. So there will be many SYN packets with no corresponding received SYN-ACK packets. The authors respectively counted the number of SYN packets sent out and the number of SYN-ACK packets received in the adjacent time interval. CUSUM algorithm is used to monitor the abrupt changes of the balanced relationship between SYN and SYN-ACK packets. By adjusting CUSUM algorithm with different collection methods, the proposed method can be applied to detect both high-speed and low-speed scanning. But their method cannot support all additional requirements.

Guo *et al.* [121] proposed a behavior-based detection method for Instant Messaging worms. They first defined three characteristic functions of Instant Messaging worms: the number of different users (distinct IP addresses) that one user communicates with using the same content, the number of users (distinct IP addresses) that one user communicates with in a certain period, and the number of packets with same size that one user sends out. Mahalanobis distance was employed to calculate the distance between the characteristic distributions of the normal behavior profile established in training phrase and newly observed traffic. Instead of setting a static threshold to measure the distance, the authors used CUSUM algorithm to monitor the distance changes to detect Instant Messaging worms during detection stage. The method cannot satisfy all additional requirements.

Zou *et al.* [122] presented a worm detection method by using the notion of "detecting monitored traffic trend, not burst" [123]. The "trend detection" method based on the basic characteristic of worms that a worm propagates exponentially with a constant and positive exponential rate in its early stage. They designed an Internet worm monitoring system. In the system, monitors record the number of scanning packets (such as TCP SYN) and the amount of different source addresses to calculate the average scan rate and scan distribution in a unit time. These two features were analyzed by Kalman filter to estimate whether there exist some illegitimate scan activities caused by a worm in the monitored traffic. In addition, the authors presented a formula to predict a worm's population size to show how many computers over the Internet are really infected based on monitored data. The method can support PI, but cannot achieve the goals of SD, DT and DFC.

2) Use Flow-Level Data: Yu *et al.* [124] introduced a new worm class, called Camouflaging Worm (C-Worm, in short). The C-Worm intelligently manipulates the volume of its scanned traffic in order to avoid generating any noticeable trends that are tracked by existing worm detection systems. The intelligent manipulation of C-Worm means that there is no significant difference between its traffic and non-worm traffic. However, the recurring manipulative nature of the C-Worm can be used as a basic characteristic to discriminate C-Worm since it is found that its distinction is clear in frequency domain. In the frequency domain, the Power Spectral Density (PSD) function shows a comparatively even distribution across a wide spectrum range for non-worm scan traffic, while the PSD of C-Worm scan traffic show spikes or noticeably high concentrations at a certain range of spectrum. The function of Spectral Flatness Measure (SFM) is applied to measure the degree of flatness of PSD, a bigger SFM value implies flatter PSD distribution and vice versa. Based on the above knowledge, they aggregated packets into flows with the same port number. For each flow, the distribution of PSD of the number of unique destination IP addresses and its corresponding SFM are calculated. If the SFM value is smaller than a predefined threshold, a C-Worm propagation alert is generated. This method satisfies the requirements of SD and PI, but cannot achieve the goals of DT and DFC.

Comparing with normal TCP-based traffic, the traffic generated by TCP-based scanning worms has a determinate rate prescribed by worm's self-propagation codes. This leads to different characteristics in frequency domain that can be used to distinguish worm traffic from normal traffic. Kim *et al.* [125] employed autocorrelation and Power Spectral Density (PSD) estimations to extract the frequency characteristic of SYN packet arrival time from a flow defined as SYN arrivals with the same source and destination address pair. The estimation methods showed that the frequency characteristic of arrival time of SYN packets from legitimate hosts spreads out all over the frequency band, whereas the packets from a worm infected host does not. Their method has low implementation complexity and the parameters of the method are independent from the network size and time-of-day. Thus, the method can support SD, but cannot achieve the goals of DT, PI and DFC.

Because the hit-list makes worm scan more targeted, the infection speed is much higher than the initial spreading phase. Based on this intrinsic property of hit-list scanning worms, Collins and Reiter [126] used Netflow traffic to detect them. They built protocol graphs that each of them is a representation of a traffic log for a single protocol (e.g., HTTP, FTP, SMTP and Oracle). In the graph, the vertices represent the IP addresses used by clients or servers for a particular protocol, and the edges represent communications between those addresses. Normally, the number of vertices in the graph is stable over time and also the pattern of communications has the same property. Under the attack of hit-list scanning worms, these regularities will be disturbed with a large number of vertices in the graph (the scanned hosts) and enlarged communications. Their method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

Wagner and Plattner [127] developed a Netflow-based entropy analysis that highlights outbreaks of fast worms. During worm scanning, the flow count with the same source IP address (the scanning host) will drastically increase. This changing characteristic leads to the decrease of the entropy in the distribution of the source IP addresses. At the same time, the flow count with different destination IP addresses will also increase since the scanning host attempts to contact others. This changing characteristic leads to the increase of the entropy in the distribution of destination IP addresses. By observing variations of the entropy values, the source of worms (the scanning host) will be detected efficiently. But their method cannot satisfy all additional requirements. Based on the same principle, Stoecklin *et al.* [128] also measured the variation of these two types of flow counts to detect attacks. Unlike Wagner's work, Gates *et al.* [129] used histogram-based analysis instead of entropy-based analysis.

Bin *et al.* [130] introduced a protocol independent flow analysis method based on NetFlow. They used the number of transmitting bytes (sent/received), outgoing flow records and bidirectional flow records to build a feature vector for each flow. Variance similarity or Euclidean distance is calculated between monitored flows and normal profile that is trained as the average of normal traffic records. If the value of similarity or distance is greater than a predefined threshold, the flow is considered as an attack flow. This method is capable of detecting worms, Trojan horses, malicious network attacks, and unexpected network applications. Their method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

Muraleedharan and Parmar [131] presented a behavior-based detection for TCP-based fast and slow scanning. They generated two profiles namely short-term profile and long-term profile of TCP scanning. The short-term profile is a time-based profile for fast scanning, which works with flow duration, flow size, average packet size in each flow, etc. Differently, the long-term profile is independent of time and is used to detect slow scanning. It takes source address and port, destination address and port, and packet size as parameters and applies entropy to build the profile. Both fast and slow scanning can be detected by measuring the distance between currently collected traffic and the two profiles. However, their method cannot support all additional requirements.

B. Machine Learning Against Worm Attacks

The main role of machine learning in worm detection is to discriminate worm traffic from legitimate traffic. It learns the distinct characteristics between worm traffic and legitimate traffic and then builds a judge criterion. According to the criterion, the current collected network traffic is analyzed in order to determine whether the traffic is generated by a worm. In the following, we discuss machine learning for detecting worms using packet-level data, flow-level data and host-level data.

1) Use Packet-Level Data: Farag *et al.* [132] designed a model that consists of four modules to identify worm traffic from normal traffic and predict infection percentage in a network. The first module is Traffic Statistical Analyzer

Module that collects traffic and then calculates traffic features in terms of packet rate, the number of packets generated by each source/destination port in a unit time period, and the number of packets per protocol in a unit time period. The second module is Port Matching Module that records the number of packets per port that matches worm infection behaviors. The third module is ANN Module that takes the above traffic features as input and identifies the worm traffic from normal traffic. The last module is Classification/Prediction Separated Model that employs two ANNs, one is used to generate worm behavior class and normal behavior class, the other is used to obtain the percentage of infection in the network. But the method cannot satisfy all additional requirements.

Sun and Chen [133] proposed a clustering and rough set based worm detection method to distill and block worm traffic in an early stage. They extracted many header features to construct feature vector for each packet, such as packet size, source and destination addresses, TTL value, header length, etc. An improved clustering algorithm was used to cluster packets. After that, similar clusters, measured by a matching value that represents the number of equivalent characteristic values between two clusters, were merged to generate a super cluster. This super cluster is a suspicious cluster due to the similarity among the worm traffic. The authors also measured the growth rate of the super cluster to confirm its suspiciousness. If it exceeds a pre-defined threshold, the super cluster is highly possible a result of worm attacks. This step eliminates the false alarm caused by flash crowds. Blocking rules were established by employing a rough set theory to calculate boundary approximation and lower approximation of super cluster. Thus, their method can satisfy the requirements of SD, PI and DFC, but cannot support DT.

