Application Layer Denial-of-Service Attacks and Defense Mechanisms: A Survey

NIKHIL TRIPATHI, Fraunhofer Institute for Secure Information Technology
NEMINATH HUBBALLI, Indian Institute of Technology Indore

Application layer **Denial-of-Service (DoS)** attacks are generated by exploiting vulnerabilities of the protocol implementation or its design. Unlike volumetric DoS attacks, these are stealthy in nature and target a specific application running on the victim. There are several attacks discovered against popular application layer protocols in recent years. In this article, we provide a structured and comprehensive survey of the existing application layer DoS attacks and defense mechanisms. We classify existing attacks and defense mechanisms into different categories, describe their working, and compare them based on relevant parameters. We conclude the article with directions for future research.

CCS Concepts: • **Security and privacy** → **Denial-of-service attacks**; **Intrusion detection systems**; *Firewalls*; • **Networks** → **Application layer protocols**; Protocol testing and verification

Additional Key Words and Phrases: Protocol-specific and generic DoS attacks, distributed DoS attacks, defense mechanisms

ACM Reference format:
Nikhil Tripathi and Neminath Hubballi. 2021. Application Layer Denial-of-Service Attacks and Defense Mechanisms: A Survey. *ACM Comput. Surv.* 54, 4, Article 86 (April 2021), 33 pages.
https://doi.org/10.1145/3448291

1 INTRODUCTION

Denial-of-Service (DoS) attacks and its variant, **Distributed Denial-of-Service (DDoS)** attacks, have been a matter of serious concern for network administrators for the past two decades [11, 169]. These attacks intend to exhaust the resources (memory, CPU cycles, and network bandwidth) and render them unavailable for benign users, thereby violating one of the major components of cybersecurity—*Availability*. Launching a DoS attack typically requires less bandwidth from the malicious client's perspective and thus can be launched using a very small number of devices. However, launching a DDoS attack requires sending a flood of packets to the victim. A malicious client can launch a DDoS attack using two methods. In the first method, the malicious client sends the flood of packets using spoofed IP addresses (e.g., amplification/reflection attacks [126, 169]). In the second method, the malicious client controls a large number of bots that are compromised using

This work was carried out while N. Tripathi was with IIT Indore.
Authors' addresses: N. Tripathi, Fraunhofer Institute for Secure Information Technology, Darmstadt, Rheinstraße 75, Darmstadt, Hessen, 64295, Germany; email: nikhil.tripathi@sit.fraunhofer.de; N. Hubballi, Indian Institute of Technology Indore, Khandwa Road, Indore, Madhya Pradesh, 453552, India; email: neminath@iiti.ac.in.

0360-0300/2021/04-ART86 $15.00
https://doi.org/10.1145/3448291

malware (e.g., [30]) and commands these bots to send the flood of packets to the victim. Hacking groups have several motives behind launching these attacks. These range from simply getting recognition in the underground communities to the incentives given by organizations to launch these attacks against their potential competitors in the market. The well-known network/transport layer DDoS attacks [169] target network equipment and infrastructure to disturb their victim's connectivity. These attacks have been known to the community for quite a long time, and thus several surveys have been published that discuss these attacks [41, 101, 116, 169].

Lately, a new class of DoS attacks known as application layer DoS attacks [120] has started gaining popularity. These attacks exploit potential flaws and vulnerabilities present in the application layer protocols [16, 59, 94, 137]. These vulnerabilities are often consequences of inadequate efforts by designers and developers toward secure protocol development. Designing secure protocols is not always considered equally important compared to functionality (e.g., **Hypertext Transfer Protocol (HTTP)** 2.0 vs 1.1 [155]). This leaves behind a large attack surface that is then used by adversaries to launch application layer DoS attacks. These attacks can bring down a server with huge computational power and network bandwidth using very limited resources. Most application layer DoS attacks typically require only one computer to create a DoS scenario at the victim side. Thus, these attacks are usually considered as belonging to the DoS category (instead of the DDoS category) unless stated otherwise. The application layer DoS attacks target specific services on the victim with minimal implication on network resources [59, 94]. These attacks intend to prevent either a server from serving legitimate clients or the individual clients from accessing a resource available at a server. For example, Slow Rate DoS attacks against HTTP [50] intend to prevent a web server from serving the requests sent by genuine HTTP clients [169]. However, application layer DoS attacks against **Network Time Protocol (NTP)** prevent individual clients from synchronizing their clock with an NTP server [90, 91]. The application layer DoS attacks are known to generate less traffic as compared to network/transport layer DDoS attacks, and thus they are stealthier. Mitigating these attacks by modifying the operation of a protocol is not feasible, as it requires modifications in the corresponding RFCs, which is a cumbersome process and needs deliberations and discussions between stakeholders, which takes a substantial amount of time. Moreover, the reflection of these changes in the protocol implementations and releasing their newer versions in the wild by different vendors also take a long time.

Recent trends and incidents. According to the *2019 Global DDoS Threat Landscape Report* of Imperva [72], the largest application layer DoS attack in history was recorded in 2019. This attack lasted for 13 days and peaked at 292,000 requests per second. Moreover, as per another report [70], the number of application layer DoS/DDoS attacks is doubling after every quarter of a year, although the number of network layer assaults in the fourth quarter of 2017 was decreased by a huge margin of 50% from the third quarter of 2017. The popularity of application layer DoS attacks is reflected from another fact that there have been several incidents as shown in Table 1 wherein these attacks were encountered. Besides such reported incidents of DoS/DDoS attacks, there have been several incidents of these attacks due to insider threats that are not reported by the victim organizations fearing negative publicity [150]. In addition, several popular application layer DoS attacks such as **Dynamic Host Configuration Protocol (DHCP)** starvation attacks are launched within a local network and thus are not reported. According to a 2018 Insider Threat report [150], 53% of the organizations confirmed insider attacks against them just in the past year.

Besides these incidents, security researchers have recently scrutinized protocols such as HTTP 2.0 [1, 155], DHCP [67, 152, 154], NTP [90–92], and the **Domain Name System (DNS)** [126, 160], among others, to explore potential vulnerabilities in these protocols that can be exploited to launch DoS attacks. To counter these attacks, researchers have also recently proposed appropriate defense mechanisms [37, 67, 92, 153–155]. These recent attacks and corresponding detection and mitigation

Table 1. Application Layer DDoS Attack Incidents

Year	Target	Scale	Attack	Impact
2020 [124]	A state voter registration site	≈ 200,000 DNS requests	DNS flood	Not disclosed
2019 [72]	An Imperva client (name not disclosed)	292,000 requests per second	Not disclosed	Not disclosed
2019 [100]	National Union of Journalists of the Philippines website	76 Gbps	HTTP flood	Website went offline
2018 [63]	Three banks: ABN AMRO, ING and Rabobank	Not disclosed	Not disclosed	Disruption of mobile banking services
2017 [113]	Bitcoin gold website	10M requests per minute	HTTP flood	The website went down
2017 [109]	Spanish government websites	Not disclosed	HTTP flood	Websites of the constitutional court were taken offline
2017 [9]	Swedish transportation services	Not disclosed	HTTP flood	Attack caused train delays and disruption of travel services
2016 [7]	Finland heating systems	Not disclosed	DNS flood	The heating systems stopped working
2016 [129]	HSBC bank website	Not disclosed	HTTP flood	Attack mitigated successfully
2016 [122]	Rio Olympic games websites	Not disclosed	HTTP flood	Several Rio Olympic Games websites denied access to legitimate clients
2016 [161]	DYN's DNS infrastructure	Not disclosed	DNS flood	Popular websites such as Etsy, GitHub, Spotify, and Twitter suffered service interruptions or went offline altogether
2016 [95]	Liberia's Internet infrastructure	Not disclosed	Not disclosed	The Internet in the country went down
2015 [5]	GitHub's website	Not disclosed	HTTP flood	GitHub managed to overcome the attack
2014 [52]	Hong Kong media website	250M DNS requests per second	DNS flood	Campaigning for a democratic voting system affected
2012 [26]	Web infrastucture of several banks	63.3 Gbps	HTTP flood	Access to online and mobile banking services was affected
2009 [71]	Iranian presidential election campaign	Not disclosed	Slowloris	Not disclosed

schemes are not covered in the previous surveys as discussed later in Section 2. Therefore, we argue that a survey covering a comprehensive study of attacks against the most commonly targeted application layer protocols and the defense mechanisms is of critical importance to understand the state of the art in this domain.

In this article, we present a structured survey of various application layer DoS attacks. We also review various state-of-the-art defense mechanisms known to counter attacks against these protocols. We make the following specific contributions in this article:

(1) We present a comprehensive survey of application layer DoS attacks and classify them depending on whether they are effective against a specific protocol or a large number of protocols.
(2) We discuss the working of attacks and compare them based on different parameters. We also describe tools/utilities/libraries that can be used to launch the attacks.
(3) We review various defense mechanisms known to counter different application layer DoS attacks and classify them based on their working principle. We also describe the strengths and weaknesses of each defense mechanism and compare them based on several parameters.
(4) We compare several popular commercial DoS mitigation products based on their ability to counter different attacks.
(5) Toward the end of the article, we provide various directions for future research in this domain.

The article is organized as follows. In Section 2, we review other related surveys and compare our work with them, justifying the motivation. The organization of the rest of this article closely follows Figure 1. We discuss protocol-specific application layer DoS attacks and defense mechanisms known to counter these attacks in Section 3. In Section 4, we describe various generic application layer DoS attacks and the defense mechanisms known to counter them. Based on a variety of parameters, we compare different application layer DoS attacks and defense mechanisms in Section 5. We present a comparative study of several popular commercial DoS mitigation products based on their ability to counter different attacks in Section 6. In Section 7, we conclude the article and suggest some promising future research directions through a careful analysis of the research gap in this domain.

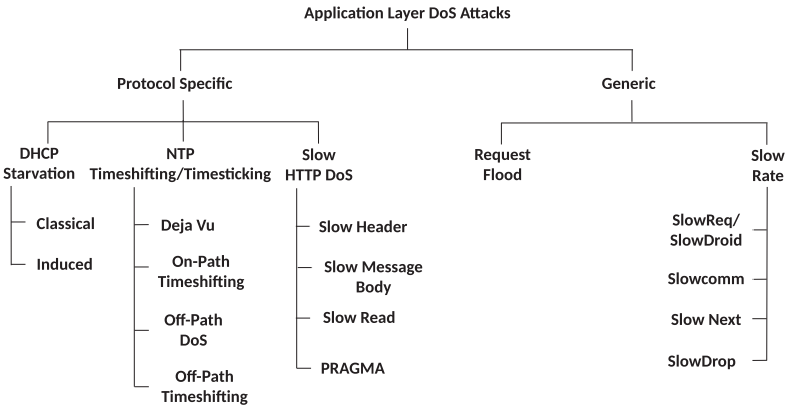

Fig. 1. Categorization of application layer DoS attacks.

2 PRIOR SURVEYS

DoS attacks, in general, have been known to the community for quite a long time, and therefore several works including surveys have been published in the literature. There are few survey works [101] in the literature wherein the authors presented a taxonomy of different types of DoS attacks. Zargar et al. [169] presented a comprehensive analysis of various defense mechanisms to counter different attacks such as HTTP flooding attacks and Slowloris attacks. Some of the surveys [45, 54, 80] discussed application layer DoS attacks against SIP [61]. In two other recent studies [119, 136], the authors attempted to discuss the application layer DoS attacks against HTTP and other web-related protocols. The authors also discussed the defense mechanisms known in the literature to counter these attacks. Our survey differs from the previous surveys in view of the following points:

(1) Mirkovic and Reiher [101] presented a taxonomy and survey of application layer DoS attacks but did not discuss the known defense mechanisms to counter those attacks. However, we not only present a comprehensive survey of different application layer DoS attacks but also discuss the detection and mitigation performance of various defense mechanisms to counter each attack type.
(2) Unlike Zargar et al. [169], who primarily focus on flooding-based DDoS attacks only, we cover different aspects of non-volumetric attacks and discuss how these attacks exploit the working of different protocols.[1] Moreover, the survey presented by Zargar et al. [169] was published more than 5 years ago and thus does not cover recent application layer DoS attacks and defense mechanisms.
(3) Geneiatakis et al. [54], Keromytis [80], and Ehlert et al. [45] discussed attacks and defense mechanisms related to SIP only. Recently, Praseed and Thilagam [119] covered different application layer DDoS attacks that target web infrastructure. However, we present a detailed analysis of DoS attacks and corresponding defense mechanisms by taking several application layer protocols into account.
(4) The survey on Slow HTTP DoS attacks and HTTP-GET flood DDoS attacks is presented in by Singh et al. [136]. However, in this survey, we not only take these attacks into account but also discuss a variety of other protocol-specific and generic DoS attacks.

[1]We do not cover attacks that exploit vulnerabilities present in a protocol's specific software implementation, as security updates released by vendors from time to time appropriately patch these vulnerabilities.

Details about each, as well as reasons

Table 2. Comparison with Some of the Existing Surveys

References	Starvation	Timeshifting/ Timesticking	Slow HTTP	Slow Rate	Request Flood	Corresponding Defense Mechanisms
Praseed and Thilagam [119]	✗	✗	✓	✗	✓	✓
Singh et al. [136]	✗	✗	✓	✗	✓	✓
Keromytis [80]	✗	✗	✗	✗	✓	✓
Zargar et al. [169]	✗	✗	✓	✗	✓	✓
Geneiatakis et al. [54]	✗	✗	✗	✗	✓	✗
Ehlert et al. [45]	✗	✗	✗	✗	✓	✓
This survey	✓	✓	✓	✓	✓	✓

A comparison of the existing surveys on application layer DoS attacks and corresponding defense mechanisms with our survey is given in Table 2. A tick mark (✓) in the table represents that a survey work discusses the attack or defense mechanism. We can notice from the table that most of the previous survey works focused only on request flood attacks but did not review other application layer DoS attacks. Taking motivation from this, in this survey we present a systematic review of different application layer DoS attacks and corresponding defense mechanisms.

3 PROTOCOL-SPECIFIC DOS ATTACKS

Application protocol, not protocol in general (e.g. TCP)

A protocol-specific DoS attack is effective against a particular application layer protocol only. Examples of protocol-specific attacks are DHCP starvation attacks, NTP timeshifting/timesticking DoS attacks, and Slow HTTP DoS attacks as shown in Figure 1. We describe these attacks and the corresponding defense mechanisms to counter them in this section.

3.1 DHCP Starvation Attacks

DHCP [43] is used to automate the IP address allocations to the clients within a network. A client, on joining the network, exchanges four DHCP messages viz. `Discover`, `Offer`, `Request`, and `Acknowledgment` with the DHCP server to obtain an IP address. The DHCP server is configured with an IP address pool from which it selects one freely available address and offers it to the client. Subsequently, the client uses the offered IP address to access network resources.

DHCP is known to be vulnerable to DHCP starvation attacks due to the lack of any built-in authentication scheme [152]. These attacks intend to prevent a legitimate client from obtaining an IP address. In Figure 1, two types of DHCP starvation attacks are shown that are known in the literature: classical and induced DHCP starvation attacks. Classical DHCP starvation attack has been known in the community for quite a long time; however, recently a new type of starvation attack called *induced DHCP starvation attack* [67, 152, 154] was disclosed.

