A Multilayer Overlay Network Architecture for Enhancing IP Services Availability Against DoS

Dimitris Geneiatakis, Georgios Portokalidis and Angelos D. Keromytis

Department of Computer Science,Columbia University, New York, NY, USA
{dgen,porto,angelos}@cs.columbia.edu

Abstract. Protection against Denial of Service (DoS) attacks is a challenging and ongoing problem. Current overlay-based solutions can transparently filter unauthorized traffic based on user authentication. Such solutions require either pre-established trust or explicit user interaction to operate, which can be circumvented by determined attackers and is not always feasible (*e.g.,* when user interaction is impossible or undesirable). We propose a Multi-layer Overlay Network (MON) architecture that does not depend on user authentication, but instead utilizes two mechanisms to provide DoS resistant to any IP-based service, and operates on top of the existing network infrastructure. First, MON implements a threshold-based intrusion detection mechanism in a distributed fashion to mitigate DoS close to the attack source. Second, it randomly distributes user packets amongst different paths to probabilistically increase service availability during an attack. We evaluate MON using the Apache web server as a protected service. Results demonstrate MON nodes introduce very small overhead, while users' service access time increases by a factor of 1.1 to 1.7, depending on the configuration. Under an attack scenario MON can decrease the attack traffic forwarded to the service by up to 85%. We believe our work makes the use of overlays for DoS protection more practical relative to prior work.

1 Introduction

Denial of service attacks (DoS), and their distributed counterparts DDoS, frequently cause disruptions, and sometimes even complete outages, of services offered over the Internet. E-mail, financial, publishing, and even e-government services have repeatedly been the targets of such attacks, which have only intensified with the proliferation of botnets that employ compromised PCs ("zombies") to perform large scale DDoS attacks. Recently, we have also witnessed incidents, where groups of users deliberately performed DDoS attacks using freely available tools, like the low orbit ion cannon [1].

Centralized approaches such as [23,12,27] have not been able to thwart DDoS attacks, mainly because they can also be congested, while distributed network level solutions [15,33,16] require operators to deploy and manage new, and potentially complex, architectures. On the other hand, distributed architectures based on overlay networks [18,28,29,6,20,21,5] *can* operate on existing network infrastructure, and have shown to be able to withstand attacks involving *millions* of attackers. Overlay based solutions can be categorized in three types:

1. Strict access: These approaches [18,6,20,5,29] assume that the users of the protected service are pre-authorized, and only them are allowed to access the service. They cater to scenarios like attacks against *emergency services* used during disasters.
2. Relaxed access: Such networks [37,36] still use some form of authentication, but they either allow anyone to “register” with the service, or mix authorized and unauthorized users. In the first case, authentication is used to uniquely identify clients without relying on their IP address (*i.e.,* spoofing is no longer relevant), while in the latter authorized user traffic is given precedence.
3. Open access: Open systems [28,29] can be accessed by everyone. They limit or distinguish attackers by using user sessions and require some type of user interaction, like a CAPTCHA [32], that separates real users from programs.

These techniques mitigate the consequences of DoS attacks using authentication or user interaction to distinguish authorized from unauthorized traffic, limiting or completely warding off the latter. However, limited access systems are not applicable to open Internet services like the world wide web (WWW), and, more importantly, they are still vulnerable to DoS attacks from authenticated users. Also, user interaction mechanisms, like CAPTCHAs, is impractical, while they are not impervious to attacks either [8].

We propose *a new multilayer overlay network* (MON) architecture that builds on the advantages of ticketing and multi-path traffic distribution proposed in previous relaxed access work [29] to *enable open access* to the protected service *without* requiring any user interaction. Our solution operates on existing network infrastructure and consists of multiple layers of nodes that are interposed between the client and the protected service. MON *collectively* applies a throttling-based DoS prevention mechanism (DPM) that alleviates the effects of DoS attacks. The mechanism is applied in a distributed fashion by lightweight DoS detection and mitigation engines (DDME) running on each overlay node. A client accesses the service by contacting any of the nodes of the first layer, which randomly routes his packets through the overlay, increasing MON’s robustness in the case of node failures and DoS attacks.

We implemented a prototype using an uncomplicated practical threshold-based filtering mechanism. Briefly, the DDME on each node monitors the IP packets being send to the service, calculating the packet sending rate of each client based on his IP address. When a client exceeds a predefined threshold, some or all of his packets are dropped depending on the employed security policy. Note that we do not invent a new defense against DoS attacks, but instead propose the distributed application of prevention mechanisms based on overlay network architecture. Results show that the overhead introduced by MON nodes is small (in the range of 30 and 50 microseconds), while using MON to retrieve a web site served by the Apache web server increases the client’s access time by a factor of 1.1× to 1.7×, depending on the configuration. Moreover, we demonstrate that the proposed solution is able to throttle and block malicious network flows effectively when a service is under a DoS attack.