2) Use Flow-Level Data: Comar *et al.* [134] designed a novel integrated detection method that leverages the accuracy of supervised classification in detecting known attacks and holds the adaptability of unsupervised learning for detecting new attacks. First, they extracted 108 flow-level and packet-level features in each flow, such as flow duration, flow direction, flow size, statistical values of packet size, etc. Then, they used an effective tree-based feature transformation approach to mitigate data imperfection issues and construct informative, non-linear features for accurate detection. Meanwhile, an intrusion detection system module performs Deep Packet Inspection (DPI) and tags each flow whether it belongs to some threat. Otherwise, the flow is labeled as "good/unknown". By combining the results of these two processes, a macro-level binary classifier and many micro-level classifiers are built to detect malicious flows. Their method can not only detect worms but also some malwares. Thus, their method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

Email worms remain a serious security threat over the Internet. It causes network congestion and delivers many kinds of viruses. The propagation strategy of email worms relies on social engineering, spreading via email among social contacts. Due to its specific propagation strategy, traditional worm detection methods are incapable of detecting this class of Internet worms. But by analyzing the traffic that email worms generate, researchers found that it can lead some anomalies in DNS traffic [135], [136]. Based on this knowledge, Chatzis and Popescu-Zeletin [135] proposed a method that can detect email worms in a local name server at early stage. They aggregated DNS query packets with source IP addresses. Normal DNS query flows share many same canonical behaviors, while abnormal DNS query flows that email worms generate are also similar to each other. Thus, Hierarchical clustering and similarity search over time series based on flow sizes were used to find the classes of normal and abnormal traffic respectively. The method can only support SD.

Email worm generates a large number of traffic that does not rely on DNS to translate names into numeric IP addresses. Based on the similar principle, Abdulla *et al.* [137] captured flow-level data using Netflow standard within a certain period of time and classified the DNS flows into four types: DNS request data, DNS response data, DNS normal data that the DNS flows sent by using fully qualified domain names, and DNS anomalies that the DNS flows sent by using IP addresses rather than fully qualified domain names. The flow counts of the above types of flows were considered as inputs of K-Nearest Neighbors and Naive Bayes to judge whether the traffic in this period of time is abnormal. But their method cannot support all additional requirements.

3) Use Host-Level Data: Stopel *et al.* [138] presented a novel approach based on Artificial Neural Network (ANN) for analyzing computer behaviors to detect worms. They collected many features from equipment operation logs (e.g., about writable objects, cursor changes, windows, and keyboard, etc.), system operation logs (e.g., about memory, network interface, physical disk, processes, processor, etc.) and network behaviors (e.g., the statistic information of TCP, UDP and ICMP). These features are merged to generate a vector of 323 features. Fisher's score ranking is preferable applied to select important behavior features that can be used to detect the presence of worms. Although their method exhibits a sound performance in detecting new behaviors of known worms, the process of measuring features consumes significant amount of computing power. The method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

Corporations usually deployed many defensive mechanisms to protect themselves from worm attacks while personal computer (PC) users have insufficient protection [139]. Major obstacles for detector deployment are the high false positive alarms and the operation overload. Early work such as BINDER [139], based on the intuition that network connections initiated by worms are rarely triggered by user inputs (e.g., keyboard and mouse click events), so it uses time difference between network connections and user triggered events as a detection feature. BINDER's detection method is incapable of detecting unknown worms and can be evaded by attackers with fake user events or infected normal programs. Seo *et al.* [140] designed an improved detection method called PC-WDS based on the extend ideas proposed in BINDER. It applies sophisticated features to improve detection accuracy while reducing false alarms. PC-WDS conducts the following analysis: (i) user interaction analysis: analyzing the time difference between equipment operation logs and network connections to estimate the likelihood that a user's interaction that really triggers network connections, (ii) destination address analysis: analyzing the change rate of destination addresses to predict whether the system is undertaking the worm scanning, (iii) network connection analysis: analyzing the number of failed network connections to calculate the likelihood of machine-triggered activity for every event of failed network connection, (iv) port creation analysis: analyzing system operation logs to calculate the likelihood that port creation has been triggered by user events. A normal behavior profile is built based on the likelihood of machine-triggered failed network connections, port creations and new destination addresses. Finally, a one-class SVM algorithm is used to detect outliers as the Internet worm behaviors by using the normal profile. Experimental results show that PC-WDS that utilizes the basic characteristic of worms has the ability of detecting unknown worms with high accuracy. Thus, PC-WDS can support PI, but cannot satisfy the requirements of SD, DT and DFC.

C. Knowledge-Based Methods Against Worm Attacks

In knowledge-based methods, some techniques are widely employed to detect worms. One is signature-based techniques. There are three types of signatures: content-based, semantic-based and vulnerability-based. Content-based signatures are established based on strings or substrings in byte sequences. Semantic-based signatures are generated by using the structure of the executable code or the analysis information of worms. Vulnerability-based signatures are generated by capturing the characteristics of the vulnerability the worm exploits. Another technique is expert system that matches network activities with prebuilt attack rules of worms. In the following, we discuss knowledge-based methods for detecting worms using packet-level data, flow-level data and connection-level data.

1) Use Packet-Level Data: Many worms use a pseudo random number generator to generate random addresses as infection targets. This behavior without undertaking a DNS query is different from legitimate publicly available services and then is considered as a sign of propagation of worms. Ahmad *el al.* [141] presented a detection system that keeps tracking outgoing SYN and UDP packets of monitored traffic. They correlated SYN and UDP packets with a DNS resolution cache to determine the absence of DNS lookup. If the number of this type of packets exceeds a threshold, the propagation of worms is detected. Then a containment solution starts to block traffic. Their method can only support SD. Similar approach is proposed in [142].

When spreading, worms always exploit the same set of vulnerabilities and try to infect others as soon as they have infected the current host. Chen *et al.* [143] proposed a real-time detection called WormTerminator. They set a virtual machine that clones the host and runs in parallel to the host. Any outgoing traffic from the host is firstly delivered to the virtual machine. Therefore, if a worm has successfully infected the current host, the virtual machine must be subsequently infected after receiving the traffic that the worm generated. Then the virtual machine will exhibit malicious behaviors and start to infect other hosts. Two timing correlation parameters are used: (i) T_{time}, the maximum time interval between the time when the virtual machine receives traffic and the time when the virtual machine starts to send out traffic, (ii) T_{size}, the time that is consumed to transfer all the traffic. Fast spreading worms strive to propagate to and infect as many other hosts as possible in the shortest possible time and the size of them is usually small. So, if the virtual machine receives some continuous traffic whose transmission time is less than T_{size}, and starts to send similar traffic to other hosts within time T_{time}, the delivered traffic is considered as worm traffic. However, their method cannot satisfy all additional requirements.

Xiao *et al.* [144] proposed a detection method based on the traffic information of each single process instead of all the processes. For one process whose traffic is TCP or UDP protocol based, they considered it as suspicious in case that the number and change rate of source ports exceed a threshold. For one process based on ICMP protocol, if it has sent out too many ICMP packets and these packets have different destination addresses, this process is considered as suspicious. Then they used the traffic similarity among suspicious processes to check worm traffic. But the method cannot satisfy all additional requirements.

Tang and Chen [145] analyzed the characteristics of polymorphic worms in detail and then introduced a new Position Aware Distribution Signature (PADS)-based detection method to resist them. They first partitioned network traffic into clusters using Normalized Cuts algorithm. After partitioning, Expectation-Maximization and Gibbs Sampling are used to compute PADSs. Instead of using single PADS, the multi-segment position-aware distribution signature that contains a set of PADSs is utilized to match incoming byte sequence to identify potential worms. However, their method cannot satisfy all additional requirements.

2) Use Flow-Level Data: The worms that utilize buffer overflow vulnerabilities often send longer length of certain protocol fields than those in normal packets to overflow the buffer. Wang *et al.* [146] proposed the first network-based vulnerability-signature method that generates length-based signatures for detecting buffer overflow worms. They first classified network traffic into different protocol flows based on port numbers or other protocol identifiers. Then, for each protocol, they filtered out known worms and separated the traffic into a suspicious traffic pool and a normal traffic pool. Both the pools are analyzed by a protocol parser to obtain each protocol message of a set of packet header fields in flows. Each field is expressed with a type and a length. The field length information of both the pools is then considered as input to generate signatures. The length-based signatures are effectively used to detect buffer overflow worms, and are very hard for attackers to evade. Their method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

Wang *et al.* [147] designed a payload-based system for polymorphic worm detection and signature extraction. As a front-end processing stage, a multi-dimensional traffic clustering and classification scheme is used for flow classification and separating clusters into anomaly/attack. Then, a generalized suffix tree is built based on the payload of all packets in flows in suspicious clusters. After building the tree, all nodes are traversed and the prefix of each selected node is taken as a putative worm signature. Their method can support PI, but cannot satisfy the requirements of SD, DT and DFC.