3.1.1 Classical DHCP Starvation Attack. This attack involves sending bogus requests repeatedly (using spoofed random **Message Authentication Code (MAC)** addresses [158, 159]) to consume the IP address pool [67]. Once the pool is consumed, the DHCP server is unable to offer IP addresses to the new clients. As a result, the clients cannot communicate with other devices, which leads to DoS. Recently, Hubballi and Tripathi [67] showed that since this attack requires MAC spoofing, and it raises the bar for the malicious client to launch this attack in wireless networks secured with Wireless Protection Access 2.

3.1.2 Induced DHCP Starvation Attack. Similar to the classical DHCP starvation attack, induced DHCP starvation attack also prevents clients from obtaining an IP address. This attack exploits IP conflict detection schemes present at server and client sides. Using this scheme, a DHCP server first ensures that an IP address it is going to offer to a client is not being used by some other client in the network. Similarly, the client before using the offered IP address also checks that no other client

is using that IP address [43]. For conflict detection purpose, a broadcast probe request is sent into the network. To prevent the client from obtaining the IP address, a malicious client simply injects a spoofed probe reply. As a result, IP conflict [43] is detected in the network, and the DHCP client is not able to successfully obtain the IP address. Depending on which (client's or server's) conflict detection scheme is exploited and what type of probe is used (**Address Resolution Protocol (ARP)** or Internet Control Message Protocol), there are three different variants [67, 152, 154] of induced DHCP starvation attack.

Tools. There are a plethora of tools that can be used to launch classical DHCP starvation attack. Among them, the most popular ones are Gobbler [56] and DHCpig [35]. However, there are no publicly available tools to launch an induced DHCP starvation attack. Nevertheless, packet crafting tools such as Scapy [128] can be used to launch this attack. Scapy runs natively on Linux and most Unix operating systems with libpcap and its Python wrappers.

3.1.3 Defense Mechanisms. Although works on induced DHCP starvation attack [67, 152, 154] have been published recently, classical DHCP starvation attack has been known to the security community for a long time. Thus, several researchers have proposed methods to detect and defend against classical DHCP starvation attack. We classify the known defense mechanisms into different categories and subsequently discuss their ability to counter classical and induced DHCP starvation attacks.

Cryptographic Techniques. To mitigate the DHCP starvation attacks, various cryptographic techniques [32, 33, 44, 144] are proposed in the literature that enforce authentication in DHCP. These techniques are well known to mitigate any spoofing attack including the DHCP starvation attacks, but due to high deployment complexity and manual intervention of network administrators for managing certificates, these techniques are rarely deployed in real networks.

Security Features in Switches. Security features such as port security [25] and **Dynamic ARP Inspection (DAI)** [24] can be configured in state-of-the-art network switches to mitigate different network attacks. Port security restricts the count of MAC addresses seen on a switch port. Thus, if the number of MAC addresses seen on a port exceeds a pre-defined threshold, it is immediately blocked, thereby preventing MAC spoofing. DAI checks the validity of an ARP [117] message to prevent ARP spoofing by querying DHCP snooping database that contains valid IP-MAC bindings. However, Hubballi and Tripathi [67] showed that DAI fails to mitigate ARP spoofing (and thus, induced DHCP starvation) if the spoofed ARP reply sent by the malicious client does not pass through the switch [67].

Using DHCP Relay Agent. Patrick [115] proposed an approach wherein a DHCP relay agent (default gateway) puts the switch ID and switch port number (to which a client is connected) in a DHCP message and forwards it to the DHCP server. Subsequently, the DHCP server counts the IP addresses allotted to the client by matching the switch ID and port number and ensures that this count does not cross a pre-defined threshold, thereby preventing DHCP starvation attacks.

DHCP Traffic Profiling. Defense mechanisms falling under this category compare traffic profiles generated during different time intervals of testing and training phases. An abstract overview of the working of these defense mechanisms is shown in Figure 2. In this direction, Hubballi and Tripathi [67] proposed a detection approach wherein a normal DHCP traffic profile is generated during the training phase and subsequently compared with the traffic profiles generated during different time intervals of the testing phase using **Hellinger Distance (HD)** [19]. If the calculated HD exceeds a pre-defined threshold, the time interval under consideration is declared as containing attack traffic. In another work, Tripathi and Hubballi [153] presented a detection approach that

anomaly detection in traffic

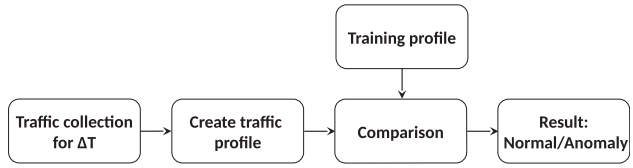

Fig. 2. Working of DHCP traffic profiling–based defense mechanisms.

Table 3. Defense Mechanisms vs DHCP Starvation Attacks

Defense Mechanisms		Classical [67]	Induced: 1st Variant [67]	Induced: 2nd Variant [152]	Induced: 3rd Variant [154]
Cryptographic Techniques		✓	✓	✓	✓
Security Features in Switches	Port Security	✗	✗	✗	✗
	DAI	✓	✗	✗	✗
Using DHCP Relay Agent		✓	✗	✗	✗
DHCP Traffic Profiling		✓	✓	✓	✗
Machine Learning based		✗	✗	✗	✓

Table 4. Approaches to Counter DHCP Starvation Attacks

Category	Research Work	Defense Mechanism	Strength	Weakness
Cryptographic Techniques	Graaf et al. [32], Demarjian and Serhrouchni [33], Duangphasuk et al. [44], Su et al. [144]	Mitigation by enforcing authentication in DHCP	Can mitigate all types of identity spoofing–based attacks	Very high implementation complexity
Security Features in Switches	Port Security [25]	Mitigation by blocking a switch port if more than the threshold number of MAC addresses are seen	Easy to deploy	Cannot mitigate induced DHCP starvation attack [154]
	DAI [24]	Mitigation by dropping packets based on MAC-IP associations present in DHCP snooping database	Easy to deploy	Cannot mitigate induced DHCP starvation attack [154]
Using DHCP Relay Agent	Patrick [115]	Mitigation by putting a limit on the number of IP addresses assigned to a client	Can detect and mitigate classical DHCP starvation attack	Cannot mitigate induced DHCP starvation attack
DHCP Traffic Profiling	Hubballi and Tripathi [67]	Detection by comparing DHCP traffic profiles using HD	Lightweight	Cannot detect classical DHCP starvation attack
	Tripathi and Hubballi [153]	Detection by predicting the count of DHCP messages in different time intervals	Can detect both classical and induced DHCP starvation attack	Difficult to set a pre-defined threshold
Machine Learning based	Tripathi and Hubballi [154]	Detection using machine learning algorithms	Easy to deploy	Cannot detect classical DHCP starvation attack

simply compares the number of different DHCP messages received (at the detector within the network) in a time period (e.g., 10:00 am to 10:15 am) of a day during training and testing phases. If the difference in the number of messages exceeds a preset threshold, an anomaly is detected in that period.

Machine Learning based. Tripathi and Hubballi [154] proposed a detection approach wherein they tested the ability of different one-class classification algorithms to detect variants of the induced DHCP starvation attack. However, the authors did not test the performance of the algorithms to detect classical DHCP starvation attack.

Table 3 shows the ability of various defense mechanisms to detect/mitigate different DHCP starvation attacks, whereas Table 4 summarizes different categories of defense mechanisms along with their strengths and limitations.

3.2 Timeshifting/Timesticking DoS Attacks Against NTP

NTP [99] is one of the oldest protocols currently being used on the Internet. It was designed to synchronize clocks of different computer systems connected to the Internet. Malhotra et al.

[90–92] recently disclosed timeshifting/timesticking attacks that target the working of NTP protocol and aim to disturb the clock synchronization process of computers with NTP servers. A wrongly synchronized clock may lead to failure of various core Internet services such as DNS and Resource Public Key Infrastructure [13], which results in DoS at a large scale. As shown in Figure 1, different types of known attacks against NTP are presented next.

3.2.1 Deja Vu (On-Path Timesticking Attack). To launch this attack [91], a **Man-in-the-Middle (MitM)** malicious client between an NTP broadcast server and a victim client first captures a contiguous sequence of broadcast packets sent from an NTP server and then replays this sequence at regular intervals to the victim client. This causes the victim's clock to be stuck at a particular time, which results in the failure of core Internet services on the victim.

3.2.2 On-Path Timeshifting Attack. To launch this attack [90], a malicious client sends bogus small time shifts to a victim client, then when the malicious client is ready, it sends a big time shift to the victim client to create DoS.

Very detailed descriptions here

3.2.3 Off-Path DoS Attacks. Malhotra et al. [90–92] presented three variants of off-path DoS attacks against NTP wherein a malicious client irrespective of its location can prevent clock synchronization of an NTP client connected to the Internet.

DoS by Spoofed Kiss-o'-Death [90]. NTP defines a rate-limiting mechanism using a special purpose packet known as the **Kiss-o'-Death (KoD)** packet. This packet is sent by an NTP server to a client if the client sends NTP packets to the server with a rate that exceeds a pre-defined threshold rate. To launch this attack variant, an off-path malicious client exploits this rate-limiting mechanism by sending a spoofed KoD packet to the victim NTP client for each of the client's pre-configured NTP servers. On receiving KoD packets, the client immediately stops sending NTP packets to the servers, due to which it is not able to synchronize its local clock. In another recent work, Tripathi and Hubballi [156] proposed an attack that prevents an NTP broadcast client from synchronizing its clock with an NTP server. To prevent this attack, the malicious client forces the NTP broadcast server to generate genuine KoD packets by sending it a large number of NTP packets that are spoofed to look like they are coming from the victim client. However, this attack is relatively less stealthy, as it requires sending several spoofed NTP packets.

DoS by Bad Authentication [91]. In this variant, an off-path adversary targets a victim NTP client that relies on a broadcast NTP server to synchronize its local clock. The NTP client and the broadcast server usually authenticate each other using a pre-configured key. To launch this variant, the adversary simply sends to the victim a badly authenticated NTP broadcast packet that is spoofed to look like it is coming from the broadcast NTP server. Due to this bad authenticated packet, the victim client breaks its association with the broadcast server.

Interleaved-Pivot Attack [92]. In this variant, a malicious client first sends a single packet to the victim client that is spoofed to look like it is coming from the NTP server. On receiving this packet, the victim client believes that the server is in interleaved mode (a very high accurate mode of NTP operation) [99]. Due to this, the victim client also enters into the interleaved mode and subsequently rejects all legitimate client/server mode packets exchanged with the server (which is actually not in the interleaved mode).

3.2.4 Off-Path Timeshifting Attacks. Malhotra et al. [90] presented two variants of off-path timeshifting attacks against NTP. To launch the first attack variant, Pinning to Bad Timekeepers, the malicious client sends a KoD packet to the victim client that is spoofed to look like it is coming from the legitimate NTP server to which the victim client has correctly synchronized. The malicious client does so to prevent the client from synchronizing its clock with the synchronized

server and ultimately forcing it to take time from an NTP server not having an up-to-date clock. To launch the second attack variant, DoS using IP Fragmentation, the malicious client hijacks an unauthenticated NTP connection established between a client and a server by exploiting certain IPv4 fragmentation policies used by the client's and server's operating systems.

Tools. To launch these attacks, there are no specific tools available. Nevertheless, Scapy [128] and other similar libraries providing raw socket programming support can be used to execute these attacks.

3.2.5 Defense Mechanisms. Since launching NTP timeshifting/timesticking DoS attacks requires spoofing the IP address of an NTP client or server, schemes that are known to detect spoofed IP packets can also be deployed to detect these attacks [68, 162, 169]. Zargar et al. [169] presented a comprehensive study of different IP spoofing mitigation approaches. In this survey, we cover only those defense mechanisms that are specifically designed to counter DoS attacks against NTP. We classify defense mechanisms known to counter attacks against NTP into different categories and discuss their ability to counter the attacks:

Cryptographic Techniques. Some approaches [42, 92] use cryptographic techniques to authenticate NTP packets to mitigate the known attacks. However, though NTP supports cryptographic authentication, NTP traffic is very rarely authenticated in practice [90] for various reasons, such as a cumbersome key distribution mechanism and weaknesses in the Autokey protocol for public-key authentication [103]. This leads to the development of NTPsec [85, 151]. Unfortunately, the adoption of NTPsec is still very slow, and moreover, authentication and encryption do not mitigate MitM attacks, as an MitM adversary can simply drop traffic destined to port 123 (default for NTP).

Path Redundancy. A class of work in the literature utilizes path redundancy on the Internet to avoid MitM adversaries [102, 134]. Under this approach, multiple paths on the Internet are used to connect NTP clients and servers. Thus, even if one of the paths between an NTP client and a server is compromised, the client is able to synchronize its clock by exchanging NTP packets over other paths. The drawback of this approach is that it cannot mitigate attacks that do not require a MitM position (e.g., off-path DoS).

Server Redundancy. Deutsch et al. [34] proposed a new NTP client challed *Chronos* that first generates server redundancy by creating large server pools and then carefully samples servers in these pools. Since Chronos synchronizes its clock with the help of large server pools, even a malicious client with an MitM position cannot stop the client from obtaining correct time information from other servers. As a result, this approach is resilient to on-path attacks. In the case of off-path attacks, Chronos' clock selection algorithm rejects wrong time information received from a malicious adversary, as it differs from the clock values of servers present in the pool. Thus, Chronos is resilient to off-path attacks as well, thereby overcoming the limitations of multi-path approaches proposed in other works [102, 134].

Table 5 shows the detection/mitigation ability of various defense mechanisms to counter attacks against NTP, whereas Table 6 summarizes different categories of defense mechanisms along with their strengths and limitations.

3.3 Slow HTTP DoS Attacks

HTTP is considered as one of the most studied application layer protocols against DoS attacks. Several types of Slow HTTP DoS attacks have been studied over the past decade. Slow HTTP DoS attacks [157, 169] involve sending specially crafted web requests that interact with the server very slowly. Unless a web request is completely served, the server keeps it in a connection queue with

Explanatory summary tables

Table 5. Defense Mechanisms vs Different Attacks Against NTP

Defense Mechanisms	On-Path Timesticking [91]	On-Path Timeshifting [90]	Off-Path DoS-1 [90]	Off-Path DoS-2 [91, 156]	Off-Path DoS-3 [92]	Off-Path Timeshifting-1 [90]	Off-Path Timeshifting-2 [90]
Cryptographic Techniques	✗	✗	✓	✓	✓	✓	✓
Path Redundancy	✓	✓	✗	✗	✗	✗	✗
Server Redundancy	✓	✓	✓	✓	✓	✓	✓

Table 6. Approaches to Counter Different Attacks Against NTP

Category	Research Work	Defense Mechanism	Strength	Weakness
Cryptographic Techniques	Malhotra et al. [92]	Mitigation using a new backward-compatible client/server protocol for NTP	Can mitigate all types of identity spoofing–based attacks	Difficult to deploy
	Dowling et al. [42]	Mitigation by infrequently performing a key exchange using public key cryptography and then relying solely on symmetric cryptography	Suitable for large-scale deployments	Creates an attack surface for NTP amplification attack
Path Redundancy	Mizrahi [102] and Shpiner et al. [134]	Mitigation by establishing multiple paths between NTP clients and servers	Can detect and mitigate on-path attacks	Ineffective in case of off-path DoS attack
Server Redundancy	Deutsch et al. [34]	Mitigation by preventing the client from taking time from a compromised server	Can mitigate both on-path and off-path attacks	Requires modification at the client-side, which may hinder its adaptation

limited space. Once this queue is filled with unserved requests, the server does not entertain any more requests, resulting in a DoS attack. The DoS attacks belonging to this category are shown in Figure 1 and are as follows.