The rest of the paper is organized as follows. In Sect. 2, we describe the types of DoS attacks that we tackle with this work. In Sect. 3, we describe the architecture, while we discuss implementation details in Sect. 4. In Sect. 5, we evaluate MON in terms of overhead and effectiveness. Sect. 6 presents an overview of the related work, and compares it with ours. Finally, we conclude this paper and give some pointers for future work in Sect. 7.

2 Threat Model

The goal of DoS attacks is to render a targeted services inaccessible to legitimate users, by draining server resources like memory and CPU, or consuming network bandwidth [22]. The simplest DoS attacks consume network bandwidth and server resources simply by saturating the target with a high number of requests, generated either from a single or multiple sources. However, as server processing power and network bandwidth increases alternative, more sophisticated, methods requiring fewer connections have been devised. For instance, a flood of SYN TCP packets can be used to prevent the establishment of new connections with a server [9]. Other approaches exploit the way the service handles users' requests to achieve the same goal [7]. Based on their method of operation, we classify DoS attacks into the following categories:

(a) *Attacks consuming network bandwidth.* These attacks aim to congest the target's network by generating an overwhelmingly large number of data packets.
(b) *Attacks consuming server resources.* Malicious users send a large number of otherwise legitimate requests to the server to consume its resources. They usually require less traffic than (a) to be sent by a malicious user, as network bandwidth increases at a higher rate than computational power and storage.
(c) *Attacks consuming limited operating system (OS) resources.* These attacks exploit the way the target's software and OS operates to consume limited resources such as file and socket descriptors, and TCP connections.
(d) *Attacks exploiting server bugs.* Software frequently has bugs like null pointers [14] and deadlocks [17]. If such bugs can be triggered using user input, an attacker sending the appropriate data to the server can cause it to crash, or simply stop responding to new requests. This type of attacks are beyond the scope of this paper, since are tackled by solutions such as [34,10].

3 A Secure Multilayer Overlay Network Architecture

The goal of the MON architecture is to improve the availability of critical services under a DoS attack, by "hiding" the protected service behind a semistructured overlay network. Users of the service communicate with it through the overlay, which by design and by means of a distributed DoS prevention mechanism (DPM) mitigates DoS attacks. We adopt an overlay network architecture to incorporate certain properties in our design. Specifically:

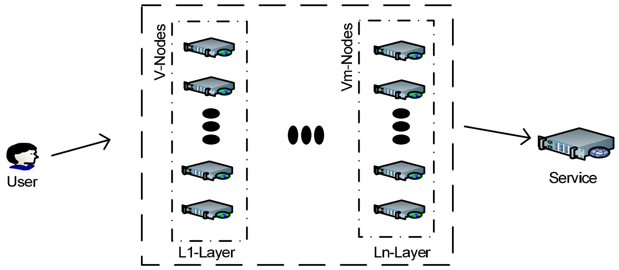

Fig. 1. The multilayer overlay network (MON) architecture. Multiple layers of nodes are interposed between the clients and the provided service. The number of layers is configurable at overlay setup, and can be tuned to favor performance over resistance to intrusion by reducing the degree of internal redirection.

(a) Easy deployment on existing network infrastructure.
(b) Transparency to end-users.
(c) Decentralized design inherently strong against DoS attacks, and suitable for applying intrusion prevention mechanisms in a distributed way.

An overview of the MON architecture is illustrated in Fig. 1. The first layer is the entry point between users and the service. It ensures that no spoofed traffic is forwarded to the protected service based on tickets (discussed in Sect 3.2), and blocks attacks using the DPM. The rest of the layers remain invisible to end-users, while they also use the DPM to protect from malicious or compromised nodes of the previous layer. However, if an attacker discovers and compromises a node in the last layer, the service will be exposed. The number of layers deployed, depends on the network delay that the protected service can tolerate, and the desired protection level that must be established.

MON provides open access to users, without requiring user interaction or pre-established trust between clients and the provided service. Instead, it throttles user traffic that exhibits abnormal behavior on a per- flow and ticket basis. Also, MON operates at the IP level, supporting multiple transport layer protocols like TCP, UDP and SCP.