3) Use Connection-Level Data: Worms send out large numbers of scanning packets to find victims. During blind scanning, the scan includes unused addresses and closed ports, causing the failure of connection attempts. Keeping track of the outbound connection attempts is a traditional way to detect worms in the scanning phase. Rasheed *et al.* [148] recorded the number of first failed connection by counting ICMP error packets or TCP RST packets returned from external destination addresses in each connection. An alarm will be raised when the number of first failed connections is higher than an estimated threshold. The detection method is flexible because it calculates the alarm threshold at different time. The method can achieve the goals of SD and DT, but cannot support PI and DFC. Moreover, based on the high failure probability of First Contact Connections (FCCs) of worms, some detection methods were proposed [149]–[152].

After analyzing real worms and botnet traffic with Snort, Braun *et al.* [153] concluded that the majority of the signatures of such attacks can be found by checking the first few kilobytes of payload during TCP connections. They employed two time-out Bloom filters that are composed of timestamps instead of bits to respectively store the start and end time of a connection. Meanwhile, one sketch was employed to store the number of payload bytes in a connection. Then they used a packet sampling algorithm to collect the first N bytes of payload of a connection to find whether there are existing attack signatures. However, their method cannot satisfy all additional requirements.

D. Discussion

With the same format as in Table V, we summarize and compare detection methods of worm attacks in Table IX. From the table, we find that IP address, port and the volume of traffic are widely used to detect worms due to the basic characteristics of worm scanning [120]–[122], [124]–[127], [130], [131], [141], [143], [144], [148]. Some works used the features such as packet rate, packet arrival time, packet size, payload, etc., to detect worms in their transferring phase [132]–[135], [137], [138], [140], [145]–[147], [153]. Host-level data are used in conjunction with packet-level and flow-level data because worms are malicious programs that directly execute at host-side [138], [140]. Each kind of worms has different reflecting characteristics. Thus, detection methods for worms are also quite different. Normally, we should make full use of the specific behaviors of worms in both scanning and transferring phases to detect them.

Besides, most of the worm detection methods do not consider the problem of flash crowds. A worm can easily hide its malicious activities under the flash crowds. Also, dynamic thresholds are not widely employed to measure the changes of network traffic, which may lead to low detection performance due to the variability of worms.

VI. Open Issues and Future Research Directions

A. Open Issues

According to the above literature review and comparisons, we figure out a number of open issues with regard to the detection of DDoS flooding and worm attacks, security data collection and data analytics.

First, at present, the configuration of reasonable relationships among data types for generating a normal profile is not well solved. Entropy and probability distributions are widely used in the calculation of deviation/information distance between the normal profile and a real-time traffic profile. But during daily network traffic transmission, inevitable network congestion or other abrupt events may change the relationships among data types that were set up in advance and saved in the normal profile. The normal profile that represents fragile relationships among data types could cause a high false alarm rate. For example, some methods detect SYN flooding attacks by monitoring the changes in the balanced relationship between the numbers of SYN packets and SYN-ACK packets. However, in actual network transmission, the number of SYN packets is much higher than that of SYN-ACK packets due to network congestion. It is hard to accurately decide quantitative deviation as an alert threshold for SYN flooding attacks. In another instance, some detection methods for SIP flooding attacks build a normal profile based on the probability distributions of the number of SIP INVITE, SIP 200 OK, SIP ACK and SIP BYE packets. But the detection methods will become ineffective if the four packet counts are simultaneously and proportionally increased. Thus, setting up a high-quality normal profile that can effectively detect the abnormal still need further and deep investigation.

Second, energy and cost efficiency are not considered in the most of current detection methods, which impacts practical deployment of the proposed methods. Especially when using machine learning technologies to detect attacks. Massive features of network traffic are needed in order to achieve high detection accuracy, as shown in Tables V–IX. But extracting massive features for analysis could consume time and many resources. Energy efficient solutions are highly expected in practice. But only a few existing studies take this factor into consideration.

Third, context-aware and flexible detection methods are seldom studied at present. There are two main issues. First, most threshold-based detection methods set a static threshold. Since network traffic fluctuates over time, a static threshold lacks flexibility and is unable to be applied into different networking scenarios. How to dynamically set a threshold that is adaptive to networking contexts at real time should be urgently studied. Second, how to effectively discriminate attack traffic from flash crowds is still an open issue. Most of the detection methods do not take the problem of flash crowds into account. Flash crowds are unexpected, but inevitable.

Forth, the literature still lacks a comprehensive and holistic detection method that can fully make use of all four categories of security data to measure the Internet security. There are few studies about combining host-level data, connection-level data with packet-level data and flow-level data to detect

TABLE IX
SUMMARY AND COMPARISON OF DETECTION METHODS OF WORM ATTACKS

References	Collected Security Data	Analysis Method	DA	FA	SD	DT	PI	DFC	Remarks
[120]	The number of packets	CUSUM algorithm	N.A.	N.A.	N	N	N	N	Detect both high-speed and low-speed scanning
[121]	The number of packets, packet size, packet payload, destination address	Distance variation and CUSUM algorithm	Around 80%	N.A.	N	N	N	N	Detection method is flexible and short latency
[122]	The number of packets, source address	Kalman filter	Around 95%	N.A.	N	N	Y	N	The trend detection method poses many interesting research issues
[124]	Destination address	Frequency domain analysis	99.3%	1%	Y	N	Y	N	Superior detection performance against Camouflaging worm
[125]	Packet arrival time	Frequency analysis	N.A.	0%	Y	N	N	N	Detect both high-speed and low-speed scanning
[126]	Source and destination address	Graph theory analysis	Around 95%	Around 5%	N	N	Y	N	This detection method can be applied into a large-scale network
[127]	Flow count, source and destination address	Entropy variation	N.A.	N.A.	N	N	N	N	This detection method does not need parametrization
[130]	Netflow records, the number of transmitting bytes	Similarity/distance variation	99%	1%	N	N	Y	N	This detection method is easily expanded
[131]	Flow duration, flow count, flow size, source address etc.	Distance variation	N.A.	N.A.	N	N	N	N	Detect both high-speed and low-speed scanning
[132]	Packet rate, the number of packets	ANN	99.71%	0.07%	N	N	N	N	Have the ability to predict the infection percentage of worms in a network
[133]	Packet size, source and destination address, TTL value, etc.	Clustering algorithm	99.82%/ 99.93%	1.67%/ 0.48%	Y	N	Y	N	The detection and blocking scheme is very effective and accurate
[134]	Flow duration, flow size, flow direction, packet size, etc.	SVM with a weighted linear kernel	Around 95%	10%	N	N	Y	N	Use of two-level classifier to address the problem of data quality
[135]	Flow size, source and destination port	Hierarchical clustering	99.5%	1%	Y	N	N	N	Utilize common spreading behavior to detect worms
[137]	Flow count, source and destination port	KNN and Naive Bayes	85.64%-98.07%	1%-16%	N	N	N	N	Tested the method with real-life worm variants
[138]	The number of packets, operation logs	ANN	Around 90%	0.04%	N	N	Y	N	Select important features by applying feature selection techniques
[140]	Source and destination address, source port, operation logs, etc.	SVM algorithm	100%	0.08%	N	N	Y	N	Detect worms at personal computers
[141]	Destination address, destination port	Match attack rule	100%	0%	Y	N	N	N	Utilize specific transferring behaviors to detect worms
[143]	Packet rate, inter-arrival time of packets	Match attack rule	N.A.	0%	N	N	N	N	Have the ability to detect zero-day and polymorphic worms
[144]	Destination address, source and destination port , the number of packets	Match attack rule	N.A.	Around 1.5%	N	N	N	N	Detect worms at end hosts
[145]	Packet payload	Signature-based detection	N.A.	0%	N	N	N	N	Have the ability of separating new worm variants
[146]	Packet size in each flow	Length-based signature detection	100%	0%	N	N	Y	N	The first network-based length-based signature generator
[147]	Packet payload in each flow	Signature-based detection	100%	0%	N	N	Y	N	Use different granularity of network traffic
[148]	Connection count	Match attack rule	N.A.	N.A.	Y	Y	N	N	Have the ability of detecting stealthy and unknown worms
[153]	Packet payload in each connection	Signature-based detection	N.A.	Around 7%	N	N	N	N	Only check the first N bytes in each connection