3.3.1 Slow Header or Slowloris Attack. To launch the Slow Header attack [157, 169], an adversary sends incomplete HTTP GET requests to a web server to exhaust the connection queue space. Tripathi et al. [157] evaluated this attack against 100 live websites and showed that several of them are vulnerable to it. Tripathi and Hubballi [155] proposed an attack similar to the Slow Header attack against HTTP 2.0 and showed that the attack is effective against different popular web servers.

3.3.2 Slow Message Body Attack. Slow Message Body attack [157, 169] requires sending a complete header of POST request but an incomplete message body. Thus, the server keeps these requests in the connection queue unless it receives a complete message body. Tripathi et al. [157] presented an evaluation of this attack against four web server implementations and showed that the attack is effective against three of them. Tripathi and Hubballi [155] proposed an attack similar to the Slow Message Body attack against HTTP 2.0 and showed that the attack is effective against different popular web servers.

3.3.3 Slow Read Attack. To launch Slow Read attack [73], an adversary initially sends a benign GET request to a web server and then sends a TCP packet advertising window size of zero bytes. This prevents the server from sending any data to the adversary. Nevertheless, the server stores the connection in connection queue space and waits to receive from the adversary a TCP packet advertising "non-zero window size" [157]. The adversary, however, does not advertise a new window size due to which the server waits for an indefinite amount of time. Tripathi and Hubballi [155] tested a similar attack against HTTP 2.0 and showed that the attack is effective against different popular web servers.

3.3.4 HTTP PRAGMA Attack. HTTP PRAGMA attack [31] exploits a field called *PRAGMA* in the HTTP header. This field is used by the browsers to request the web server to send a resource against which the client has already requested earlier. On receiving such a request, the server resets the connection timeout due to which the connection stays longer in the connection queue.

Table 7. Defense Mechanisms vs Slow HTTP DoS Attacks

Defense Mechanisms		Slow Header [142]	Slow Message Body [157]	Slow Read [73]	HTTP PRAGMA [31]
Implement-ation Modules	Core	✓	✓	✗	✗
	Antiloris	✓	✓	✓	✓
	Limitipcon	✓	✓	✓	✓
	mod_reqtimeout	✓	✓	✗	✗
Comparing Traffic Profiles		✓	✓	✗	✗
Monitoring Traffic Features		✓	✓	✓	✓

The malicious client creates several such connections and sends the PRAGMA requests from each of them to exhaust the server's connection queue space, thereby preventing legitimate requests from being served.

Tools. Slowloris [142] and SlowHTTPTest [141] are two such tools that can be used to generate Slow Header attack. R U DEAD YET (RUDY) [127] is a popular tool that can be used to launch a Slow Message Body attack. However, there is no tool available to launch Slow Read and HTTP PRAGMA attack. Nevertheless, special requests can be crafted using Scapy [128] to launch these attacks.

3.3.5 Defense Mechanisms. We classify defense mechanisms to counter Slow HTTP DoS attacks into three different categories as described next.

Implementation Modules. To mitigate Slow HTTP DoS attacks, four modules—Core [27], Antiloris [104], Limitipconn [105], and mod_reqtimeout [106]—are particularly available for the Apache server. The first module, Core, buffers entire HTTP requests at the kernel level before forwarding them to the server. This ensures that incomplete HTTP requests do not go to the server's connection queue. However, the second and third modules limit the number of incoming complete/incomplete requests on a per-IP basis, whereas the fourth module requires a sender to send a complete request and/or complete message body within a pre-defined amount of time.

Comparing Traffic Profiles. Tripathi et al. [157] proposed an approach wherein a normal HTTP traffic profile is generated during the training phase and subsequently compared with the traffic profiles generated during different time intervals of the testing phase using HD [19]. The traffic profile is comprised of four vectors denoting the probability of occurrence of incomplete and complete POST and GET requests. If the calculated HD crosses a pre-defined threshold, the time interval under consideration is declared as containing attack traffic. In another work, Tripathi and Hubballi [155] proposed a similar approach to detect different Slow Rate DoS attacks against HTTP 2.0.

Monitoring Traffic Features. A class of work in the literature proposes to monitor different traffic features to detect Slow HTTP DoS attacks. One such approach is discussed in the work of Dantas et al. [31] that detects attacks by monitoring the number of bytes the server sends and receives for each request. If the server runs out of connection queue space, it drops a connection over which the number of sent and received bytes exceeds a pre-defined threshold.

Table 7 shows the detection/mitigation ability of various defense mechanisms to counter Slow HTTP DoS attacks, whereas Table 8 summarizes different categories of defense mechanisms along with their strengths and limitations.

4 GENERIC DOS ATTACKS

Unlike protocol-specific attacks, this class of application layer DoS attacks is effective against a large number of application layer protocols. As shown in Figure 1, examples of generic application layer DoS attacks are Slow Rate Generic and Request Flood attacks, which are effective against a large number of popular protocols such as HTTP, **Simple Mail Transfer Protocol (SMTP)**, **File**

Table 8. Approaches to Counter Slow HTTP DoS Attacks

Category	Research Work	Defense Mechanism	Strength	Weakness
Implementation Modules	Core [27]	Mitigation by preventing incomplete requests from going to the server's connection queue	Can be easily configured in the Apache web server	Cannot mitigate HTTP PRAGMA attack
	Antiloris [104], Limitipconn [105]	Mitigation by limiting the number of incoming complete/incomplete requests on a per-IP basis	Can mitigate all types of Slow HTTP DoS attacks	Ineffective if an attack is launched using multiple IP addresses
	mod_reqtimeout [106]	Mitigation by ensuring that a complete request is received within a predefined amount of time	Can be easily configured in the Apache web server	Cannot mitigate HTTP PRAGMA attack
Comparing Traffic Profiles	Tripathi et al. [157]	Detection by comparing HTTP traffic profiles generated during training and testing phase using HD	Can detect both Slow Message Body and Slow Header attacks with very high accuracy	Cannot mitigate the attacks
	Tripathi and Hubballi [155]	Detection by comparing HTTP 2.0 traffic profiles generated during training and testing phases using chi-square statistical test	Can detect all types of DoS attacks known against HTTP 2.0	Cannot mitigate the attacks
Monitoring Traffic Features	Dantas et al. [31]	Mitigation by dropping an established connection if the number of bytes sent and received from it exceeds a predefined threshold	Can mitigate the attacks	May drop legitimate connections if the web page being accessed contains large objects

Transfer Protocol (FTP), SIP, and DNS. These attacks are also known as *meta* attacks [16]. In this section, we describe these attacks and the defense mechanisms known to counter them.

4.1 Slow Rate Generic DoS Attacks

There are four protocol-independent Slow Rate Generic DoS attacks shown in Figure 1. These attacks exploit features that are known to exist in TCP-based application layer protocols such as HTTP, SMTP, and FTP [17, 18]. Moreover, these attacks can also adopt their behavior to target a wide variety of TCP-based protocols [3]. In this section, we describe the working of these attacks and subsequently discuss the defense mechanisms known to counter them.

SlowReq/SlowDroid attack. Slowloris attack, as discussed earlier in Section 3.3.1, involves sending multiple incomplete HTTP requests to the web server to exhaust the connection queue space. The malicious client never sends the remaining portion of the HTTP request, and thus the server deletes the incomplete request from the connection queue after the timeout counter (≈300 seconds for Apache [157]) expires. Thus, it becomes relatively difficult for the malicious client to fill the connection queue, as it has to keep sending incomplete requests. However, Aiello et al. [3] proposed that if small chunks of a request (e.g., one byte each) are sent continuously at regular intervals, the target server keeps on resetting the connection timeout counter. This forces the target server to keep the incoming request in the connection queue for a very long duration of time (≈990 seconds for Apache [157]). Once the connection queue space is exhausted, the server is not able to serve legitimate requests. Based on this observation, the authors proposed the SlowReq attack [3]. Moreover, the authors discussed that SlowReq can adopt this behavior of sending request chunks slowly at regular intervals for other TCP-based protocols such as SMTP and FTP as well. This makes SlowReq effective against a large number of TCP-based application layer protocols, unlike Slowloris that is effective only against HTTP. In another work, Cambiaso et al. [14] proposed the SlowDroid attack, a variant of SlowReq attack that requires minimal computational resources and thus can be launched even from mobile phones.

Slowcomm attack. Cambiaso et al. [15] proposed an attack called *Slowcomm* that exploits the knowledge of the Retransmission Time Out timer that is used for congestion control in TCP. The proposed attack targets TCP-based application layer protocols. It can detect connection closes and immediately establish them again. Similar to SlowReq, Slowcomm also requires sending

Table 9. Defense Mechanisms vs Slow Rate DoS Attacks

Defense Mechanisms	SlowReq [3]	Slowcomm [15]	Slow Next [17]	SlowDrop [18]
Monitoring Traffic Features	✓	✓	✓	✓

Table 10. Approaches to Counter Slow Rate DoS Attacks

Category	Research Work	Defense Mechanism	Strength	Weakness
Monitoring Traffic Features	Mongelli et al. [108] and Aiello et al. [2]	Detection by tracking the number of packets the server receives in a time interval	Can be easily deployed to find the traffic anomalies	High false-positive rate in case of scenarios such as Flash Event [10]
	Shtern et al. [135]	Mitigation by checking the rate with which a user is sending the requests and further scrutinizing the requests based on how they impact server resources	Can detect and mitigate all types of Slow Rate DoS attacks	Difficult to develop a performance model based on different server resources

protocol-independent request chunks at regular intervals, but detecting connection closes is what makes Slowcomm more effective than the SlowReq attack.

Slow Next attack. To launch the Slow Next DoS attack [17], an adversary uses persistent connection features of protocols such as HTTP, SMTP, FTP, and SSH [17, 18], which keep the connections alive for a longer duration. The adversary establishes several persistent connections with the server and sends benign requests from each connection periodically to consume the connection queue space. This attack is stealthier than Slow Rate DoS attacks, as the server successfully parses the complete requests and sends back the response.

SlowDrop attack. Similar to the Slow Next attack, the SlowDrop attack [18] also uses a persistent connection feature of different protocols to prolong the connection waiting time at the server. In this attack, an adversary first requests a large resource to the server and then drops the receiving packets by behaving as an unreliable client. Due to the persistent connection and the intentional dropping of packets, the waiting time for the connection is prolonged at the server. An adversary establishes a large number of such connections with the server to prevent legitimate clients from establishing a connection with the server, thereby causing the DoS scenario.

Tools. The tool to launch the Slowdroid attack is available at https://downloadapk.net/SlowDroid-DoS-Tool.html [140], but for other attacks there are no tools available. Thus, these attacks can be launched by crafting special requests using Scapy [128] and sending them using raw sockets.

4.1.1 Defense Mechanisms. We classify the known defense mechanisms to detect Slow Rate Generic DoS attacks into one category as discussed next.

window-based traffic statistics

Monitoring Traffic Features. Mongelli et al. [108] and Aiello et al. [2] proposed defense mechanisms that detect attacks by counting the number of packets received by a server in an interval. If the number of packets received in the interval exceeds a pre-defined threshold, the interval is considered as containing attack traffic. Another detection scheme was proposed by Shtern et al. [135] that monitors the rate of requests sent by a client to the server. If the sending rate exceeds a pre-defined threshold, the requests are further scrutinized by analyzing their impact on the server's resources (CPU, memory, etc.). If all requests from this suspicious user fall within thresholds, the user is then considered as a legitimate one and its requests are transferred to the regular application.

Table 9 shows the detection/mitigation ability of the defense mechanisms to counter Slow Rate Generic DoS attacks, whereas Table 10 summarizes the category of defense mechanisms along with their strengths and limitations.

4.2 Request Flood Attacks

Another category of generic application layer DoS attacks is Request Flood attacks as shown in Figure 1. In these attacks, an adversary sends a flood of requests to a victim server to overwhelm it and prevent it from serving legitimate requests. Launching these attacks require more computers as compared to Slow Rate DoS attacks. As a result, they are typically considered as belonging to the DDoS category. Since the adversary has the option of sending a flood of packets belonging to different protocols, there are different types of request flood attacks, of which the most popular ones are HTTP Flood, SIP Flood, and DNS Flood.

HTTP Flood. HTTP Flood attack is the most popular type of request flooding attack. An adversary can launch this attack by sending a flood of HTTP requests to a web server to exhaust its resources, which leads to a DoS scenario for legitimate clients. The popular attacks belonging to this category are HTTP GET and HTTP POST flooding attacks [169] in which the adversary sends several benign HTTP requests to a victim server. An adversary also has the option of sending a specially crafted small request from each established connection that causes the web server to perform some CPU-intensive tasks that result in a high workload at the web server. This attack is commonly known as Repeated One Shot or Asymmetric Workload attack [120]. Adi et al. [1] described different flooding attacks against HTTP 2.0 that involve sending a flood of web requests to an HTTP 2.0–enabled server to create a DoS scenario.

SIP Flood. SIP Flood attack [69] is characterized by sending a flood of SIP messages to an SIP entity to make it difficult to process these messages and eventually to crash the system. This attack mainly targets the resources of the SIP system like memory, CPU, and bandwidth to crash it. SIP Flood attacks can be launched by sending several REGISTER, BYE, or INVITE messages to different components of the SIP system, such as SIP Registrar and SIP proxy. A malicious client can also launch a malformed message–based flood attack against an SIP entity wherein it sends a flood of malformed SIP messages to the victim. On receiving these messages, the entity processes them but eventually drops them. This leads to the wastage of CPU cycles of the SIP entity.

DNS Flood. The DNS protocol is one of the core protocols of Internet infrastructure. Compromising the working of DNS protocol may lead to disruption of other important services such as web and mail. To launch a DNS Flood attack [161], an adversary sends a flood of DNS queries to a DNS server to prevent it from processing and serving DNS queries sent by benign clients.

Tools. Depending on the protocol being targeted, different tools can be used to launch the Request Flood attacks. The most popular Request Flood attack is the HTTP Flood attack, and thus there is a large number of popular automated tools, such as HTTP Unbearable Load King (HULK) [62] and Low Orbit Ion Cannon (LOIC) [88], that can be used to launch HTTP Flood attack. However, tools like StarTrinity SIP Tester [143] and inviteflood [75] can be used to launch SIP Flood attack. To launch DNS Flood attack, tools such as DNS-flood-ng [40] can be used.

Defense mechanisms. Several works are available in the literature that discuss defense approaches to detect and mitigate Request Flood attacks. We classify these defense mechanisms into different categories as shown in Figure 3. Since the defense mechanisms are specific to the protocol, we discuss them by taking one protocol into account at a time.

4.2.1 Defense Mechanisms to Counter HTTP Flood Attack. The defense mechanisms known to counter the HTTP Flood attack can broadly be classified into four different categories as follows.