3.1 The MON Structure

MON consists of a number of overlay-nodes N_n, which are organized in different layers L_m. Clients can access the protected service through the first layer (entry point) nodes after acquiring an access ticket. Tickets protect against DoS attacks themselves, as they allow us to control the rate new clients can connect to the network, but they also allow us to validate the sender of each packet. The nodes of the first layer know the nodes of the next one, where they forward packets to, and so forth (*i.e.,* nodes of L_i know the nodes L_{i+1}). As a result, the actual location (IP address) of the service is only known by the nodes in the last layer.

Also nodes instead of routing packets through a specific path, they randomly route it in one of the available nodes of the next layer. The last hop of the

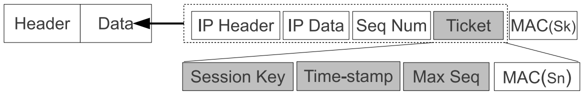

Fig. 2. MON packet structure. User's IP traffic is encapsulated into a new transport layer packet to enable routing by MON nodes, including an encrypted access ticket and message authentication code to ensure MONs' message integrity, and authenticity. Only entry nodes can decrypt and validate the ticket.

employed architecture delivers the packets to the protected service. If one of the nodes becomes "unavailable" due to network congestion or software failure, traffic is forwarded through a different node in the same layer.

Consequently, an attacker would have to concurrently flood all nodes of at least one layer of the overlay to successfully disrupt the service. Assuming that the layer with the smallest number of nodes is L_v and contains V nodes, the attacker would have to successfully flood all V nodes. However, since the route followed by packets is chosen randomly, and every node implements the DPM, which throttles flows classified as malicious, this task is far from trivial. Even if an attacker compromises a first layer node (*e.g.,* by exploiting a vulnerability in its software) and uses it in its DoS attack, the next layer nodes can identify and mitigate the attack, while the service remains hidden. *Hence, MON is resistant to DoS attacks by design.* All the nodes in MON architecture operate autonomously. MON encapsulates at the client side (see also Sect. 3.2) user packets into a new packet (similarly to IP-in-IP encapsulation), as illustrated in Fig. 2.

3.2 Users and the Ticket Mechanism

Users access a MON-protected service through the first layer's nodes. This is done transparently, by installing an *end-user module* at the client, that captures data packets destined for the service and re-routes them to one of the entry points of MON. To overcome single points of failure, users' traffic is randomly distributed among the nodes of L_1, similarly to the random routing described in Sect. 3.1. MON-enabled users are allowed to connect to the protected service, only if they have acquired a session key and the corresponding ticket from an entry point. Particularly, a client receives a session key and a valid ticket by issuing an access request to a randomly chosen entry node. The request is performed over a secure channel (*e.g.,* using SSL) to ensure the confidentiality of the session key sent to the client.

Every session key S_k is computed based on the user's IP and a pseudo-random id generated by the contacted node, encrypted using a master key K_n shared among all MON nodes using the following formula.

$$S_k = Enc(K_n, User\ IP || Random\ Id)$$

The *ticket* sent to the user includes the user's session key, a time-stamp, the maximum number of packets allowed to be sent with the specific ticket, and a message authentication code (MAC) generated using the master key. The latter enables any MON-node to validate the integrity and authenticity of every ticket. Finally, the entire ticket is also encrypted using the master key and sent to the user as a response.

$$ticket = Enc(K_n, session\ key||timestamp||MAC(K_n))$$

Clients include the ticket and a MAC based on their session key in all subsequent packets (see Fig. 2). MON nodes are able to decrypt the ticket and validate MON packet's integrity and authenticity, using the shared master secret key. This way an attacker cannot spoof a client's packets (*i.e.,* send packets pretending to be another client), and nodes can validate tickets without having to store additional information for the session key and the ticket.

3.3 A Collaborative DoS Detection and Mitigation Mechanism

MON is robust by design, but as the resources of the attackers grow, additional protective measures are necessary. Although, various centralized intrusion detection and prevention solutions against DDoS attacks have been proposed in literature [25], they can also be congested and rendered useless by the attack against the protected service. MON adopts a collaborative DoS prevention mechanism (DPM) that is applied individually by every node, and is thus more resistant to attacks.

The core of the mechanism is implemented by the DoS detection and mitigation engines (DDME) running on the nodes. Its goal is to throttle or drop incoming packets, whenever it detects that a client is performing a DoS attack. It works by classifying incoming packets into IP network flows based on their source, destination IP addresses, and the ticket (as it uniquely identifies a client). This way, we can detect abnormal behavior and match it to a specific user. We use IP layer flows, primarily because we desire that MON protects services independently of the transport and application protocol being used. Secondly, this will allow us to easily expand MON to support the IPv6 protocol.