DA: Detection Accuracy; FA: False Alarm; N.A.: Not Available;
SD: Self-adaptive Detection; DT: Dynamic Threshold; PI: Protocol Independence; DFC: Deal with Flash Crowds; Y: Yes; N: No

DDoS flooding and worm attacks. For example, an attacker can launch application layer DDoS flooding attacks by using legitimate connection. In this condition, tracking connection status and drilling down packet-level data or flow-level data are necessary. It is also possible to detect worms at the propagation stage. Moreover, quite a number of detection methods do not consider host-level data that provides comprehensive information of system events. Some work applies host-level data (such as application operation logs, equipment operation logs, host behaviors, etc.) to detect network intrusions on hosts [154]–[159] and concerns host data privacy protection during data collection and processing [160]. A comprehensive detection method by applying all four categories of security data still lacks, but highly expected in the literature for detecting all or most security threats for the purpose of network security measurement.

Last but not the least, a generic and pervasive solution for detecting various attacks and providing a thorough view on the Internet security at real time is still missing but highly expected in practice. Such a solution is of significantly important to network security.

B. Future Research Directions

All above open issues motivate future research. We suggest a number of interesting future research directions as outlined below.

First, adaptive security data collection methods should be researched for achieving efficient and comprehensive data collection. Current detection methods choose data category in advance and then collect data constantly in any cases. As we have discussed in Section II, collection methods of packet-level data are not suitable in high-speed network. Likewise, collection methods of flow-level data convey less information about network traffic than packet-level data. Different network contexts request different traffic collection methods. Moreover, as we have presented in Sections IV and V, each kind of attacks exhibits its own characteristics and can be detected in different circumstances by selecting appropriate data categories and data items. In order to counter any abrupt changes in network traffic, data collection methods that can adaptively collect appropriate data are urgently needed.

Second, a generic and pervasive Internet security measurement solution should be studied by applying all four typical categories of security data. Each security data category reveals different information about network traffic. How to efficiently and effectively make use of these data categories to detect all potential attacks in different circumstances is worth exploring. We should make use of each data category in an integrated way to figure out the Internet security in general. For example, it has potential to work out advanced detection methods for detecting AL-DDoS flooding attacks by using connection-level data, packet-level and flow-level data, for detecting DRDoS flooding attacks by combining flow-level data with packet-level data, and for designing worm detection methods by using all four data categories. In addition, how to economically collect sufficient data to detect synthetic attacks (different types of attacks happening at the same time) is an interesting and significant research topic.

Third, the Internet security measurement with efficiency and traceability is a very interesting research topic. It is significant to explore new theories and methods or apply some existing theories for addressing this issue. For example, Granular Computing as a growing and powerful theory for complex problem solving, large-scale data mining and fuzzy information processing can be applied to detect and trace attacks. Yao *et al.* [161] classified granular computing research into three groups: philosophical and fundamental views of Granular Computing, individual Granular Computing techniques, and Granular Computing applications. It is anticipated that Granular Computing can lead to new computational paradigms. Throughout the developments in these years, Granular Computing has shown many advantages when dealing with big data, such as attribute reduction [162], multi-source data aggregation [163], and feature selection [164]. We can apply the same theory into network security data collection and analytics. In terms of network security data, packets as the basic granules can be constructed into flows with flow keys and flows can be further constructed into connection data with addresses and ports. Each connection data drills down into flows or packets. Based on these relationships, selecting appropriate original information granularity and optimal granular description of network security data in different network scenarios can not only reduce the size of data used for analytics, but also has the ability of dealing with elaborate attacks.

Forth, trustworthy security data fusion is an essential and significant research topic for the purpose of measuring the Internet security as a whole. Efficient and effective preprocessing of network security data is the precondition for detecting attacks. Incomplete, uncertain, imprecise or vague information is the main reason of wrong or poor-quality detection. How to efficiently process these data in a trustworthy (at least dependable and reliable) way is still an open research issue. In the future, it is beneficial to pay more attention to data fusion and composition methods to deal with network security data, such as the sketch technique applied in [73] and [74]. Data composition represents a process of data preprocessing and aims to extract most valuable and useful data with high trustworthiness. It is an essential step for intrusion detection and security level calculation, considering the big data feature of network security data over the Internet.

VII. Conclusion

Although there are numerous surveys about detection methods for network attacks, there is still an acute lack of a perspective from the viewpoint of the categories of security data. In this paper, we first classified security data into four categories and presented a detailed description of each of them. For each category, we discussed its types that are commonly used to detect network attacks. We also discussed analytic methods of security data and proposed four additional requirements for evaluating their performance in order to support detection scalability and flexibility. Then, we thoroughly surveyed current detection methods for DDoS flooding and worm attacks from the perspective of security data and data analytic methods. We elaborated in detail what categories of data are used and which analytic method is applied. Each attack has its own specific characteristics. When designing a detection method, we should thoroughly understand attack characteristics and select appropriate data categories and data analytic methods to meet the needs. Some open issues and future research directions show that there is still a long way to go, especially from the view of the Internet security measurement. For achieving this goal, we should figure out all potential threats by processing all categories of security data.

Working out effective data composition or fusion methods is essential for realizing efficient and economic security data analytics. Forming more effective analytic methods can emerge as a promising direction worth following.

REFERENCES

[1] Y. L. Zou, J. Zhu, X. B. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proc. IEEE*, vol. 104, no. 9, pp. 1727–1765, Sep. 2016.

[2] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.

[3] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci.*, vol. 305, pp. 357–383, Jun. 2015.

[4] E. G. AbdAllah, H. S. Hassanein, and M. Zulkernine, "A survey of security attacks in information-centric networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1441–1454, 3rd Quart., 2015.

[5] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Comput. Surveys*, vol. 39, no. 1, p. 3, 2007.

[6] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046–2069, 4th Quart., 2013.

[7] Q. Yan, F. R. Yu, Q. X. Gong, and J. Q. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.

[8] P. L. Li, M. Salour, and X. Su, "A survey of Internet worm detection and containment," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 20–35, 1st Quart., 2008.

[9] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1520–1549, 3rd Quart., 2014.

[10] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Botnet in DDoS attacks: Trends and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2242–2270, 4th Quart., 2015.

[11] M. F. Umer, M. Sher, and Y. X. Bi, "Flow-based intrusion detection: Techniques and challenges," *Comput. Security*, vol. 70, pp. 238–254, Sep. 2017.

[12] Y. Tang, J. Q. Luo, B. Xiao, and G. Y. Wei, "Concept, characteristics and defending mechanism of worms," *IEICE Trans. Inf. Syst.*, vol. 92, no. 5, pp. 799–809, 2009.

[13] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014.

[14] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "DDoS attacks in cloud computing: Issues, taxonomy, and future directions," *Comput. Commun.*, vol. 107, no. 10, pp. 30–48, 2017.

[15] V. Jacobson, C. Leres, and S. McCanne, *The TCPDUMP Manual Page*, Lawrence Berkeley Lab., Berkeley, CA, USA, 1989.

[16] G. Combs. *Wireshark*. Accessed: Aug. 14, 2018. [Online]. Available: http://www.wireshark.org

[17] *Snort*. Accessed: Aug. 14, 2018. [Online]. Available: https://www.snort.org

[18] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Sunnyvale, CA, USA: Insecure, 2009.

[19] S. Alcock, P. Lorier, and R. Nelson, "Libtrace: A packet capture and analysis library," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2012, pp. 42–48.

[20] J. Weber, *The Fundamentals of Passive Monitoring Access*, Net Opt. Inc., Santa Clara, CA, USA, 2006.