Analyzing Web Browsing Behaviour. Most of the defense mechanisms known to counter HTTP Flood attack fall under this category. These defense mechanisms involve modeling normal web

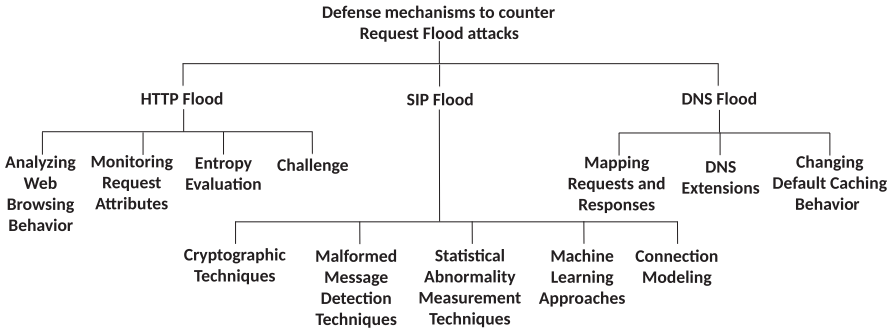

Fig. 3. Defense mechanisms to counter Request Flood attacks.

anomaly detection based on profiles

browsing behaviour using different features such as request rate, number of unique source IP addresses encountered in a time interval, packet size, and sequence of web page access, among others. Ranjan et al. [120] proposed an approach that analyzes characteristics like Session Arrival Time, Session Inter-Arrival Time, and Request Inter-arrival Time [120] to create a web profile of different users. If a user's profile significantly deviates from this pre-defined profile, the user is considered as a malicious one. Yatagai et al. [168] proposed an approach that correlates average user reading time and page information size to detect anomalies. In another work, Lee et al. [83] represented normal web surfing behavior using attributes such as Number of User Requests, Average Number of Requests, and Access Frequency of Most Popular Page [83]. Subsequently, the authors used a multiple PCA model to distinguish legitimate and flood attack traffic. Oikonomou and Mirkovic [112] used three aspects of web browsing behavior—Request Dynamics, Request Semantics, and Ability to Process Visual Clues [112]—to distinguish legitimate and malicious requests. Xu et al. [167] introduced an extended random walk model to establish a benign pattern of browsing sequences to detect asymmetric DoS attacks. For each client, a browsing pattern is generated, and if it deviates from the pre-established normal pattern, the attack is detected. Xie and Yu [165] proposed an extended **Hidden semi-Markov Model (HsMM)** to create web browsing profiles of different users. In particular, the authors approximated the time required by a client to read a web page in terms of the number of in-line requests made by the page. Subsequently, an Original Likelihood Distribution is constructed that represents the client's browsing behavior. If this browsing behavior deviates from the pre-constructed model, the client is detected as anomalous. In other similar work, Xie and Yu [166] constructed HsMM based on the web page popularity to describe web browsing behavior. Wang et al. [163] and Singh et al. [137] proposed approaches that rely on the fact that the surfing preference of a benign user is much more consistent with the web page popularity than that of an adversary.

Monitoring Request Attributes. Chwalinski et al. [22, 23] proposed approaches that differentiate malicious and legitimate HTTP connections based on the request count per resource sent from each connection. Based on this distinguishing feature, the detection approach groups users into different clusters using K-means clustering. An incoming connection that cannot be assigned to any cluster is considered as anomalous one. To handle cases when values of some of the features are less than the threshold values while some features' values exceed this threshold, the authors used a HsMM for accurately classifying the connection. Jazi et al. [76] proposed a detection approach that identifies changes in the number of HTTP request packets received at a server and the number of corresponding network packets carrying these requests to detect HTTP Flood attack.

Entropy Evaluation. A class of works in the literature uses entropy as a measure to differentiate legitimate and malicious client. Legitimate HTTP requests vary in size, rate, and intent, whereas flood attack requests are uniform and follow some periodicity. As a result, requests belonging to a flood attack possess lower entropy as compared to legitimate requests [119]. Singh et al. [77, 138] used **Multilayer Perceptron with Genetic Machine Learning Algorithm (MLP-GA)** to distinguish malicious and legitimate connections. In their work [77], the features used are the Entropy Variance per IP Address, Entropy of HTTP GET Request Count per Connection, and the Number of HTTP GET Request Counts, whereas in the work of Singh and De [138], the features used are Number of IP Addresses, Number of HTTP Requests, and Fixed Frame Length.

Challenge based. Some of the works identify automated requests sent during a flood attack using challenges such as **Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA)** and **Are You a Human (AYAH)** [119]. Ndibwile et al. [111] proposed a mitigation scheme wherein three web servers are used, namely a real web server, a bait, and a decoy server. The incoming requests are first forwarded to the bait server where the incoming requests are marked as suspicious or legitimate based on a JavaScript pop-up dialogue box. Subsequently, the legitimate requests are redirected to the real server, but the suspicious requests are redirected to the decoy server. At this server, the same authentication method is again used to authenticate the suspicious traffic, which, if found legitimate, is routed to the real web server. Zhang et al. [170] proposed a defense approach that identifies malicious traffic by using client-side physical uplink bandwidth as a challenge to identify malicious traffic. Suriadi et al. [146] proposed an approach that sends to a client a hash-based computation-bound puzzle and the corresponding solution inside a SOAP header. In response, the client needs to determine a partial pre-image in a hash function to differentiate itself from a bot.

Table 11 gives an overview of different categories of defense mechanisms and research works falling under each category along with their strengths and limitations.

4.2.2 Defense Mechanisms to Counter SIP Flood Attack. As shown in Figure 3, The defense mechanisms known to counter SIP flood attack [69] can broadly be classified into five different categories as follows.

Cryptographic Techniques. To mitigate the SIP flooding attack, Arkko et al. [6] proposed an approach for negotiating security mechanisms such as **Transport Layer Security (TLS)** protocol [36] that are used between SIP entities in the SIP infrastructure. In another work, Farley and Wang [48] proposed pre-distributed shared key-based shields for both the end users and their proxy server. The pre-shared key is then used by a hash function to generate a MAC that is appended to all of the exchanged SIP packets. In this way, this approach helps authenticate the origin of an SIP message and verify its integrity.

Malformed Message Detection Techniques. Defense approaches falling under this category require pre-processing the collected traffic traces to pre-define encoded patterns for different attack variants. These encoded patterns are then stored in a signature database. During the classification phase, these approaches then scan an incoming packet for the pre-defined patterns, and depending on whether a match is found or not, the packet is classified as anomalous (malformed) or normal, respectively. The typical working of defense mechanisms belonging to this category is shown in Figure 4. In this direction, Geneiatakis et al. [53] presented a scheme that considers SIP message syntax [125] as attack signature. The syntax of different types of malformed messages is stored in a database, and if syntax of a received message is similar to one of the signatures in the database, it is characterized as malformed. Ehlert et al. [46] proposed a two-level security mechanism wherein the first-level check mitigates different network- and transport-layer related attacks

Table 11. Approaches to Counter HTTP Flood Attack

Category	Research Work	Defense Mechanism	Strength	Weakness
Web Browsing Behavior	Ranjan et al. [120]	Detection by comparing profiles generated during training and testing phases	Can classify different attack types and detect and mitigate them	Can be bypassed by controlling packet-sending rates using a large-scale botnet [133]
	Lee et al. [83]	Detection by comparing profiles based on parameters extracted from web page request sequences without considering sequence order of requested pages	Provides early detection functionality and better detection rate than HsMM	Does not mitigate the attack
	Oikonomou and Mirkovic [112]	Detection using comparing request dynamics, request semantics, and visual cues processing	Low false positives	Complex deployment procedure
	Xu et al. [167]	Detection using a random walk model (prediction of page request sequence)	Low computational complexity	Analyzing the page request sequence may cause false alarms [83]
	Yatagai et al. [168]	Detection by analysis of page access behavior based on browsing sequence and correlation of browsing time and page size	Can detect dynamic flood attack as well	False alarms due to analyzing page request sequence, as it might vary for different individuals
	Xie and Yu [165, 166]	Detection by classifying the benign and malicious browsing behavior based on HsMM	Provides early detection functionality	False alarms due to analyzing page request sequence, as it might vary for different individuals
	Wang et al. [163]	Detection by comparing surfing preferences of a benign user and an adversary	Better detection performance as compared to other transition probability–based mechanisms [119]	Web page popularity is known to vary over time, which may cause false alarms
	Singh et al. [137]	Detection by finding anomalies based on request, response, popularity, and repetition index using machine learning algorithms	Can detect different strategies employed to launch the attack	Web page popularity is known to vary over time, which may cause false alarms
Monitoring Request Attributes	Chwalinski et al. [22, 23]	Detection using clustering based on the number of requests per resource per connection	Can detect stealthy bots	Inadequate in detecting short attack sequences
	Jazi et al. [76]	Detecting change in proportion of HTTP requests and the network packets carrying them	Low computational complexity	Asymmetric attack may go undetected
Entropy Evaluation based	Singh et al. [77]	Detection using MLP-GA based on GET request count, entropy, and variance	Can be adapted easily to detect different types of flood attacks	Cannot mitigate the attack
	Singh et al. [138]	Detection using MLP-GA based on GET request count and number of IP addresses	Can be adapted easily to detect different types of flood attacks	Cannot mitigate the attack
Challenge based	Ndibwile et al. [111]	Detection using JavaScript pop-up dialogue box	Lower false positives	Complex to deploy
	Zhang et al. [170]	Mitigation using client-side physical uplink bandwidth as a challenge to identify malicious traffic	Can detect and mitigate the attack	Complex deployment in case of a high workload environment
	Sivabalan and Radcliffe [139]	Mitigation by identifying and blocking suspicious requests using the AYAH challenge	Can detect and mitigate the attack	Turnaround time of the user requests increases
	Suriadi et al. [146]	Mitigation using a hash-based computation-bound puzzle to identify suspicious requests	Using a non-interactive challenge does not reduce the user's experience	Turnaround time of the user requests increases

Paper lists of all papers

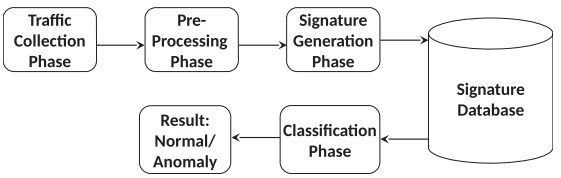

Fig. 4. Working of signature-based anomaly detection.

and SIP Flood attacks. In the next level, similar to the approach proposed by Geneiatakis et al. [53], the scheme checks for the message syntax of a received SIP packet to classify it as a normal or malformed one. In another work, Lahmadi and Festor [82] proposed an approach that first analyzes the SIP exploit traffic collected beforehand to extract contextual information of an attack and subsequently use this information to block illegitimate packets after its deployment. Wu et al. [164] proposed an approach, SCIDIVE, that first processes incoming packets and then decodes them into information units based on the protocol within them. The units belonging to the same

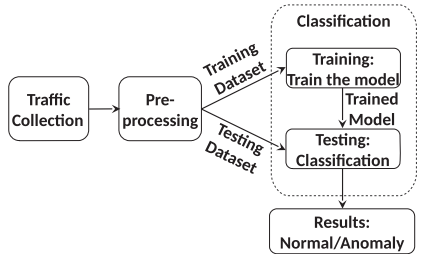

Fig. 5. Working of machine learning techniques.

session are grouped together and mapped into events. Subsequently, a rule engine matches these events against pre-defined rule sets [164]. If the system's aggregated event does not conform to any of the rules, SCIDIVE raises an alarm. In another work, Seo et al. [132] proposed an approach called *SIP-VoIP Anomaly Detection* (SIPAD) that makes use of a stateful SIP rule tree to identify the presence of a flooding attack. Based on the present state and the type of message, the proposed approach deduces the corresponding branches from the tree. Subsequently, it checks the structure of an SIP message by comparing it to the branches to classify the message as normal or malformed.

Statistical Abnormality Measurement Techniques. Statistical abnormality measurement techniques detect abnormalities in the behavior of incoming traffic by comparing it with the pre-defined normal behavior. For this comparison, different methods [49] can be put to use. In this direction, some authors [131, 148, 149] used HD [19] to measure the correlation between a normal and an anomalous traffic profile (probability distribution). This profile is comprised of different vectors that denote the occurrence probability of the different types of SIP messages in a particular time period. A similar approach was proposed by Golait and Hubballi [57] that ensures that the SIP messages counts in different intervals of the testing phase do not cross a pre-defined threshold. Reynolds and Ghosal [123] exploited the fact that in flooding attack, the number of attempted connections is higher than the number of completed connections and thus proposed an approach that calculates the difference between the number of completed and attempted connections. If this difference is greater than a pre-set threshold, attack presence is declared. Tang and Cheng [147] proposed an approach that observes the change in a wavelet's energy level to detect flood attacks. This is motivated by the fact that in event of an attack, the signal energy rises sharply even if the traffic rate rises slowly, which leads to detection of the attack. Semerci et al. [130] proposed a change detection algorithm that computes the change in the SIP messaging traffic patterns at the server side using Mahalanobis distance [89]. If any significant change in the characteristics of messaging flows is found, the algorithm raises an alarm. In another work, Geneiatakis et al. [55] proposed an approach that monitors the number of INVITE Requests, Responses, and ACKs [55] received in a particular time interval. If these counts are greater than a pre-defined threshold, the detection system raises the alarm.

Machine Learning Approaches. A class of defense approaches uses machine learning techniques as anomaly detectors to detect SIP flood attack. An abstract overview of the working of defense mechanisms belonging to this category is shown in Figure 5. In this direction, Nassar and Festor [110] proposed an approach that first trains a **Support Vector Machine (SVM)** using 38 statistical features of various message types and subsequently used the trained model to classify normal and attack traffic. Similarly, Akbar and Farooq [4] attempted to detect flood attack using Naive Bayes and Decision Tree algorithms. The authors used different features extracted from a time window to train the machine learning algorithms.

All these methods kinda look similar. Patterns, sigatures, anomaly detection. I can't see the underlying differentiation between them (e.g. Statistical Abnormality vs MachineLearning vs Signatures etc). Ok, ML = learning, but the others..