Each DDME keeps track of all the incoming flows using a flow information monitor (FIM). The FIM records the number of packets observed per flow, and the time that each flow was initiated. Then, the DDME periodically examines (with a period T) whether the packet rate of any flow has exceeded the legitimate configurable threshold. If this is the case, packets belonging to this specific flow are "punished" that is, delayed or dropped. For the scope of this work, we adopt an exponential punishment mechanism, as in [2]. This technique is widely used in networking to penalize users causing conflicts during transmission. The delay introduced by the DDME for "punished" flows is computed by the following formula $delay = delay \times 2$. Devising mechanisms to calculate an appropriate threshold is beyond the scope of this paper. In the future, we plan to investigate threshold calculation mechanisms like the ones proposed in [15].

4 Implementation

4.1 Ticket Acquisition

A user inquires a ticket from the service by sending a ticket acquisition request to a randomly chosen MON node. The request contains the user's RSA public key and the corresponding digital signature. As soon as a node receives a ticket request, it validates the included digital signature and generates the response message, which consists of the session key and the ticket as described in Sect. 3.2. Note that the session key and the ticket are encrypted using the user's RSA public key and the AES algorithm. To compute the MAC, both for the ticket and the response message, we use an HMAC function based on SHA-1. All the cryptographic functions used are part of the OpenSSL library [31].

4.2 MON-enabled Users

On the client side, we developed a service, namely MONeU, operating transparently to existing applications in order to deliver user traffic to MON nodes. User packets destined to the protected service, instead of following the normal network path, are sent to a virtual interface implemented using the tun pseudo-device driver [19]. As soon as a packet is received in this interface, the MONeU service encapsulates it in a new UDP packet, which includes the ticket, the packet's sequence number, and an SHA-1 HMAC computed on the whole MON packet to protect its integrity. This new packet is forwarded to a randomly chosen first layer MON node. The technique shields end-users from single point failures, as their traffic does not follow a specific path. This stands even for packets belonging to the same session. As a result, a malicious user needs to compromise all available MON entry nodes to cause a DoS to a legitimate user.

In the current implementation the available first layer nodes are included in a pre-defined list. However, an alternative solution for MONeU to receive the list is through DNS. In that case, the DNS instead of returning the actual IP address of the protected service, will send back the IP addresses of all available first layer nodes.

Whenever a response packet arrives, the MONeU service passes it to the virtual interface, which is responsible for delivering it to the corresponding application. The decision to use UDP instead of TCP for packet encapsulation is based on the fact that the encapsulation of TCP within TCP is not efficient [30].

4.3 MON Nodes

MON nodes are the core of the proposed architecture and are responsible for detecting and mitigating the effects of malicious flows (*i.e.,* flows responsible for a DoS attack), and routing user traffic to the actual destination. First, entry nodes validate MON packet authenticity and integrity. To accomplish this task the nodes decrypt the ticket using AES and the secret key S_n shared among the MON nodes. They also ascertain its validity using the same key to validate the

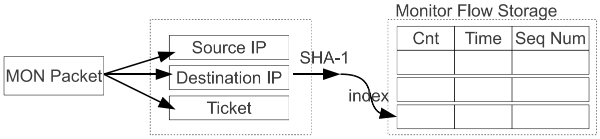

Fig. 3. MON flow information monitor (FIM) architecture. FIM uses the hash value of the source and destination IP as an index to the monitor flow storage, and records the number of packets per IP flow, the time-stamp of the first packet in the flow, as well as MON packet sequence number.

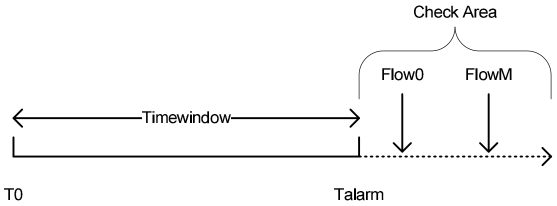

Fig. 4. DDME's time-window structure. The DDME checks all the flows exceeding the allowed time in FIM.

SHA-1-based HMAC. If the ticket is valid, we extract the session key S_k, which is used to confirm the authenticity and integrity of the whole packet, similarly to the process used to validate the ticket.