[21] M. V. Mahoney and P. K. Chan, "PHAD: Packet header anomaly detection for identifying hostile network traffic," Dept. Comput. Sci., Florida Inst. Technol., Melbourne, FL, USA, Rep. CS-2001-04, 2001.

[22] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: An effective defense against spoofed DDoS traffic," in *Proc. 10th ACM Conf. Comput. Commun. Security*, 2003, pp. 30–41.

[23] L. Rudman and B. Irwin, "Characterization and analysis of NTP amplification based DDoS attacks," in *Proc. Inf. Security South Africa*, 2015, pp. 1–5.

[24] L. Kavisankar, C. Chellappan, P. Sivasankar, A. Karthi, and A. Srinivas, "A pioneer scheme in the detection and defense of DRDoS attack involving spoofed flooding packets," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 5, pp. 1726–1743, 2014.

[25] T. AbuHmed, A. Mohaisen, and D. Nyang, "A survey on deep packet inspection for intrusion detection systems," *Mag. Korea Telecommun. Soc.*, vol. 24, no. 2, pp. 25–36, 2008.

[26] R. Hofstede *et al.*, "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2037–2064, 4th Quart., 2014.

[27] *Netflow*. Accessed: Aug. 14, 2018. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

[28] T. Fioreze, M. O. Wolbers, R. van de Meent, and A. Pras, "Finding elephant flows for optical networks," in *Proc. 10th IFIP/IEEE Int. Symp. Integr. Netw. Manag.*, 2007, pp. 627–640.

[29] B. D. Li, J. Springer, G. Bebis, and M. H. Gunes, "A survey of network flow applications," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 567–581, 2013.

[30] A. Sperotto *et al.*, "An overview of IP flow-based intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 343–356, 3rd Quart., 2010.

[31] *Collectl*. Accessed: Aug. 14, 2018. [Online]. Available: http://collectl.sourceforge.net/index.html

[32] *Loadrunner*. Accessed: Aug. 14, 2018. [Online]. Available: https://saas.hpe.com/zh-cn/software/loadrunner

[33] P. Berezinski, B. Jasiul, and M. Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, no. 4, pp. 2367–2408, 2015.

[34] W. Lee and D. Xiang, "Information-theoretic measures for anomaly detection," in *Proc. IEEE Symp. Security Privacy*, 2001, pp. 130–143.

[35] M. Yu, "A nonparametric adaptive CUSUM method and its application in network anomaly detection," *Int. J. Adv. Comput. Technol.*, vol. 4, no. 1, pp. 280–288, 2012.

[36] N. Ye, S. Vilbert, and Q. Chen, "Computer intrusion detection through EWMA for autocorrelated and uncorrelated data," *IEEE Trans. Rel.*, vol. 51, no. 1, pp. 75–82, Mar. 2003.

[37] J. D. Brutlag, "Aberrant behavior detection in time series for network service monitoring," in *Proc. Usenix Conf. Syst. Admin.*, 2000, pp. 139–146.

[38] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Netw.*, vol. 51, no. 12, pp. 3448–3470, 2007.

[39] T. T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 4th Quart., 2008.

[40] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[41] M. S. Mahdavinejad *et al.*, "Machine learning for Internet of Things data analysis: A survey," *arXiv:1802.06305*, 2018. Accessed: Aug. 14, 2018. [Online]. Available: https://arxiv.org/abs/1802.06305

[42] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Softw. Qual. J.*, vol. 26, no. 3, pp. 891–919, 2018.

[43] J. Peng, K.-K. R. Choo, and H. Ashman, "User profiling in intrusion detection: A review," *J. Netw. Comput. Appl.*, vol. 72, pp. 14–27, Sep. 2016.

[44] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, "A taxonomy of botnet behavior, detection, and defense," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 898–924, 2nd Quart., 2014.

[45] M. Bellaiche and J.-C. Gregoire, "SYN flooding attack detection based on entropy computing," in *Proc. Glob. Telecommun.*, 2009, pp. 1–6.

[46] H. Sengar, W. Haining, D. Wijesekera, and S. Jajodia, "Detecting VoIP floods using the Hellinger distance," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 6, pp. 794–805, Jun. 2008.

[47] D. Boro, H. Basumatary, T. Goswami, and D. K. Bhattacharyya, "UDP flooding attack detection using information metric measure," in *Proc. Int. Conf. ICT Sustain. Develop.*, 2016, pp. 143–153.

[48] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: A statistics-based packet filtering scheme against distributed denial-of-service attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 3, no. 2, pp. 141–155, Apr./Jun. 2006.

[49] P. E. Ayres, H. Z. Sun, H. J. Chao, and W. C. Lau, "ALPi: A DDoS defense system for high-speed networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 10, pp. 1864–1876, Oct. 2006.

[50] Q. Chen, W. M. Lin, W. C. Dou, and S. Yu, "CBF: A packet filtering method for DDoS attack defense in cloud environment," in *Proc. IEEE 9th Int. Conf. Depend. Auton. Secure Comput.*, 2012, pp. 427–434.

[51] L. Zhou, M. C. Liao, C. Yuan, Z. Y. Sheng, and H. Y. Zhang, "DDoS attack detection using packet size interval," in *Proc. Int. Conf. Wireless Commun. Netw. Mobile Comput.*, 2015, pp. 1–7.

[52] J. David and C. Thomas, "DDoS attack detection using fast entropy approach on flow—Based network traffic," *Procedia Comput. Sci.*, vol. 50, no. 4, pp. 30–36, 2015.

[53] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using artificial neural networks," *Neurocomputing*, vol. 172, pp. 385–393, Jan. 2016.

[54] R. Vijayasarathy, B. Ravindran, and S. V. Raghavan, "A system approach to network modeling for DDoS detection using a naive Bayesian classifier," in *Proc. 3rd Int. Conf. Commun. Syst. Netw.*, 2011, pp. 1–10.

[55] M. Y. Su, "Real-time anomaly detection systems for denial-of-service attacks by weighted *k*-nearest-neighbor classifiers," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3492–3498, 2011.

[56] B. Kong, K. Yang, D. G. Sun, M. M. Li, and Z. X. Shi, "Distinguishing flooding distributed denial of service from flash crowds using four data mining approaches," *Comput. Sci. Inf. Syst.*, vol. 14, no. 3, pp. 839–859, 2017.

[57] K. Lee, J. Kim, K. H. Kwon, Y. Han, and S. Kim, "DDoS attack detection method using cluster analysis," *Expert Syst. Appl.*, vol. 34, no. 3, pp. 1659–1665, 2008.

[58] L. F. Zi, J. Yearwood, and X.-W. Wu, "Adaptive clustering with feature ranking for DDoS attacks detection," in *Proc. 4th Int. Conf. Netw. Syst. Security*, 2010, pp. 281–286.

[59] P. Xiao, W. Y. Qu, H. Qi, and Z. Y. Li, "Detecting DDoS attacks against data center with correlation analysis," *Comput. Commun.*, vol. 67, pp. 66–74, Aug. 2015.

[60] C. Wagner, J. Francois, R. State, and T. Engel, "Machine learning approach for IP-flow record anomaly detection," in *Proc. Int. IFIP TC 6 Conf. Netw.*, 2011, pp. 28–39.

[61] X. Qin, T. G. Xu, and C. Wang, "DDoS attack detection using flow entropy and clustering technique," in *Proc. Int. Conf. Comput. Intell. Security*, 2015, pp. 412–415.

[62] P. A. R. Kumar and S. Selvakumar, "Distributed denial of service attack detection using an ensemble of neural classifier," *Comput. Commun.*, vol. 34, no. 11, pp. 1328–1341, 2011.

[63] C. Sun, C. Hu, and B. Liu, "SACK^2: Effective SYN flood detection against skillful spoofs," *LET Inf. Security*, vol. 6, no. 3, pp. 149–156, Sep. 2012.

[64] V. K. Yadav, M. C. Trivedi, and B. M. Mehtre, "DDA: An approach to handle DDoS (ping flood) attack," in *Proc. Int. Conf. ICT Sustain. Develop.*, 2016, pp. 11–23.

[65] A. Sanmorino and S. Yazid, "DDoS attack detection method and mitigation using pattern of the flow," in *Proc. Int. Conf. Inf. Commun. Technol. (ICoICT)*, 2013, pp. 12–16.