Table 12. Approaches to Counter SIP Flood Attack

Category	Research Work	Defense Mechanism	Strength	Weakness
Cryptographic Techniques	Arkko et al. [6]	Mitigation by negotiating security mechanisms such as TLS between UA and proxy	Can detect and mitigate the attack	High implementation complexity
	Farley and Wang [48]	Mitigation by verification of all exchanged SIP messages using MAC	Backward compatible	Requires pre-distribution of shared key
Malformed Message Detection Techniques	Geneiatakis et al. [53]	Detection by comparing SIP message syntax to characterize malformed messages	Can be adapted for other VoIP signaling protocols	Can only detect pre-encoded attacks
	Ehlert et al. [46]	Mitigation using double-level security architecture	Can detect and mitigate the attack	Can only detect pre-encoded attacks
	Lahmadi and Festor [82]	Mitigation using context-aware prevention specifications	Can automatically infer prevention specifications by analyzing captured SIP exploit traffic	Can only detect pre-encoded attacks
	Wu et al. [164]	Mitigation by stateful and cross-protocol intrusion detection architecture	Can detect different DoS and media stream–based attacks	Difficult to define an exhaustive rule-set
	Seo et al. [132]	Detection of flooding attack using stateful SIP rule tree	Can detect both flooding attack and malformed SIP messages	Computational overhead due to rule tree generation
Statistical Abnormality Measurement Techniques	Sengar et al. [131], Tang et al. [148], Tang et al. [149]	Anomaly detection by computing the HD between traffic distributions	Lightweight approach	Can be bypassed by balancing the different types of messages during attack [58]
	Reynolds and Ghosal [123]	Detection by computing the difference between the number of attempted and completed connections	Lightweight approach	Results in higher false positives in case of flash crowd
	Tang and Cheng [147]	Detection by observing change in energy of the detail signal	Highly scalable	Results in higher false positives in case of flash crowd
	Semerci et al. [130]	Detection by computing the difference between SIP messaging traffic patterns at the server side using Mahalanobis distance	Can detect attack and identifies the attacker as well	Can be bypassed by balancing the different types of messages during attack
	Geneiatakis et al. [55]	Detection by checking if the number of different messages is less than a pre-defined threshold	Low implementation complexity	Cannot detect a DDoS flood launched using large number of compromised bots
Machine Learning Approaches	Akbar and Farooq [4]	Detection based on packet-based analysis using a set of spatial and temporal features	No requirement of transforming network packet streams into traffic flows	Cannot detect attack if launched using carefully crafted malicious SIP messages [96]
	Nassar and Festor [110]	Detection using trained SVM	Low implementation complexity	Cannot detect attack if launched using carefully crafted malicious SIP messages [96]
Connection Modeling based	Ding and Liu [38]	Detection by identifying vulnerable SIP INVITE transaction states by analyzing CPN state space	Can detect all attacks pertaining to INVITE transaction	Cannot detect flood attack launched using other message types
	Liu [86] and Liu [87]	Detection by modeling SIP INVITE transactions using CPNs over reliable and unreliable transport mediums	Can detect all attacks pertaining to INVITE transaction	Cannot detect flood attack launched using other message types
	Ding and Su [39]	Detection using timed HCPN	Can detect all attacks pertaining to INVITE transaction	Cannot detect flood attack launched using other message types
	Golait and Hubballi [58]	Detection by modeling SIP communication using a PCDTA	Can detect flood attack launched at varying rates	Computational overhead due to PCDTA generation
	Chen [21]	Detection by modeling SIP communication using FSM	Can detect attacks irrespective of the existing SIP software	Ineffective against attacks launched by carefully crafting malicious SIP messages

INTERESTING

SIP Connection Modeling based. Golait and Hubballi [58] proposed an approach wherein an SIP connection is modeled as a **Probabilistic Counting Deterministic Timed Automata (PCDTA)** [58]. Only if the connection conforms to PCDTA such that the probability of traversing the path from starting state to final state is greater than a predetermined threshold is the connectiondeclared as legitimate. Chen [21] modeled the SIP communication using **Finite State Machine (FSM)** and used different thresholds based on the rate of transactions and the number of allowed packets per transaction. If any of the threshold value is crossed, the detection system raises the alarm. In other works [38, 39, 86, 87], **Colored Petri Nets (CPNs)** were used to model SIP INVITE transactions and subsequently identify which states in the constructed model are vulnerable to flooding attack.

Table 12 gives an overview of different categories of defense mechanisms and research works falling under each category along with their strengths and limitations.

Table 13. Approaches to Counter DNS Flood Attack

Category	Research Work	Defense Mechanism	Strength	Weakness
Mapping DNS Requests and Responses	Kambourakis et al. [79]	Mitigation by putting a threshold on DNS requests for which there is no corresponding response	Can be easily deployed	Not scalable for busy nameservers
	Sun et al. [145]	Mitigation by correlating DNS requests and responses using Bloom filters	Can be easily deployed	Ineffective if the adversary and victim nameserver are using the same ISP
Extension to DNS	Guo et al. [60]	Detection of spoofed DNS requests by associating each request originator with a unique cookie	Does not require modifying DNS protocol	Ineffective if attack is launched using multiple computers without spoofing the source IP address
	Zhu et al. [171]	Mitigation by using DNS over TLS	Low memory requirement for implementation	Provides attack surface for connection-oriented DoS attacks [119]
Changing Default Caching Behavior of DNS Records	Ballani and Francis [8]	Mitigation by proposing a use of secondary stale cache	Can detect and mitigate the attack	High implementation complexity
	Pappas et al. [114]	Mitigation by setting longer TTL values for DNS infrastructure records	Operationally feasible and immediately deployable even by large zones	Makes it harder for operators to update their records [8]

4.2.3 Defense Mechanisms to Counter DNS Flood Attack. As shown in Figure 3, the defense mechanisms known to counter DNS flood attack can broadly be classified into three different categories as follows.

Mapping DNS Requests and Responses. The easiest way to detect DNS Flood attack is to match the DNS requests received at a server to the responses sent by it. In case the number of DNS requests for which there is no corresponding response crosses a pre-defined threshold, it can be concluded that the server is under a flood attack and subsequently the firewall rules can be updated to block the malicious traffic. Such an approach is presented in the work of Kamourakis et al. [79]. Similarly, Sun et al. [145] proposed an approach that mitigates DNS Flood attack by correlating DNS requests and responses using a Bloom filter.

Extension to DNS. Guo et al. [60] proposed a firewall module called *DNS Guard* that detects identity spoofing–based DNS Flood attacks. In particular, this strategy requires a DNS server to issue a distinct cookie to each client, and the client in turn associates each request it sends to the server with the previously issued cookie. Using this cookie, it can be determined if a DNS request indeed is sent from the address indicated in the packet. Zhu et al. [171] proposed an extension called *T-DNS* to use TLS protocol [36] for DNS communication to ensure security and privacy.

Changing Default Caching Behavior of DNS Records. Ballani and Francis [8] proposed that DNS resolvers should not erase cached records even after their **Time to Live (TTL)** has been expired. Instead, these stale records should be stored in a secondary cache. In an attack scenario, if a resolver does not receive a response from the authoritative nameservers of the domain under consideration, the resolver should answer the query using the information stored in the secondary cache. In another work, Pappas et al. [114] showed that the DNS protocol can be made significantly resilient against flood attack just by setting longer TTL values for DNS infrastructure records that are used to navigate the DNS hierarchy. As a result, the infrastructure records are stored in the resolver's cache for a longer duration, which can then be used to answer the query when the nameserver is under attack.

Table 13 gives an overview of different categories of defense mechanisms and research works falling under each category along with their strengths and limitations.

5 COMPARING DIFFERENT APPLICATION LAYER DOS ATTACKS AND DEFENSE MECHANISMS

In this section, we first present a comparative study of different application layer DoS attacks. Subsequently, we also compare the known defense mechanisms to counter the attacks based on different parameters.

5.1 Comparison of Application Layer DoS Attacks

We consider the following parameters to compare different application layer DoS attacks.

Execution complexity. This parameter corresponds to the complexity of executing the attack in a real network setup. The higher the execution complexity, the more difficult it is to launch the attack and vice versa. The execution complexity of launching an attack depends on factors such as the requirement of escalated privileges (e.g., MitM position) and resource consumption (e.g., bandwidth) of the malicious client. The attack execution complexity also increases if the malicious client has to fulfill complex requirements such as the exploitation of IP fragmentation policies of the underlying operating system.

Target. This parameter refers to the entity that is targeted during an application layer DoS attack. The targeted entity can be either an individual client or a server. For example, a malicious client targets NTP clients individually while launching DoS attacks against NTP. However, in other application layer DoS attacks that we discussed, the malicious client targets a server to prevent it from serving the incoming requests.

Traffic overhead. This parameter refers to the quantum of traffic that a malicious client sends to a target (either to a victim client or to a server). If the malicious client targets the victim clients individually (e.g., in case of DoS attacks against NTP), this parameter refers to the traffic sent to prevent one victim client from accessing the service. However, if the malicious client targets a server as a whole, this parameter refers to the traffic sent to prevent the server from serving any incoming request. It is to be noted that this parameter is applicable for comparing the attacks falling under the same category only. This is because the different categories of DoS attacks target different entities (either individual clients or servers), and thus they should not be compared solely based on this parameter.

Stealthiness. If a malicious client needs to send a large number of specially crafted abnormal requests to launch an application layer DoS attack, the chances of the attack detection by a DoS detection/mitigation scheme may increase. Thus, from the malicious client's perspective, launching an attack should involve sending a minimum number of such requests to keep the traffic generation bar as low as possible. If the traffic generated due to launching an attack is high, we consider the attack as less stealthy.

Relevance. Some of the DoS attacks are relevant within a **Local Area Network (LAN)** only, as the scope of the application layer protocols they target is restricted within LAN. However, some of the DoS attacks target popular application layer protocols being used on the Internet and thus are relevant to the Internet. We used this parameter to differentiate the attacks relevant to either LAN or the Internet.

Spoofing? To launch some of the application layer DoS attacks, a malicious client needs the ability to spoof the identity of the involved entities (client/server). Identity spoofing can be in the form of either MAC address spoofing or IP address spoofing.

Real incidents? We use this parameter to differentiate the application layer DoS attacks based on whether there have been real incidents on the Internet where commercial security solution providers observed these attacks.

Attack type. Depending on the number of devices required to create a DoS scenario against a target, we classify the discussed attacks as belonging to either DoS or DDoS. If a malicious client requires several devices to launch an application layer DoS attack, the attack is considered a DDoS attack. However, if the malicious client requires only one device to launch an attack, the attack is considered a DoS attack.

Table 14. Comparison of Application Layer DoS Attacks

Attack Category	Specific Attack	Complexity	Target	Overhead	Stealthiness	Relevance	Spoofing?	Real Incidents?	Attack Type	Booters Applicable?
Attacks Against DHCP	Induced Starvation	Low	Server	Low	High	LAN	Yes	Yes	DoS	No
	Classical Starvation	High	Server	High	Relatively Low	LAN	Yes	Yes	DoS	No
Attacks Against NTP	Deja Vu	High	Individual Clients	High	Low	Internet	Yes	No	DoS	No
	On-Path Timeshifting	High	Individual Clients	Minimal	Very High	Internet	Yes	No	DoS	No
	DoS by Spoofed Kiss-o'-Death	Low	Individual Clients	Minimal	Very High	Internet	Yes	No	DoS	No
	DoS by Bad Authentication	Low	Individual Clients	Minimal	Very High	Internet	Yes	No	DoS	No
	Interleaved-Pivot Attack	Low	Individual Clients	Minimal	Very High	Internet	Yes	No	DoS	No
	Pinning to Bad Timekeepers	Low	Individual Clients	Low	High	Internet	Yes	No	DoS	No
	Fragmentation attack	High	Individual Clients	High	Low	Internet	Yes	No	DoS	No
Slow HTTP DoS	Slow Header	Low	Server	Low	High	Internet	No	Yes	DoS	No
	Slow Message Body	Low	Server	Low	High	Internet	No	Yes	DoS	No
	Slow Read	Low	Server	Low	High	Internet	No	Yes	DoS	No
	HTTP PRAGMA	Low	Server	Low	High	Internet	No	Yes	DoS	No
Slow Rate Generic DoS	SlowReq	Low	Server	Low	High	Internet	No	No	DoS	No
	Slowcomm	Low	Server	Low	High	Internet	No	No	DoS	No
	Slow Next	Low	Server	Low	Very High	Internet	No	No	DoS	No
	SlowDrop	Low	Server	Low	Very High	Internet	No	No	DoS	No
Request Flood	HTTP Flood	High	Server	Very High	High	Internet	No	Yes	DDoS	Yes
	SIP Flood	High	Server	Very High	High	Internet	No	Yes	DDoS	Yes
	DNS Flood	High	Server	Very High	High	Internet	No	Yes	DDoS	Yes

DDoS-for-hire applicable? With the unfolding of DDoS-for-hire services, also known as booters [78], the bar for launching DDoS attacks has significantly lowered. For a very nominal cost, these services sell to an adversary access to a large number of malware-infected devices. Subsequently, the adversary uses these devices to launch DDoS attacks against a victim. Although these services provide the adversary resources to launch DDoS attacks, they are not required to launch simple DoS attacks wherein the adversary just needs only one device.

Based on these parameters, the comparison of different application layer DoS attacks is shown in Table 14. The following observations can be made from this table:

1. The complexity of the classical DHCP starvation attack is high, as it requires sending multiple bogus requests using random MAC addresses. Moreover, the complexity of DoS attacks against NTP such as Deja Vu and On-Path Timeshifting is high, as they require the malicious client to obtain a privileged position between the victim client and the NTP server. The complexity of fragmentation-based off-path timeshifting attack against NTP is high because it requires the exploitation of IP fragmentation policies, which is a cumbersome process. The complexity of request flood attacks is high, as they require sending a large number of requests to the target server.

2. Launching an induced DHCP starvation attack involves sending only spoofed probe replies and thus requires less traffic overhead as compared to a classical DHCP starvation attack. This also makes the induced DHCP starvation stealthier than the classical DHCP starvation attack. In addition, since launching a Deja Vu attack requires sending a sequence of packets over and over, it results in relatively higher traffic overhead and thus is less stealthy as compared to other attacks against NTP. Moreover, the on-path timeshifting and the off-path DoS attacks (excluding the attack proposed by Tripathi and Hubballi [156]) against NTP require sending a very small number of spoofed NTP packets to prevent a victim client from synchronizing its clock. Thus, these attacks require the least amount of traffic and are highly stealthy. However, the fragmentation-based off-path timeshifting attack requires sending several IP fragments to exploit the IP fragmentation

This is just paraphrasing the table

policy which results in high traffic overhead. This makes the attack less stealthy. From the table, we can also notice that launching the Slow HTTP DoS and Slow Rate Generic DoS attacks generates less traffic overhead and thus are stealthier in nature. Moreover, Slow Next and SlowDrop are stealthier than other attacks belonging to this category, as they require sending benign requests instead of incomplete requests at a slow rate. Launching Request Flood attacks requires sending a large number of legitimate requests from several devices due to which they generate very high traffic overhead. However, it is difficult to differentiate these attacks from scenarios such as flash crowd [10], as these attacks involve sending legitimate requests. This makes these attacks stealthier in nature.

3. Out of all application layer DoS attacks discussed in this article, only DHCP starvation attacks are limited to local networks. All other attacks are relevant over the Internet and thus can target a large number of victims.

4. As discussed earlier in Section 1, 53% of the organizations confirmed insider attacks against them [150]. Thus, we presume that there may have been incidents of DHCP starvation attacks as well. The incidents of Slow HTTP DoS and HTTP Flood attacks on the Internet have also been reported in the past [20, 52, 71, 84, 100].

5. Since the Request Flood attacks require sending a large number of requests to the target server, an adversary can make use of services like booters [78] to send the request flood from a large number of devices. For this reason, these attacks fall under the DDoS category. Moreover, DDoS-for-hire services are also applicable to these application layer DDoS attacks.

5.2 Comparison of Defense Mechanisms

We compare defense mechanisms known to counter various application layer DoS attacks based on the following parameters [107]:

(1) *Proactive or reactive?* We use this parameter to differentiate the defense mechanisms based on whether they defend attacks by proactively preventing them from occurring or they only react to the ongoing attacks.
(2) *Holistic defense?* We use this parameter to differentiate the defense mechanisms based on whether they can mitigate the DoS attacks or they can only detect the attacks.
(3) *Implementation complexity:* The deployment of defense mechanisms should be easy and feasible to implement. We use this parameter to compare different defense mechanisms based on their implementation complexity.
(4) *Scalability:* We use this parameter to differentiate the defense mechanisms based on whether they can be effective to counter the DoS attacks even if the number of adversaries and the amount of attack traffic increase.