[66] L. H. Miao, W. Ding, and J. Gong, "A real-time method for detecting Internet-wide SYN flooding attacks," in *Proc. Int. Workshop Local Metropolitan Area Netw.*, 2015, pp. 1–6.

[67] H. Rahmani, N. Sahli, and F. Kamoun, "DDoS flooding attack detection scheme based on F-divergence," *Comput. Commun.*, vol. 35, no. 11, pp. 1380–1391, 2012.

[68] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, "DDoS-shield: DDoS-resilient scheduling to counter application layer attacks under imperfect detection," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 26–39, Feb. 2009.

[69] M. H. Jiang, C. X. Wang, X. P. Luo, M. T. Miu, and T. Chen, "Characterizing the impacts of application layer DDoS attacks," in *Proc. IEEE Int. Conf. Web Services*, 2017, pp. 500–507.

[70] R. Alonso, R. Monroy, and L. A. Trejo, "Mining IP to domain name interactions to detect DNS flood attacks on recursive DNS servers," *Sensors*, vol. 16, no. 8, 2016, Art. no. E1311.

[71] M. Alenezi and M. J. Reed, "Denial of service detection through TCP congestion window analysis," in *Proc. World Congr. Internet Security*, 2013, pp. 145–150.

[72] W. Zhou, W. J. Jia, S. Wen, Y. Xiang, and W. L. Zhou, "Detection and defense of application-layer DDoS attacks in backbone Web traffic," *Future Gener. Comput. Syst.*, vol. 38, pp. 36–46, Sep. 2014.

[73] J. Tang, Y. Cheng, Y. Hao, and W. Song, "SIP flooding attack detection with a multi-dimensional sketch design," *IEEE Trans. Depend. Secure Comput.*, vol. 11, no. 6, pp. 582–595, Nov./Dec. 2014.

[74] C. X. Wang, T. T. N. Miu, X. P. Luo, and J. H. Wang, "SkyShield: A sketch-based defense system against application layer DDoS attacks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 559–573, Mar. 2018.

[75] T. Thapngam, S. Yu, W. L. Zhou, and G. Beliakov, "Discriminating DDoS attack traffic from flash crowd through packet arrival patterns," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2011, pp. 952–957.

[76] S. Yu, S. Guo, and I. Stojmenovic, "Fool me if you can: Mimicking attacks and anti-attacks in cyberspace," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 139–151, Jan. 2014.

[77] S. Bhatia, "Ensemble-based model for DDoS attack detection and flash event separation," in *Proc. Future Technol. Conf.*, 2017, pp. 958–967.

[78] S. Yu *et al.*, "Discriminating DDoS attacks from flash crowds using flow correlation coefficient," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 6, pp. 1073–1080, Jun. 2012.

[79] R. Saravanan, S. Shanmuganathan, and Y. Palanichamy, "Behavior-based detection of application layer distributed denial of service attacks during flash events," *Turkish J. Elect. Eng. Comput. Sci.*, vol. 24, no. 2, pp. 510–523, 2016.

[80] L. C. Giralte, C. Conde, I. M. de Diego, and E. Cabello, "Detecting denial of service by modelling Web-server behaviour," *Comput. Elect. Eng.*, vol. 39, no. 7, pp. 2252–2262, 2013.

[81] M. Zolotukhin, T. Kokkonen, T. Hamalainen, and J. Siltanen, "Weighted fuzzy clustering for online detection of application DDoS attacks in encrypted network traffic," in *Proc. Int. Conf. Next Gener. Wired/Wireless Netw. Conf. Internet Things Smart Spaces*, 2016, pp. 326–338.

[82] H. Beitollahi and G. Deconinck, "ConnectionScore: A statistical technique to resist application-layer DDoS attacks," *J. Ambient Intell. Humanized Comput.*, vol. 5, no. 3, pp. 425–442, 2014.

[83] E. Adi, Z. Baig, and P. Hingston, "Stealthy denial of service (DoS) attack modelling and detection for HTTP/2 services," *J. Netw. Comput. Appl.*, vol. 91, pp. 1–13, Aug. 2017.

[84] A. Ramamoorthi, T. Subbulakshmi, and S. M. Shalinie, "Real time detection and classification of DDoS attacks using enhanced SVM with string kernels," in *Proc. Int. Conf. Recent Trends Inf. Technol.*, 2011, pp. 91–96.

[85] C. Y. She, W. S. Wen, K. S. Zheng, and Y. Y. Lyu, "Application-layer DDoS detection by K-means algorithm," in *Proc. Int. Conf. Elect. Electron. Eng. Comput. Sci.*, 2016, pp. 75–78.

[86] K. J. Singh, K. Thongam, and T. De, "Entropy-based application layer DDoS attack detection using artificial neural networks," *Entropy*, vol. 18, no. 10, p. 350, 2016.

[87] D. Geneiatakis, N. Vrakas, and C. Lambrinoudakis, "Utilizing bloom filters for detecting flooding attacks against SIP based services," *Comput. Security*, vol. 29, no. 7, pp. 578–591, 2009.

[88] Q. Liao, H. Li, S. L. Kang, and C. C. Liu, "Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching," *Security Commun. Netw.*, vol. 8, no. 17, pp. 3111–3120, 2015.

[89] M. Zhang, W. Zhang, and K. Fan, "Application layer DDoS detection model based on data flow aggregation and evaluation," in *Communications and Information Processing*. Heidelberg, Germany: Springer, 2012, pp. 37–45.

[90] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks and counter strategies," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 683–696, Aug. 2006.

[91] J. T. Luo and X. L. Yang, "The NewShrew attack: A new type of low-rate TCP-targeted DoS attack," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 713–718.

[92] G. Maciá-Fernández, J. E. Díaz-Verdejo, P. García-Teodoro, and F. D. Toro-Negro, "LoRDAS: A low-rate DoS attack against application servers," in *Proc. Int. Workshop Crit. Inf. Infrastruct. Security*, 2007, pp. 197–209.

[93] M. Guirguis, A. Bestavros, and I. Matta, "Exploiting the transients of adaptation for RoQ attacks on Internet resources," in *Proc. 12th IEEE Int. Conf. Netw. Protocols*, 2004, pp. 184–195.

[94] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "FFSc: A novel measure for low-rate and high-rate DDoS attack detection using multivariate data analysis," *Security Commun. Netw.*, vol. 9, no. 13, pp. 2032–2041, 2016.

[95] Y. Xiang, K. Li, and W. L. Zhou, "Low-rate DDoS attacks detection and traceback by using new information metrics," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 2, pp. 426–437, Jun. 2011.

[96] Z. J. Wu, L. Y. Zhang, and M. Yue, "Low-rate DoS attacks detection based on network multifractal," *IEEE Trans. Depend. Secure Comput.*, vol. 13, no. 5, pp. 559–567, Sep./Oct. 2016.

[97] M. Yue, L. Liu, Z. J. Wu, and M. X. Wang, "Identifying LDoS attack traffic based on wavelet energy spectrum and combined neural network," *Int. J. Commun. Syst.*, vol. 31, no. 2, 2018, Art. no. e3449.

[98] Z. J. Wu, H. T. Zhang, M. H. Wang, and B. S. Pei, "MSABMS-based approach of detecting LDoS attack," *Comput. Security*, vol. 31, no. 4, pp. 402–417, 2012.

[99] L. Zhou, M. C. Liao, C. Yuan, and H. Y. Zhang, "Low-rate DDoS attack detection using expectation of packet size," *Security Commun. Netw.*, vol. 2017, no. 1, pp. 1–14, 2017.

[100] L. B. Wu, J. Cheng, Y. X. He, A. Xu, and P. Wen, "A low-rate DoS detection based on rate anomalies," in *Proc. Int. Conf. Appl. Informat. Commun.*, 2011, pp. 189–196.

[101] C. W. Zhang, Z. P. Cai, W. F. Chen, X. P. Luo, and J. P. Yin, "Flow level detection and filtering of low-rate DDoS," *Comput. Netw.*, vol. 56, no. 15, pp. 3417–3431, 2012.

[102] C. Rossow, "Amplification hell: Revisiting network protocols for DDoS abuse," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2014, pp. 1–15.

[103] M. Kuhrer, T. Hupperich, C. Rossow, and T. Holz, "Exit from hell? Reducing the impact of amplification DDoS attacks," in *Proc. 23rd USENIX Security Symp. (USENIX Security)*, 2014, pp. 111–125.

[104] F. J. Ryba, M. Orlinski, M. Wahlisch, C. Rossow, and T. C. Schmidt, "Amplification and DRDoS attack defense—A survey and new perspectives," *arXiv: 1505.07892*, 2015. Accessed: Aug. 14, 2018. [Online]. Available: https://arxiv.org/abs/1505.07892

[105] W. Wei, F. Chen, Y. J. Xia, and G. Jin, "A rank correlation based detection against distributed reflection DoS attacks," *IEEE Commun. Lett.*, vol. 17, no. 1, pp. 173–175, Jan. 2013.

[106] L. Xiao, W. Wei, W. D. Yang, Y. L. Shen, and X. L. Wu, "A protocol-free detection against cloud oriented reflection DoS attacks," *Soft Comput.*, vol. 21, no. 13, pp. 3713–3721, 2016.

[107] Y. X. Gao, Y. K. Feng, J. Kawamoto, and K. Sakurai, "A machine learning based approach for detecting DRDoS attacks and its performance evaluation," in *Proc. Asia Joint Conf. Inf. Security*, 2016, pp. 80–86.

[108] I. L. Meitei, K. J. Singh, and T. De, "Detection of DDoS DNS amplification attack using classification algorithm," in *Proc. Int. Conf. Informat. Anal.*, 2016, p. 81.

[109] L. Z. Cai, Y. K. Feng, J. Kawamoto, and K. Sakurai, "A behavior-based method for detecting DNS amplification attacks," in *Proc. Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.*, 2016, pp. 608–613.

[110] T. Böttger *et al.*, "DoS amplification attacks—Protocol-agnostic detection of service abuse in amplifier networks," in *Proc. Int. Workshop Traffic Monitor. Anal.*, 2015, pp. 205–218.

[111] H. Tsunoda *et al.*, "Detecting DRDoS attacks by a simple response packet confirmation mechanism," *Comput. Commun.*, vol. 31, no. 14, pp. 3299–3306, 2008.

[112] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, "Detecting DNS amplification attacks," in *Proc. Int. Workshop Crit. Inf. Infrastruct. Security*, 2007, pp. 186–196.

[113] D. Huistra, "Detecting reflection attacks in DNS flows," in *Proc. 19th Twente Student Conf. IT*, Feb. 2013.

[114] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Proc. IEEE Symp. Security Privacy*, 2013, pp. 127–141.

[115] T. Hirayama, K. Toyoda, and I. Sasase, "Fast target link flooding attack detection scheme by analyzing traceroute packets flow," in *Proc. IEEE Int. Workshop Inf. Forensics Security*, 2015, pp. 1–6.

[116] L. Xue, X. P. Luo, E. W. W. Edmond, and X. Zhan, "Towards detecting target link flooding attack," in *Proc. Usenix Conf. Large Install. Syst. Admin.*, 2014, pp. 81–96.

[117] C. Liaskos, V. Kotronis, and X. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[118] S. B. Lee, M. S. Kang, and V. D. Gligor, "CoDEF: Collaborative defense against large-scale link-flooding attacks," in *Proc. ACM Conf. Emerg. Netw. Exp. Technol.*, 2013, pp. 417–428.

[119] S. A. Aljawarneh, R. A. Moftah, and A. M. Maatuk, "Investigations of automatic methods for detecting the polymorphic worms signatures," *Future Gener. Comput. Syst.*, vol. 60, pp. 67–77, Jul. 2016.

[120] X. Y. Yang, Y. Shi, and H. J. Zhu, "Detection and location algorithm against local-worm," *Sci. China*, vol. 51, no. 12, pp. 1935–1946, 2008.

[121] W. Guo, L. Wang, and H. X. Zhou, "A behavior approach to instant messaging worm detection," in *Proc. Int. Conf. Artif. Intell. Ind. Eng. (AIIE)*, Phuket, Thailand, 2015.

[122] C. C. Zou, W. B. Gong, D. Towsley, and L. X. Gao, "The monitoring and early detection of Internet worms," *IEEE/ACM Trans. Netw.*, vol. 13, no. 5, pp. 961–974, Oct. 2005.

[123] C. C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for Internet worms," in *Proc. ACM Conf. Comput. Commun. Security*, Washington, DC, USA, 2003, pp. 190–199.

[124] W. Yu, X. Wang, P. Calyam, D. Xuan, and W. Zhao, "Modeling and detection of camouflaging worm," *IEEE Trans. Depend. Secure Comput.*, vol. 8, no. 3, pp. 377–390, May/Jun. 2011.

[125] B. Kim, H. Kim, and S. Bahk, "FDF: Frequency detection-based filtering of scanning worms," *Comput. Commun.*, vol. 32, no. 5, pp. 847–857, 2009.

[126] M. P. Collins and M. K. Reiter, "Hit-list worm detection and bot identification in large networks using protocol graphs," in *Proc. Int. Workshop Recent Adv. Intrusion Detect.*, 2007, pp. 276–295.

[127] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast IP networks," in *Proc. IEEE Int. Workshop Enabling Technol. Infrastruct. Collaborative Enterprise*, Linköping, Sweden, 2005, pp. 172–177.

[128] M. P. Stoecklin, J.-Y. L. Boudec, and A. Kind, "A two-layered anomaly detection technique based on multi-modal flow behavior models," in *Proc. 9th Int. Conf. Passive Active Meas.*, 2008, pp. 212–221.

[129] C. Gates, J. J. McNutt, J. B. Kadane, and M. I. Kellner, "Scan detection on very large networks using logistic regression modeling," in *Proc. 11th IEEE Symp. Comput. Commun.*, 2006, pp. 402–408.

[130] L. Bin, L. Chuang, Q. Jian, H. Jianping, and P. Ungsunan, "A NetFlow based flow analysis and monitoring system in enterprise networks," *Comput. Netw.*, vol. 52, no. 5, pp. 1074–1092, 2008.

[131] N. Muraleedharan and A. Parmar, "ADRISYA: A flow based anomaly detection system for slow and fast scan," *Int. J. Netw. Security Appl.*, vol. 2, no. 4, pp. 234–245, 2010.

[132] I. A. Farag, M. A. Shouman, T. S. Sobh, and H. Z. El-Fiqi, "Intelligent system for worm detection," *Int. Arab J. e-Technol.*, vol. 1, no. 1, pp. 58–67, 2009.

[133] W.-C. Sun and Y.-M. Chen, "A rough set approach for automatic key attributes identification of zero-day polymorphic worms," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 4672–4679, 2009.

[134] P. M. Comar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 2022–2030.

[135] N. Chatzis and R. Popescu-Zeletin, "Flow level data mining of DNS query streams for email worm detection," in *Proc. Int. Workshop Comput. Intell. Security Inf. Syst. (CISIS)*, 2008, pp. 186–194.

[136] N. Chatzis and N. Brownlee, "Similarity search over DNS query streams for email worm detection," in *Proc. IEEE Int. Conf. Adv. Inf. Netw. Appl.*, 2009, pp. 588–595.

[137] S. Abdulla, S. Ramadass, A. Altaher, and A. Al-Nassiri, "Employing machine learning algorithms to detect unknown scanning and email worms," *Int. Arab J. Inf. Technol.*, vol. 11, no. 2, pp. 140–148, 2014.

[138] D. Stopel, R. Moskovitch, Z. Boger, Y. Shahar, and Y. Elovici, "Using artificial neural networks to detect unknown computer worms," *Neural Comput. Appl.*, vol. 18, no. 7, pp. 663–674, 2009.

[139] W. Cui, R. H. Katz, and W.-T. Tan, "BINDER: An extrusion-based break-in detector for personal computers," in *Proc. USENIX Annu. Tech. Conf.*, 2005, pp. 363–366.

[140] J. Seo, S. Cha, B. Zhu, and D. Bae, "PC worm detection system based on the correlation between user interactions and comprehensive network behaviors," *IEICE Trans. Inf. Syst.*, vol. E96-D, no. 8, pp. 1716–1726, 2013.

[141] M. A. Ahmad, S. Woodhead, and D. Gan, "A countermeasure mechanism for fast scanning malware," in *Proc. Int. Conf. Cyber Security Protect. Digit. Services*, London, U.K., 2016, pp. 1–8.