Based on these parameters, the comparison of the defense mechanisms is shown in Table 15. The following observations can be made from this table:

A. The defense mechanisms belonging to the Cryptographic Techniques category involve high implementation complexity. This is because of the involvement of third-party modules (e.g., authentication server), and the requirement of cumbersome key distribution mechanism, and so on. Moreover, the implementation of defense mechanisms belonging to the Server Redundancy category involves high complexity. This is because these defense mechanisms require modification at the client side to overcome compatibility issues. The implementation complexity of challenge-based defense mechanisms to counter HTTP Flood attacks is high because they require the implementation overhead of third-party modules like CAPTCHA generation and verification servers. Since these servers are also a potential target of a DoS attack, securing them again causes extra implementation overhead. The implementation complexity of defense mechanisms that involve

Compare categories, not individual

Table 15. Comparison of Defense Mechanisms to Counter Different Attacks

Countering		Category	Proactive or Reactive?	Holistic Defense?	Implementation Complexity	Scalability
Attacks Against DHCP		Cryptographic Techniques	Proactive	Yes	High	High
		Security Features in Switches	Proactive	Yes	Low	High
		Using DHCP Relay Agent	Proactive	Yes	Low	High
		DHCP Traffic Profiling	Reactive	No	Low	High
		Machine Learning based	Reactive	No	Low	High
Attacks Against NTP		Cryptographic Techniques	Proactive	Yes	High	High
		Path Redundancy	Proactive	Yes	Low	High
		Server Redundancy	Proactive	Yes	High	High
Slow HTTP DoS		Implementation Modules	Proactive	Yes	Low	High
		Comparing Traffic Profiles	Reactive	No	Low	High
		Monitoring Traffic Features	Reactive	Yes	Low	High
Slow Rate Generic DoS		Monitoring Traffic Features	Reactive	Yes	Low	High
Request Flood	HTTP Flood	Web Browsing Behavior	Reactive	No	Low	High
		Monitoring Request Attributes	Reactive	No	Low	High
		Entropy Evaluation based	Reactive	No	Low	High
		Challenge based	Proactive	Yes (excluding [111])	High	Low
	SIP Flood	Cryptographic Techniques	Proactive	Yes	High	High
		Malformed Message Detection Techniques	Proactive	Yes (excluding [53] and [132])	Low	High
		Statistical Abnormality Measurement Techniques	Reactive	No	Low	High
		Machine Learning Techniques	Reactive	No	Low	High
		SIP Connection Modeling based	Reactive	No	Low	High
	DNS Flood	Mapping DNS Requests and Responses	Proactive	Yes	Low	High
		Extension to DNS	Proactive	Yes (excluding [60])	Low	High
		Changing Default Caching Behavior of DNS Records	Proactive	Yes	High	High

changing default DNS caching behavior to counter DNS Flood attacks is high because they require modifications in the protocol operation itself.

B. The scalability of the defense mechanisms belonging to the Challenge-Based category is low. This is because, in case of a large-scale attack, the adversaries can target the challenge providing service and prevent legitimate clients from obtaining the challenge.

6 COMMERCIAL SECURITY PRODUCTS TO COMBAT APPLICATION LAYER DOS ATTACKS

Since application layer DoS attacks are one of the major security threats today and considering their scale observed in recent incidents (discussed earlier in Section 1), the market of commercial DoS mitigation products has seen a sharp growth in past few years [74]. Major players in this field have kept updating their security products with different features such as higher attack handling capacity, ability to mitigate a wide variety of DoS attacks, and adding multiple layers of security to compete with other vendors. In this section, we show a comparison of some of the popular commercial DoS mitigation products (as per [47, 93]) in terms of their abilities to counter different DoS attacks. For this comparative study, we do not consider DHCP starvation attacks since they are limited to local networks only. The features that should exist in the commercial security products to mitigate the considered attacks are discussed in the next few sections.

Detection of spoofed IP packets. As discussed in Section 3.2.5, an adversary is required to send spoofed IP packets to NTP hosts to launch NTP timeshifting/timesticking DoS attacks. Thus, to detect these attacks, a security product must be able to detect packets that disguise their origin IP address. Once detected, the spoofed IP packets must be immediately dropped by the security product before forwarding it to the end entity, like an NTP host. If the security product cannot detect spoofed IP packets, it will not be able to detect and mitigate NTP timeshifting/timesticking DoS attacks.

Analysis of incoming requests. To detect attacks such as Slowloris and SlowReq that require sending incomplete requests, a security product must be able to analyze incoming requests using

Table 16. Comparison of Commercial DoS Solutions

Product Name	Vendor	NTP Timeshifting/ Timesticking Attacks	Slow HTTP DoS	Request Flood	Slow Rate
Imperva Incapsula	Imperva	✓	✓	✓	✓(only SlowReq)
DOSarrest	DOSarrest	✗	✓	✓	✓(only SlowReq)
F5	F5 Networks	✗	✓	✓	✓(only SlowReq)
Kona Site Defender	Akamai	✗	✓[2]	✓	✓(only SlowReq)
AppWall	Radware	✗	✓	✓	✓(only SlowReq)
Availability Protection System (APS)	Arbor Networks	✓	✗	✓	✗
Nexusguard	Nexusguard	✓	✗	✓	✗
Athena	Verisign	✗	✗	✓	✗
Cloudflare-rate limit	Cloudflare	✗	✗	✓	✗
SiteProtect NG	Neustar	✗	✗	✓	✗
Sucuri DDoS Protection	Sucuri	✗	✓	✓	✓(only SlowReq)
Azure DDoS Protection	Microsoft	✗	✓	✓	✓(only SlowReq)
Project Shield[3]	Google	✗	✗	✓	✗
AWS Shield Advanced	Amazon	✗	✓	✓	✓(only SlowReq)
IBM Cloud DDoS	IBM	✗	✓	✓	✓(only SlowReq)

techniques such as Deep Packet Inspection [51, 64–66]. If the security product finds a request incomplete, it should drop the request instead of forwarding it to the target server. For such analysis, the commercial security products are usually equipped with reverse proxy modules that are deployed in front of the target servers [98].

Monitoring incoming traffic rate. The commercial security products should monitor the incoming traffic rate for their customers so that in an event of a massive DDoS attack, the traffic destined for an IP address range belonging to the customer is redirected to the security provider's data centers. At these data centers, the attack traffic is first cleaned, and subsequently the clean traffic is then forwarded to the customer's IP address range. This mechanism is usually initiated when the traffic rate goes beyond a dynamic threshold determined automatically using past traffic behavior.

Monitoring connection features and analyzing the client's behavior. It is essential for a commercial security product to monitor the connection features (e.g., persistent connection) of the underlying transport layer protocol. This is because the malicious client exploits the persistent connection feature of TCP to launch Slow Next and SlowDrop attacks as discussed in Section 4.1. Moreover, the commercial security product should also analyze how a client behaves on receiving a response from the server. This is because even if a malicious client sends a legitimate request to the server, it may behave as an unreliable client by simply dropping the response packets from the server and then sending the same request to the target server repeatedly. The malicious client uses this behavior while launching the SlowDrop attack as discussed in Section 4.1.

Based on these features, we compare the ability of various commercial security products to detect the considered attacks in Table 16. From the comparison, we can notice that almost every security product can mitigate Request Flood attacks, as most of them can monitor the incoming traffic rate. We can also notice that very few products can counter NTP timeshifting/timesticking attacks. This is because most of the security products cannot detect spoofed IP packets that are sent while launching NTP timeshifting/timesticking DoS attacks.

7 CONCLUSION AND FUTURE RESEARCH DIRECTIONS

With this survey, we presented a structured and comprehensive review of application layer DoS attacks and defense mechanisms. We discussed the execution method of different attacks and tools known to launch them, and compared them based on different parameters. We also reviewed

[2]Customers are provided with the option of defining rules using an API interface to accept or reject particular request types.

[3]This free service is available only for news, human rights, or elections monitoring organizations and individual journalists.

various state-of-the-art defense mechanisms and commercial DoS mitigation products known to counter different application layer DoS attacks and discussed their strengths and weaknesses. As is evident from the existing literature, several application layer DoS attacks have been proposed against protocols such as DHCP, NTP, HTTP 1.1, HTTP 2.0, DNS, and SIP especially in the past decade, and the tools and techniques used to launch these attacks are continuously evolving with time. We believe that the presented survey will motivate researchers to scrutinize different application layer protocols to find potential vulnerabilities in them and build robust and effective defense mechanisms to counter any resulting attack. In this section, we discuss some promising future research directions so that the research community can get an insight into the existing research gap in this field.

Slow Rate DoS and Request Flood attacks against connection-oriented protocols. It would be interesting to study the effect of Slow Rate DoS attacks and Request Flood attacks against connection-oriented (TCP-based) protocols. For example, these attacks should be tested against connection-oriented application layer protocols such as FTP [118], SMTP [81], and Internet Message Access Protocol [28]. Although it is believed that most of the connection-oriented protocols are vulnerable against Slow Rate DoS attacks [16], there is no such comprehensive work that covers this study.

Application layer DoS attacks in the context of Internet of Things. The inception of **Internet of Things (IoT)** as a result of the fourth industrial revolution (Industry 4.0) has provided hackers in various underground communities unlimited interconnected resources with several security flaws. These hackers have already been able to successfully exploit these resources so that they can be misused for nefarious purposes such as the staging of a DoS attack. In addition, there have been few incidents in the past where IoT devices were used to launch DoS attacks [12, 121]. Thus, we believe that it is necessary to conduct a study on the different IoT devices accessible on the Internet and testing if they can be exploited to launch different types of flooding-based application layer DoS attacks. Another interesting study would be to analyze by what factor the effect of each application layer DoS attack on a target victim can be magnified if the vulnerable IoT devices are put into use.

Vulnerability analysis of different implementations of application layer protocols. Different software implementations of application layer protocols sometimes behave differently while processing an abnormal event because of the configuration parameters that are specific to the implementation. For instance, Apache and IIS HTTP servers behave differently on receiving an incomplete GET request due to which Apache is vulnerable to Slow Header attack while IIS is not [157]. Thus, we believe that it would be interesting to test different implementations of other application layer protocols against known vulnerabilities. In addition, a study can be conducted to check how widely the vulnerable implementations are currently being used on the Internet to measure the overall attack surface.

Development of automated attack launching tools. There are a plethora of tools available to launch classical DHCP starvation, Slow HTTP DoS, and Request Flood attacks. However, there are no automated tools available for launching induced DHCP starvation, NTP DoS attacks, and Slow Rate Generic DoS attacks. Thus, automated tools can be developed to launch the attacks for which there are no available tools. Developing an automated tool helps in the long run, as it is useful while testing different implementations of an application layer protocol and measuring attack surface on the Internet. Another good alternate option to develop an automated tool for each attack type is to build utilities that can be integrated into popular penetration testing frameworks such as Armitage [29] and Metasploit [97].

REFERENCES

[1] Erwin Adi, Zubair A. Baig, Philip Hingston, and Chiou-Peng Lam. 2016. Distributed denial-of-service attacks against HTTP/2 services. *Cluster Computing* 19, 1 (2016), 79–86.

[2] Maurizio Aiello, Enrico Cambiaso, Maurizio Mongelli, and Gianluca Papaleo. 2014. An on-line intrusion detection approach to identify low-rate DoS attacks. In *Proceedings of the International Carnahan Conference on Security Technology (ICCST'14)*. 1–6.

[3] Maurizio Aiello, Gianluca Papaleo, and Enrico Cambiaso. 2014. SlowReq: A weapon for cyberwarfare operations. Characteristics, limits, performance, remediations. In *Proceedings of the International Joint Conferences SOCO'13-CISIS'13-ICEUTE'13*. 537–546.

[4] Muhammad Ali Akbar and Muddassar Farooq. 2014. Securing SIP-based VoIP infrastructure against flooding attacks and spam over IP telephony. *Knowledge and Information Systems* 38, 2 (2014), 491–510.

[5] Sebastian Anthony. 2015. GitHub Battles "Largest DDoS" in Site's History, Targeted at Anti-Censorship Tools. Retrieved March 6, 2021 from https://arstechnica.com/information-technology/2015/03/github-battles-largest-ddos-in-sites-history-targeted-at-anti-censorship-tools/.

[6] Jari Arkko, Gonzalo Camarillo, Tao Haukka, and Vesa Torvinen. 2003. Security Mechanism Agreement for the Session Initiation Protocol (SIP). RFC 3329. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc3329.

[7] India Ashok. 2016. Hackers Leave Finnish Residents Cold After DDoS Attack Knocks Out Heating Systems. Retrieved March 6, 2021 from https://www.ibtimes.co.uk/hackers-leave-finnish-residents-cold-after-ddos-attack-knocks-out-heating-systems-1590639.

[8] Hitesh Ballani and Paul Francis. 2008. Mitigating DNS DoS attacks. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'08)*. 189–198.

[9] Bradley Barth. 2017. DDoS Attacks Delay Trains, Halt Transportation Services in Sweden. Retrieved March 6, 2021 from https://www.scmagazineuk.com/ddos-attacks-delay-trains-halt-transportation-services-sweden/article/1473963.

[10] Sajal Bhatia, George Mohay, Alan Tickle, and Ejaz Ahmed. 2011. Parametric differences between a real-world distributed denial-of-service attack and a flash event. In *Proceedings of the International Conference on Availability, Reliability, and Security (ARES'11)*. 210–217.

[11] Monowar H. Bhuyan, Hirak Kashyap, Dhruba Kumar Bhattacharyya, and Jugal K. Kalita. 2014. Detecting distributed denial of service attacks: Methods, tools and future directions. *Computer Journal* 57, 4 (2014), 537–556.

[12] Security Boulevard. 2018. New DemonBot Discovered. Retrieved March 6, 2021 from https://securityboulevard.com/2018/10/new-demonbot-discovered/.

[13] Randy Bush and Rob Austein. 2013. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc6810.

[14] Enrico Cambiaso, Gianluca Papaleo, and Maurizio Aiello. 2014. SlowDroid: Turning a smartphone into a mobile attack vector. In *Proceedings of the International Conference on Future Internet of Things and Cloud (FITC'14)*. 405–410.

[15] Enrico Cambiaso, Gianluca Papaleo, and Maurizio Aiello. 2017. Slowcomm: Design, development and performance evaluation of a new slow DoS attack. *Journal of Information Security and Applications* 35 (2017), 23–31.

[16] Enrico Cambiaso, Gianluca Papaleo, Giovanni Chiola, and Maurizio Aiello. 2013. Slow DoS attacks: Definition and categorisation. *International Journal of Trust Management in Computing and Communications* 1, 3–4 (2013), 300–319.

[17] Enrico Cambiaso, Gianluca Papaleo, Giovanni Chiola, and Maurizio Aiello. 2015. Designing and modeling the slow next DoS attack. In *Proceedings of the International Joint Conference on Computational Intelligence in Security for Information Systems Conference (CISIS'15)*. 249–259.

[18] Giovanni Chiola Cambiaso, Enrico and Maurizio Aiello. 2019. Introducing the SlowDrop attack. *Computer Networks* 150 (2019), 234–249.

[19] Xiaofan Cao. 2007. *Model Selection Based on Expected Squared Hellinger Distance*. Colorado State University.

[20] João Marcelo Ceron, Klaus Steding-Jessen, and Cristine Hoepers. 2012. Anatomy of SIP attacks. *login:: The Magazine of USENIX & SAGE* 37, 6 (2012), 25–32.