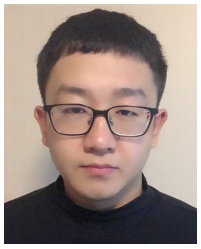
[142] K. Shahzad and S. Woodhead, "Towards automated distributed containment of zero-day network worms," in *Proc. Int. Conf. Comput. Commun. Netw. Technol.*, Hefei, China, 2014, pp. 1–7.

[143] S. Q. Chen, L. Liu, X. Y. Wang, X. W. Zhang, and Z. Zhang, "A host-based approach for unknown fast-spreading worm detection and containment," *ACM Trans. Auton. Adapt. Syst.*, vol. 8, no. 4, pp. 1–18, 2014.

[144] F. T. Xiao, H. P. Hu, B. Liu, and X. Chen, "PTBBWD: A fast process traffic behavior based worm detection algorithm," in *Proc. Int. Seminar Future Inf. Technol. Manag. Eng.*, 2008, pp. 181–186.

[145] Y. Tang and S. G. Chen, "An automated signature-based approach against polymorphic Internet worms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 7, pp. 879–892, Jul. 2007.

[146] L. J. Wang, Z. C. Li, Y. Chen, Z. Fu, and X. Li, "Thwarting zero-day polymorphic worms with network-level length-based signature generation," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 53–66, Feb. 2010.
[147] J. Wang, L. Hamadeh, G. Kesidis, and D. J. Miller, "Polymorphic worm detection and defense: System design, experimental methodology, and data resources," in *Proc. ACM SIGCOMM Workshop Large Scale Attack Defense*, Pisa, Italy, 2006, pp. 169–176.
[148] M. M. Rasheed, N. M. Norwawi, M. M. Kadhum, and O. Ghazali, "A new generation for intelligent anti-Internet worm early system detection," in *Proc. Int. Conf. Comput. Informat.*, 2009, pp. 61–72.
[149] X. H. Pan, X. S. Zhang, and T. Chen, "A novel hybrid method for polymorphic worm detection," in *Proc. Int. Conf. E-Bus. Inf. Syst. Security*, Wuhan, China, 2009, pp. 50–54.
[150] Y. F. Chen, Z. T. Xiang, Y. B. Dong, and D. M. Lu, "Cooperation system of worm detection and quarantine in real time," in *Proc. IEEE Int. Conf. Autom. Logistics*, Qingdao, China, 2008, pp. 1022–1026.
[151] M. M. Rasheed, N. M. Norwawi, O. Ghazali, and M. M. Kadhum, "Intelligent failure connection algorithm for detecting Internet worms," *Int. J. Comput. Sci. Netw. Security*, vol. 9, no. 5, pp. 280–285, 2009.
[152] M. Anbar, S. Ramadass, S. Manicka, and A. Al-Wardi, "Connection failure message-based approach for detecting sequential and random TCP scanning," *Indian J. Sci. Technol.*, vol. 7, no. 5, pp. 628–636, 2014.
[153] L. Braun, G. Müenz, and G. Carle, "Packet sampling for worm and botnet detection in TCP connections," in *Proc. IEEE-IFIP Netw. Oper. Manag. Symp.*, Osaka, Japan, 2010, pp. 264–271.
[154] F. Tong and Z. Yan, "A hybrid approach of mobile malware detection in Android," *J. Parallel Distrib. Comput.*, vol. 103, pp. 22–31, May 2016.
[155] S. X. Ma and Z. Yan, "PSNController: An unwanted content control system in pervasive social networking based on trust management," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 12, no. 1S, p. 17, 2015.
[156] L. Chen, Z. Yan, W. D. Zhang, and R. Kantola, "TruSMS: A trustworthy SMS spam control system based on trust management," *Future Gener. Comput. Syst.*, vol. 49, pp. 77–93, Aug. 2015.
[157] Y. Shen, Z. Yan, and R. Kantola, "Analysis on the acceptance of global trust management for unwanted traffic control based on game theory," *Comput. Security*, vol. 47, pp. 3–25, Nov. 2014.
[158] Z. Yan, R. Kantola, and Y. Shen, "A generic solution for unwanted traffic control through trust management," *New Rev. Hypermedia Multimedia*, vol. 20, no. 1, pp. 25–51, 2014.
[159] Z. Yan, R. Kantola, L. F. Zhang, and Y. T. Ma, "Unwanted traffic detection and control based on trust management," in *Information Fusion for Cyber-Security Analytics: Trends and Patterns*, I. Alsmadi, A. Aleroud, and G. Karabatis, Eds. Cham, Switzerland: Springer, 2016, pp. 77–109.
[160] L. F. Zhang, Z. Yan, and R. Kantola, "Privacy-preserving trust management for unwanted traffic control," *Future Gener. Comput. Syst.*, vol. 72, pp. 305–318, Jul. 2017.
[161] J. T. Yao, A. V. Vasilakos, and W. Pedrycz, "Granular computing: Perspectives and challenges," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1977–1989, Dec. 2013.
[162] H. Li, D. Y. Li, Y. H. Zhai, S. G. Wang, and J. Zhang, "A novel attribute reduction approach for multi-label data based on rough set theory," *Inf. Sci.*, vols. 367–368, pp. 827–847, Nov. 2016.
[163] W. H. Xu and J. H. Yu, "A novel approach to information fusion in multi-source datasets: A granular computing viewpoint," *Inf. Sci.*, vol. 378, pp. 410–423, Feb. 2017.
[164] R. Jensen and N. M. Parthalain, "Towards scalable fuzzy–rough feature selection," *Inf. Sci.*, vol. 323, pp. 1–15, Dec. 2015.



Xuyang Jing is currently pursuing the Ph.D. degree in cyberspace security with Xidian University, Xi'an, China. His research interests are in cyberspace security, granular computing, knowledge discovery, and data mining.

Zheng Yan (M'06–SM'14) received the B.Eng. degree in electrical engineering and the M.Eng. degree in computer science and engineering from Xi'an Jiaotong University, Xi'an, China, in 1994 and 1997, respectively, the second M.Eng. degree in information security from the National University of Singapore, Singapore, in 2000, and the Licentiate of Science and Doctor of Science in Technology degrees in electrical engineering from the Helsinki University of Technology, Helsinki, Finland, in 2005 and 2007, respectively. She is currently a Professor with Xidian University, Xi'an, and a Visiting Professor with Aalto University, Espoo, Finland. Her research interests are in trust, security, and privacy, social networking, cloud computing, networking systems, and data mining. She serves as an Associate Editor for *Information Sciences*, IEEE INTERNET OF THINGS JOURNAL, IEEE ACCESS JOURNAL, *Information Fusion*, JNCA, and *Security and Communication Networks*. She is a leading guest editor of many reputable journals including ACM TOMM, FGCS, IEEE SYSTEMS JOURNAL, and MONET. She served as a steering, organization, and program committee member for over 70 international conferences.

Witold Pedrycz (F'98) received the MS.c., Ph.D., and D.Sci. degrees from the Silesian University of Technology, Gliwice, Poland.

He is a Professor and the Canada Research Chair in computational intelligence with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is also with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland, where he is a Foreign Member. He has authored 15 research monographs covering various aspects of computational intelligence, data mining, and software engineering. His current research interests include computational intelligence, fuzzy modeling, and granular computing, knowledge discovery and data mining, fuzzy control, pattern recognition, knowledge-based neural networks, relational computing, and software engineering. He has published numerous papers in the above areas.

Dr. Pedrycz was a recipient of the prestigious Norbert Wiener Award from the IEEE Systems, Man, and Cybernetics Society, in 2007, and the IEEE Canada Computer Engineering Medal, the Cajastur Prize for soft computing from the European Center for Soft Computing, the Killam Prize, and the Fuzzy Pioneer Award from the IEEE Computational Intelligence Society. He is currently a member of number of Editorial Boards of other international journals. He is a member of numerous program committees of the IEEE conferences in the area of fuzzy sets and neurocomputing. He is intensively involved in editorial activities. He currently serves on the Advisory Board of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He is the Editor-in-Chief of *Information Sciences*, *WIREs Data Mining and Knowledge Discovery* (Wiley), and the *International Journal of Granular Computing* (Springer). He was a fellow of the Royal Society of Canada, Ottawa, ON, Canada, in 2012.