[21] Eric Y. Chen. 2006. Detecting DoS attacks on SIP systems. In *Proceedings of the Workshop on VoIP Management and Security (VOIPMS'06)*. 53–58.

[22] Pawel Chwalinski, Roman Belavkin, and Xiaochun Cheng. 2013. Detection of application layer DDoS attack with clustering and likelihood analysis. In *Proceedings of the GLOBECOM Workshops (GC Wkshps'13)*. 217–222.

[23] Pawel Chwalinski, Roman Belavkin, and Xiaochun Cheng. 2013. Detection of application layer DDoS attacks with clustering and bayes factors. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'13)*. 156–161.

[24] Cisco. 2013. Dynamic ARP Inspection. Retrieved March 6, 2021 from http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/dynarp.html.

[25] Cisco. 2013. Port Security. Retrieved March 6, 2021 from http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/port_sec.html.
[26] Lucian Constantin. 2012. DDoS Attacks Against US Banks Peaked at 60 Gbps. Retrieved March 6, 2021 from https://www.cio.com/article/2389721/ddos-attacks-against-us-banks-peaked-at-60-gbps.html.
[27] Apache. 2019. Apache Core Features. Retrieved March 6, 2021 from https://httpd.apache.org/docs/2.4/mod/core.html.
[28] Mark R. Crispin. 2003. Internet Message Access Protocol—Version 4rev1. RFC 3501. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc3501.
[29] GitHub. 2013. Cyber Attack Management for Metasploit. Retrieved March 6, 2021 from https://github.com/rsmudge/armitage.
[30] Kaspersky Daily. 2016. How to Not Break the Internet. Retrieved March 6, 2021 from https://www.kaspersky.com/blog/attack-on-dyn-explained/13325/.
[31] Yuri Gil Dantas, Vivek Nigam, and Iguatemi E. Fonseca. 2014. A selective defense for application layer DDoS attacks. In *Proceedings of the Joint Intelligence and Security Informatics Conference (JISIC'14)*. 75–82.
[32] Kathryn de Graaf, John Liddy, Paul Raison, John C. Scano, and Sanjay Wadhwa. 2013. Dynamic Host Configuration Protocol (DHCP) authentication using Challenge Handshake Authentication Protocol (CHAP). Patent No. US8555347B2. Issued October 8, 2013.
[33] Jacques Demerjian and Ahmed Serhrouchni. 2004. DHCP authentication using certificates. In *Proceedings of the IFIP International Information Security Conference (IFIP'04)*. 456–472.
[34] Omer Deutsch, Neta Rozen Schiff, Danny Dolev, and Michael Schapira. 2018. Preventing (network) time travel with Chronos. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'18)*. 1–15.
[35] GitHub. 2017. DHCPig. Retrieved March 6, 2021 from https://github.com/kamorin/DHCPig.
[36] Tim Dierks and Christopher Allen. 1999. The TLS Protocol Version 1.0. RFC 2246. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc2246.
[37] Christoph Dietzel, Georgios Smaragdakis, Matthias Wichtlhuber, and Anja Feldmann. 2018. Stellar: Network attack mitigation using advanced blackholing. In *Proceedings of the International Conference on Emerging Networking Experiments and Technologies (CoNEXT'18)*. 152–164.
[38] Lay G. Ding and Lin Liu. 2008. Modelling and analysis of the INVITE transaction of the session initiation protocol using coloured Petri nets. In *Proceedings of the International Conference on Applications and Theory of Petri Nets (ATPN'08)*. 132–151.
[39] Yanlan Ding and Guiping Su. 2007. Intrusion detection system for signal based SIP attacks through timed HCPN. In *Proceedings of the International Conference on Availability, Reliability, and Security (ARES'07)*. 190–197.
[40] GitHub. 2017. dns-flood-ng. Retrieved March 6, 2021 from https://github.com/cmosek/dns-flood-ng.
[41] Christos Douligeris and Aikaterini Mitrokotsa. 2004. DDoS attacks and defense mechanisms: Classification and state-of-the-art. *Computer Networks* 44, 5 (2004), 643–666.
[42] Benjamin Dowling, Douglas Stebila, and Greg Zaverucha. 2016. Authenticated network time synchronization. In *Proceedings of the USENIX Security Symposium (USENIX Security'16)*. 823–840.
[43] Ralph Droms. 1997. Dynamic Host Configuration Protocol. RFC 2131. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc2131.
[44] Surakarn Duangphasuk, Supakorn Kungpisdan, and Sumeena Hankla. 2011. Design and implementation of improved security protocols for DHCP using digital certificates. In *Proceedings of the International Conference on Networks (ICN'11)*. 287–292.
[45] Sven Ehlert, Dimitris Geneiatakis, and Thomas Magedanz. 2010. Survey of network security systems to counter SIP-based denial-of-service attacks. *Computers & Security* 29, 2 (2010), 225–243.
[46] Sven Ehlert, Ge Zhang, Dimitris Geneiatakis, Georgios Kambourakis, Tasos Dagiuklas, Jiri Markl, and Dorgham Sisalem. 2008. Two layer denial of service prevention on SIP VoIP infrastructures. *Computer Communications* 31, 10 (2008), 2443–2456.
[47] eSecurity Planet. 2018. Top 10 Distributed Denial of Service (DDoS) Protection Vendors. Retrieved March 6, 2021 from https://www.esecurityplanet.com/products/top-ddos-vendors.html.
[48] Ryan Farley and Xinyuan Wang. 2012. VoIP shield: A transparent protection of deployed VoIP systems from SIP-based exploits. In *Proceedings of the 2012 Network Operations and Management Symposium (NOMS'12)*. 486–489.
[49] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. 2003. Statistical approaches to DDoS attack detection and response. In *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03)*. 303–314.
[50] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. 1999. ypertext Transfer Protocol—HTTP/1.1. RFC 2616 Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc2616.
[51] Michael Finsterbusch, Chris Richter, Eduardo Rocha, Jean-Alexander Muller, and Klaus Hanssgen. 2013. A survey of payload-based traffic classification approaches. *IEEE Communications Surveys & Tutorials* 16, 2 (2013), 1135–1156.

[52] Forbes. 2014. The Largest Cyber Attack in History Has Been Hitting Hong Kong Sites. Retrieved March 6, 2021 from https://www.forbes.com/sites/parmyolson/2014/11/20/the-largest-cyber-attack-in-history-has-been-hitting-hong-kong-sites/.

[53] Dimitris Geneiatakis, Georgios Kambourakis, Costas Lambrinoudakis, Tasos Dagiuklas, and Stefanos Gritzalis. 2007. A framework for protecting a SIP-based infrastructure against malformed message attacks. *Computer Networks* 51, 10 (2007), 2580–2593.

[54] Dimitris Geneiatakis, Tasos Dagiuklas, Georgios Kambourakis, Costas Lambrinoudakis, Stefanos Gritzalis, Sven Ehlert, and Dorgham Sisalem. 2006. Survey of security vulnerabilities in session initiation protocol. *IEEE Communications Surveys & Tutorials* 8, 3 (2006), 68–81.

[55] Dimitris Geneiatakis, Nikos Vrakas, and Costas Lambrinoudakis. 2009. Utilizing Bloom filters for detecting flooding attacks against SIP based services. *Computers & Security* 28, 7 (2009), 578–591.

[56] Gobbler. 2003. Home Page. Retrieved March 6, 2021 from http://gobbler.sourceforge.net/.

[57] Diksha Golait and Neminath Hubballi. 2016. VoIPFD: Voice over IP flooding detection. In *Proceedings of the National Conference on Communication (NCC'16)*. 1–6.

[58] Diksha Golait and Neminath Hubballi. 2017. Detecting anomalous behavior in VoIP systems: A discrete event system modeling. *IEEE Transactions on Information Forensics and Security* 12, 3 (2017), 730–745.

[59] Hugo Gonzalez, Marc Antoine Gosselin-Lavigne, Natalia Stakhanova, and Ali A. Ghorbani. 2015. The impact of application-layer denial-of-service attacks. In *Case Studies in Secure Computing: Achievements and Trends*. CRC Press, Boca Raton, FL, 261–272.

[60] Fanglu Guo, Jiawu Chen, and Tzi-Cker Chiueh. 2006. Spoof detection for preventing DoS attacks against DNS servers. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS'06)*. 37–44.

[61] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. 1999. SIP: Session Initiation Protocol. RFC 2543. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc2543.

[62] GitHub. 2018. HTTP Unbearable Load King. Retrieved March 6, 2021 from https://github.com/grafov/hulk.

[63] CS Hub. 2018. Incident of the Week: DDoS Attack Hits 3 Banks Simultaneously. Retrieved March 6, 2021 from https://www.cshub.com/attacks/news/incident-of-the-week-ddos-attack-hits-3-banks.

[64] Neminath Hubballi and Mayank Swarnkar. 2017. BitCoding: Protocol type agnostic robust bit level signatures for traffic classification. In *Proceedings of the Global Communications Conference (GLOBECOMM'17)*. 1–6.

[65] Neminath Hubballi and Mayank Swarnkar. 2018. *BitCoding*: Network traffic classification through encoded bit level signatures. *IEEE/ACM Transactions on Networking* 26, 5 (2018), 2334–2346.

[66] Neminath Hubballi, Mayank Swarnkar, and Mauro Conti. 2020. BitProb: Probabilistic bit signatures for accurate application identification. *IEEE Transactions on Network and Service Management* 17, 3 (2020), 1730–1741.

[67] Neminath Hubballi and Nikhil Tripathi. 2017. A closer look into DHCP starvation attack in wireless networks. *Computers & Security* 65, C (2017), 387–404.

[68] Neminath Hubballi and Nikhil Tripathi. 2017. An event based technique for detecting spoofed IP packets. *Journal of Information Security and Applications* 35 (2017), 32–43.

[69] Intesab Hussain, Soufiene Djahel, Zonghua Zhang, and Farid Nait-Abdesselam. 2015. A comprehensive study of flooding attack consequences and countermeasures in session initiation protocol SIP. *Security and Communication Networks* 8, 18 (2015), 4436–4451.

[70] Imperva. 2017. Q4 2017 Global DDoS Threat Landscape. Retrieved March 6, 2021 from https://www.incapsula.com/ddos-report/ddos-report-q4-2017.html.

[71] Imperva. 2017. Slowloris. Retrieved March 6, 2021 from https://www.imperva.com/learn/application-security/slowloris/.

[72] Imperva. 2019. 2019 Global DDoS Threat Landscape Report. Retrieved March 6, 2021 from https://www.imperva.com/blog/2019-global-ddos-threat-landscape-report/.

[73] Qualys Inc. 2012. Are You Ready for Slow Reading? Retrieved March 6, 2021 from https://blog.qualys.com/securitylabs/2012/01/05/slow-read.

[74] Mordor Intelligence. 2019. DDoS Protection Market - Growth, Trends, COVID-19 Impact, And Forecasts (2021-2026). Retrieved March 6, 2021 from https://www.mordorintelligence.com/industry-reports/ddos-protection-market.

[75] Kali Tools. 2006. inviteflood Package Description. Retrieved March 6, 2021 from https://tools.kali.org/sniffingspoofing/inviteflood.

[76] Hossein Hadian Jazi, Hugo Gonzalez, Natalia Stakhanova, and Ali A. Ghorbani. 2017. Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling. *Computer Networks* 121 (2017), 25–36.

[77] Khundrakpam Johnson Singh, Khelchandra Thongam, and Tanmay De. 2016. Entropy-based application layer DDoS attack detection using artificial neural networks. *Entropy* 18, 10 (2016), 1–17.

[78] Jose Jair Santanna, Roland van Rijswijk-Deij, Rick Hofstede, Anna Sperotto, Mark Wierbosch, Lisandro Zambenedetti Granville, and Aiko Pras. 2015. Booters—An analysis of DDoS-as-a-Service attacks. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'15)*. 243–251.

[79] Georgios Kambourakis, Tassos Moschos, Dimitris Geneiatakis, and Stefanos Gritzalis. 2007. Detecting DNS amplification attacks. In *Proceedings of the International Workshop on Critical Information Infrastructures Security (CRITIS'07)*. 185–196.
[80] Angelos D. Keromytis. 2012. A comprehensive survey of voice over IP security research. *IEEE Communications Surveys and Tutorials* 14, 2 (2012), 514–537.
[81] John C. Klensin. 2008. Simple Mail Transfer Protocol. RFC 5321. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc5321.
[82] Abdelkader Lahmadi and Olivier Festor. 2012. A framework for automated exploit prevention from known vulnerabilities in voice over IP services. *IEEE Transactions on Network and Service Management* 9, 2 (2012), 114–127.
[83] Sangjae Lee, Gisung Kim, and Sehun Kim. 2011. Sequence-order-independent network profiling for detecting application layer DDoS attacks. *EURASIP Journal on Wireless Communications and Networking* 2011, 1 (2011), 50.
[84] Dave Lewis. 2017. The DDoS Attack Against Dyn One Year Later. Retrieved March 6, 2021 from https://www.forbes.com/sites/davelewis/2017/10/23/the-ddos-attack-against-dyn-one-year-later/.
[85] Allan Liska. 2016. *NTP Security: A Quick-Start Guide*. Apress.
[86] Lin Liu. 2009. Verification of the SIP transaction using coloured Petri nets. In *Proceedings of the Australasian Conference on Computer Science (ACSC'09)*. 75–84.
[87] Lin Liu. 2011. Uncovering SIP vulnerabilities to DoS attacks using coloured Petri nets. In *Proceedings of the International Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom'11)*. 29–36.
[88] GitHub. 2019. Low Orbit Ion Cannon. Retrieved March 6, 2021 from https://github.com/NewEraCracker/LOIC.
[89] Prasanta C. Mahalanobis. 1936. On the generalised distance in statistics. *National Institute of Sciences of India* 2, 1 (1936), 49–55.
[90] Aanchal Malhotra, Isaac E. Cohen, Erik Brakke, and Sharon Goldberg. 2016. Attacking the network time protocol. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'16)*.
[91] Aanchal Malhotra and Sharon Goldberg. 2016. Attacking NTP's authenticated broadcast mode. *ACM SIGCOMM Computer Communication Review* 46, 2 (2016), 12–17.
[92] A. Malhotra, M. Van Gundy, M. Varia, H. Kennedy, J. Gardner, and S. Goldberg. 2017. The security of NTP's datagram protocol. In *Proceedings of the International Conference on Financial Cryptography and Data Security (FC'17)*. 405–423.
[93] Enterprise Management. 2020. Top 10 DDoS Protection Vendors. Retrieved March 6, 2021 from https://www.em360tech.com/continuity/tech-news/top-ten/top-10-ddos-protection-vendors/.
[94] Georgios Mantas, Natalia Stakhanova, Hugo Gonzalez, Hossein Hadian Jazi, and Ali A. Ghorbani. 2015. Application-layer denial of service attacks: Taxonomy and survey. *International Journal of Information and Computer Security* 7, 2/3/4 (2015), 216–239.
[95] Hyacinth Mascarenhas. 2016. Massive 'Test' Cyberattacks Using Mirai Botnet Temporarily Knock Out Liberia's Internet. Retrieved March 6, 2021 from https://www.ibtimes.co.uk/liberia-goes-offline-temporarily-massive-test-cyberattacks-hit-west-african-nation-1589820.
[96] Anil Mehta, Neda Hantehzadeh, Vijay K. Gurbani, Tin Kam Ho, Jun Koshiko, and Ramanarayanan Viswanathan. 2011. On the inefficacy of Euclidean classifiers for detecting self-similar session initiation protocol (SIP) messages. In *Proceedings of the International Symposium on Integrated Network Management (IM'11) and Workshops*. 329–336.
[97] GitHub. 2018. Metasploit. Retrieved March 6, 2021 from https://github.com/rapid7/metasploit-framework/wiki/Nightly-Installers.
[98] Microsoft. 2019. What Is Azure Web Application Firewall on Azure Application Gateway? Retrieved March 6, 2021 from https://docs.microsoft.com/en-us/azure/web-application-firewall/ag/ag-overview.
[99] D. Mills, J. Martin, J. Burbank, and W. Kasch. 2010. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc2616.
[100] NUJP. 2019. DDoS Attacks on NUJP, Alternative Media Continue. Retrieved March 6, 2021 from https://nujp.org/headlines/ddos-attacks-on-nujp-alternative-media-continue/.
[101] Jelena Mirkovic and Peter Reiher. 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review* 34, 2 (2004), 39–53.
[102] Tal Mizrahi. 2012. Slave diversity: Using multiple paths to improve the accuracy of clock synchronization protocols. In *Proceedings of the International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS'12)*. 1–6.
[103] Tal Mizrahi. 2014. Security Requirements of Time Protocols in Packet Switched Networks. RFC 7384 (Informational). Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc7384.
[104] SourceForge. 2013. mod_antiloris. Retrieved March 6, 2021 from https://sourceforge.net/projects/mod-antiloris/.
[105] Dominia.org. 2002. mod_limitipconn. Retrieved March 6, 2020 from http://dominia.org/djao/limitipconn.html.
[106] Apache. 2019. mod_reqtimeout. Retrieved March 6, 2021 from https://httpd.apache.org/docs/trunk/mod/mod_reqtimeout.html.

[107] Jarmo Mölsä. 2005. Mitigating denial of service attacks: A tutorial. *Journal of Computer Security* 13, 6 (2005), 807–837.
[108] Maurizio Mongelli, Maurizio Aiello, Enrico Cambiaso, and Gianluca Papaleo. 2015. Detection of DoS attacks through Fourier transform and mutual information. In *Proceedings of the International Conference on Communications (ICC'15)*. 7204–7209.
[109] Phil Muncaster. 2017. Anonymous Attacks Spanish Government Sites. Retrieved March 6, 2021 from https://www.infosecurity-magazine.com/news/anonymous-attacks-spanish/.
[110] Mohamed Nassar and Olivier Festor. 2008. Monitoring SIP traffic using support vector machines. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection (RAID'08)*. 311–330.
[111] Jema David Ndibwile, A. Govardhan, Kazuya Okada, and Youki Kadobayashi. 2015. Web server protection against application layer DDoS attacks using machine learning and traffic authentication. In *Proceedings of the Annual Computer Software and Applications Conference (COMPSAC'15)*. 261–267.
[112] Georgios Oikonomou and Jelena Mirkovic. 2009. Modeling human behavior for defense against flash-crowd attacks. In *Proceedings of the International Conference on Communications (ICC'09)*. 1–6.
[113] Rachel Rose O'Leary. 2017. Bitcoin Gold Website Down Following DDoS Attack. Retrieved March 6, 2021 from https://www.coindesk.com/bitcoin-gold-website-following-massive-ddos-attack.
[114] Vasileios Pappas, Dan Massey, and Lixia Zhang. 2007. Enhancing DNS resilience against denial of service attacks. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'07)*. 450–459.
[115] Michael Patrick. 2001. DHCP Relay Agent Information Option. RFC 3046. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc3046.
[116] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. 2007. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys* 39, 1 (2007), 1–42.
[117] David C. Plummer. 1982. An Ethernet Address Resolution Protocol. RFC 826. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc826.
[118] Jon Postel and Joyce Reynolds. 1985. File Transfer Protocol (FTP). RFC 959. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc959.
[119] Amit Praseed and P. Santhi Thilagam. 2019. DDoS attacks at the application layer: Challenges and research perspectives for safeguarding web applications. *IEEE Communications Surveys & Tutorials* 21, 1 (2019), 661–685.
[120] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal, Antonio Nucci, and Edward Knightly. 2009. DDoS-Shield: DDoS-resilient scheduling to counter application layer attacks. *IEEE/ACM Transaction on Networking* 17, 1 (2009), 26–39.
[121] DARK Reading. 2018. 'Torii' Breaks New Ground for IoT Malware. Retrieved March 6, 2021 from https://www.darkreading.com/attacks-breaches/-torii-breaks-new-ground-for-iot-malware/d/d-id/1332930.
[122] Joy Reo. 2016. DDoS Attacks Plague Olympic & Brazilian Government Websites. Retrieved March 6, 2021 from https://www.corero.com/blog/ddos-attacks-plague-olympic-brazilian-government-websites/.
[123] Brennen Reynolds and Dipak Ghosal. 2003. Secure IP telephony using multi-layered protection. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'03)*. 1–15.
[124] Teri Robinson. 2020. FBI Warns of DDoS Attack on State-Level Voter Registration Website. Retrieved August 16, 2020 from https://www.scmagazine.com/home/security-news/fbi-warns-of-ddos-attack-on-state-level-voter-registration-website/.
[125] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. 2002. SIP: Session Initiation Protocol. RFC 3261. Retrieved March 6, 2021 from https://tools.ietf.org/html/rfc3261.
[126] Christian Rossow. 2014. Amplification hell: Revisiting network protocols for DDoS abuse. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'14)*. 1–15.
[127] Packet Storm. 2011. R-U-Dead-Yet Denial of Service Tool 2.2. Retrieved March 6, 2021 from https://packetstormsecurity.com/files/97738/R-U-Dead-Yet-Denial-Of-Service-Tool-2.2.html.
[128] Scapy. 2019. Welcome to Scapy's Documentation! Retrieved March 6, 2021 from https://scapy.readthedocs.io/en/latest/.
[129] Mathew J. Schwartz. 2016. DDoS Attack Slams HSBC. Retrieved March 6, 2021 from https://www.bankinfosecurity.com/ddos-attack-slams-hsbc-a-8835.
[130] Murat Semerci, Ali Taylan Cemgil, and Bulent Sankur. 2018. An intelligent cyber security system against DDoS attacks in SIP networks. *Computer Networks* 136 (2018), 137–154.
[131] Hemant Sengar, Haining Wang, Duminda Wijesekera, and Sushil Jajodia. 2008. Detecting VoIP floods using the Hellinger distance. *IEEE Transactions on Parallel and Distributed Systems* 19, 6 (2008), 794–805.
[132] Dongwon Seo, Heejo Lee, and Ejovi Nuwere. 2013. SIPAD: SIP–VoIP anomaly detection using a stateful rule tree. *Computer Communications* 36, 5 (2013), 562–574.
[133] Amey Shevtekar and Nirwan Ansari. 2009. Is it congestion or a DDoS attack? *IEEE Communications Letters* 13, 7 (2009), 546–548.

[134] Alexander Shpiner, Yoram Revah, and Tal Mizrahi. 2013. Multi-path time protocols. In *Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS'13)*. 1–6.

[135] Mark Shtern, Roni Sandel, Marin Litoiu, Chris Bachalo, and Vasileios Theodorou. 2014. Towards mitigation of low and slow application DDoS attacks. In *Proceedings of the International Conference on Cloud Engineering (IC2E'14)*. 604–609.

[136] Karanpreet Singh, Paramvir Singh, and Krishan Kumar. 2017. Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges. *Computers & Security* 65, C (2017), 344–372.

[137] Karanpreet Singh, Paramvir Singh, and Krishan Kumar. 2018. User behaviour analytics-based classification of application layer HTTP-GET flood attacks. *Journal of Network and Computer Applications* 112 (2018), 97–114.

[138] Khundrakpam Johnson Singh and Tanmay De. 2017. MLP-GA based algorithm to detect application layer DDoS attack. *Journal of Information Security and Applications* 36 (2017), 145–153.

[139] Sujatha Sivabalan and P. J. Radcliffe. 2013. A novel framework to detect and block DDoS attack at the application layer. In *Proceedings of the Conference on Tencon-Spring (TENCON'13)*. 578–582.

[140] Network Security Team of CNR Italy. 2015. Download SlowDroid DoS Tool 0.87.5 APK. Retrieved March 6, 2021 from https://downloadapk.net/SlowDroid-DoS-Tool.html.

[141] Sergey Shekyan. 2017. Application Layer DoS attack simulator. Retrieved March 6, 2021 from https://github.com/shekyan/slowhttptest.

[142] Oscar Andersson. 2013. Slowloris. Retrieved March 6, 2021 from https://github.com/Ogglas/Orignal-Slowloris-HTTP-DoS/blob/master/slowloris.pl.

[143] StarTrinity. 2019. StarTrinity SIP Tester (Call Generator, Simulator)—VoIP Monitoring and Testing Tool. Retrieved March 6, 2021 from http://startrinity.com/VoIP/SipTester/SipTester.aspx.

[144] Zhiyang Su, Hao Ma, Xiaojun Zhang, and Bei Zhang. 2011. Secure DHCPv6 that uses RSA authentication integrated with self-certified address. In *Proceedings of the International Workshop on Cyberspace Safety and Security (CSS'11)*. 39–44.

[145] Changhua Sun, Bin Liu, and Lei Shi. 2008. Efficient and low-cost hardware defense against DNS amplification attacks. In *Proceedings of the Global Telecommunications Conference (GLOBECOM'08)*. 1–5.

[146] Suriadi Suriadi, Douglas Stebila, Andrew Clark, and Hua Liu. 2011. Defending web services against denial of service attacks using client puzzles. In *Proceedings of the International Conference on Web Services (ICWS'11)*. 25–32.

[147] Jin Tang and Yu Cheng. 2011. Quick detection of stealthy SIP flooding attacks in VoIP networks. In *Proceedings of the International Conference on Communications (ICC'11)*. 1–5.

[148] Jin Tang, Yu Cheng, Yong Hao, and Wei Song. 2014. SIP flooding attack detection with a multi-dimensional sketch design. *IEEE Transactions on Dependable and Secure Computing* 11, 6 (2014), 582–595.

[149] Jin Tang, Yu Cheng, and Chi Zhou. 2009. Sketch-based SIP flooding detection using Hellinger distance. In *Proceedings of the Global Telecommunications Conference (GLOBECOM'09)*. 1–6.

[150] CA Technologies. 2018. Insider Threat 2018 Report. Retrieved March 6, 2021 from https://ca-security.inforisktoday.com/whitepapers/insider-threat-2018-report-w-4131.

[151] NTPsec. 2019. Welcome to NTPsec. Retrieved March 6, 2021 from https://www.ntpsec.org/.

[152] Nikhil Tripathi and Neminath Hubballi. 2015. Exploiting DHCP server-side IP address conflict detection: A DHCP starvation attack. In *Proceedings of the International Conference on Advanced Networks and Telecommunications Systems (ANTS'15)*. 19–21.

[153] Nikhil Tripathi and Neminath Hubballi. 2016. A probabilistic anomaly detection scheme to detect DHCP starvation attacks. In *Proceedings of the International Conference on Advanced Networks and Telecommunications Systems (ANTS'16)*. 1–6.

[154] Nikhil Tripathi and Neminath Hubballi. 2018. Detecting stealth DHCP starvation attack using machine learning approach. *Journal of Computer Virology and Hacking Techniques* 14, 3 (2018), 233–244.

[155] Nikhil Tripathi and Neminath Hubballi. 2018. Slow rate denial of service attacks against HTTP/2 and detection. *Computers & Security* 72, C (2018), 255–272.

[156] Nikhil Tripathi and Neminath Hubballi. 2021. Preventing time synchronization in NTP broadcast mode. *Computers & Security* 102 (2021), 102–135.

[157] Nikhil Tripathi, Neminath Hubballi, and Yogendra Singh. 2016. How secure are web servers? An empirical study of slow HTTP DoS attacks and detection. In *Proceedings of the International Conference on Availability, Reliability, and Security (ARES'16)*. 454–463.

[158] Nikhil Tripathi and Babu M. Mehtre. 2013. An ICMP based secondary cache approach for the detection and prevention of ARP poisoning. In *Proceedings of the International Conference on Computational Intelligence and Computing Research (ICCIC'13)*. 1–6.

[159] Nikhil Tripathi and Babu M. Mehtre. 2014. Analysis of various ARP poisoning mitigation techniques: A comparison. In *Proceedings of the International Conference on Control, Instrumentation, Communication, and Computational Technologies (ICCICCT'14)*. 125–132.

[160] Nikhil Tripathi, Mayank Swarnkar, and Neminath Hubballi. 2017. DNS spoofing in local networks made easy. In *Proceedings of the International Conference on Advanced Networks and Telecommunications Systems (ANTS'17)*. 1–6.

[161] Tripwire. 2016. DYN Restores Service After DDoS Attack Downed Twitter, Spotify, Others. Retrieved March 6, 2021 from https://www.tripwire.com/state-of-security/latest-security-news/dyn-restores-service-ddos-attack-brought-twitter-spotify-others/.

[162] Ravichander Vaidyanathan, Abhrajit Ghosh, Yuu-Heng Cheng, Akira Yamada, and Yutaka Miyake. 2012. Method and apparatus for detecting spoofed network traffic. Patent No. US8281397B2. Issued October 2, 2012.

[163] Jin Wang, Xiaolong Yang, and Keping Long. 2011. Web DDoS detection schemes based on measuring user's access behavior with large deviation. In *Proceedings of the Global Telecommunications Conference (GLOBECOM'11)*. 1–5.

[164] Yu-Sung Wu, Saurabh Bagchi, Sachin Garg, and Navjot Singh. 2004. SCIDIVE: A stateful and cross protocol intrusion detection architecture for voice-over-IP environments. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'04)*. 433–442.

[165] Yi Xie and Shun-Zheng Yu. 2009. A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. *IEEE/ACM Transactions on Networking* 17, 1 (2009), 54–65.

[166] Yi Xie and Shun-Zheng Yu. 2009. Monitoring the application-layer DDoS attacks for popular websites. *IEEE/ACM Transactions on Networking* 17, 1 (2009), 15–25.

[167] Chuan Xu, Guofeng Zhao, Gaogang Xie, and Shui Yu. 2014. Detection on application layer DDoS using random walk model. In *Proceedings of the International Conference on Communications (ICC'14)*. 707–712.

[168] Takeshi Yatagai, Takamasa Isohara, and Iwao Sasase. 2007. Detection of HTTP-GET flood attack based on analysis of page access behavior. In *Proceedings of the Pacific Rim Conference on Communications, Computers, and Signal Processing (PacRim'07)*. 232–235.

[169] Saman Taghavi Zargar, James Joshi, and David Tipper. 2013. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Communications Surveys & Tutorials* 15, 4 (2013), 2046–2069.

[170] Heng Zhang, Ahmed Taha, Ruben Trapero, Jesus Luna, and Neeraj Suri. 2016. SENTRY: A novel approach for mitigating application layer DDoS threats. In *Proceedings of the International Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom'16)*. 465–472.

[171] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. 2015. Connection-oriented DNS to improve privacy and security. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'15)*. 171–186.

Received September 2019; revised December 2020; accepted January 2021