After message and ticket validity is confirmed, the node extracts the source and destination IP, and combines them with the ticket in order to uniquely identify each flow. The FIM module records for every flow the number of received packets, the arrival time of the first packet in the flow, and the sequence number of the last received packet to protect MON against replay attacks. All numbers are 32-bits long, so we store 96 bits per flow, which corresponds to 12 MBs of data for one million users. For storage and searching efficiency, instead of recording the source and destination IP for every flow, we use the remainder of the hash value of the flow information with the maximum number of allowed flows (see the formula below) for identifying a record in the FIM storage.

$$index = hash(SrcIP||DestIP||Ticket)\ mod\ MAXCON$$

The main disadvantage of this approach is that if we receive more connections than FIM's capacity ($MAXCON$), it will cause a "conflicting" FIM record, affecting the DDME's accuracy. Otherwise, "conflicts" in the FIM storage solely depend on the hash function's collision probability [35]. For every received packet, the FIM computes the flow's index in order to update the packet counter, and saves the timestamp of the first packet in the flow (t_{flow}). Figure 3 illustrates FIM's general architecture. Note that the hash and modulo functions we use are OpenSSL's SHA-1 [31] and GMP's modulo [13].

The DDME is triggered by a SIGALARM signal every T_{window} seconds in order to check whether any flow in the FIM has exceeded a pre-defined threshold.

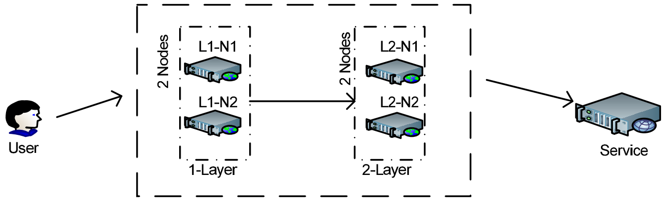

Fig. 5. The two-layer test-bed we employed for the evaluation of the MON architecture.

If this is the case, the packet is either punished by introducing a delay to its forwarding. Note that the DDME inspects all the flows found "outside" of the T_{window}, where $t_{alarm} - t_{flow} > T_{window}$ (see also Fig. 4). As soon as the DDME checks a flow, it resets its corresponding record in the FIM.

Afterwards, the DDME encapsulates the incoming packet into a new UDP packet, and forwards it to a randomly selected node at the next layer. When the packet reaches a node in the last layer of MON, it is decapsulated to obtain the original packet, and is forwarded to the protected service via a RAW socket.

For implementing the networking functionality of MON nodes, we relied on Linux's system calls. Based on current implementation MON nodes can be deployed on routers, as well as desktop PCs, that are willing to participate in the MON architecture, assuming that they are running a *nix type OS.

5 Evaluation

To evaluate our architecture in terms of performance and effectiveness, we deployed a two-layered prototype as depicted in Fig. 5. Table 1 lists the technical characteristics of the systems where we installed MON nodes. As a protected service, we utilized the Apache web server. We employed five scenarios, where a client is requesting files of different sizes from the Apache web server (sizes of 150KB, 500KB, 1MB, 5MB, and 13 MB). In all five scenarios, the client accesses the service *directly*, using MON *without DPM* functionality, and lastly *with DPM* functionality. Furthermore, to validate MON's correctness and efficiency, all the scenarios were executed in two different configurations:

1. All MON nodes running in our lab's local network (LAN configuration).
2. MON's 1^{st} layer nodes running in a different network connected to the Internet with an ADSL connection, while the 2^{nd} layer nodes were in the lab's network (ADSL configuration).

5.1 Performance Evaluation

To quantify the overhead MON introduces to the protected service, we measured the end-to-end service access time as well as the overhead introduced by the MON nodes itself. On both configurations, when users access the service using

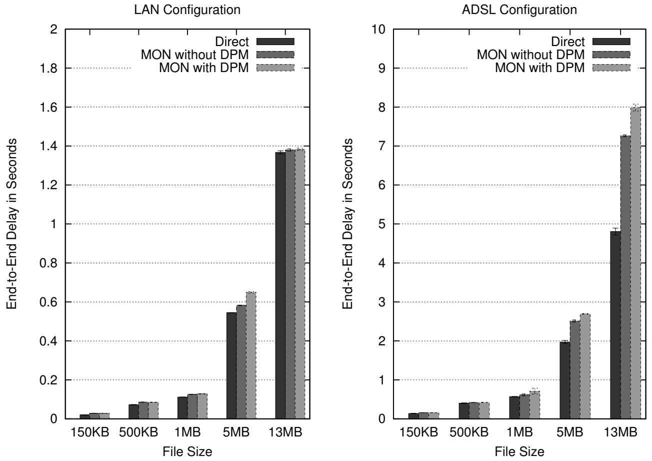

Fig. 6. End-to-end service access time (SAT) comparison among direct access, and MON without and with DPM. User SAT is increased by a factor of 1.1 to 1.7, depending on the configuration.

MON, they do not experience any apparent delay in service access time (SAT). The average SAT is very close in all the scenarios, as depicted in Fig. 6. In the results, we include the case of using MON without the DPM functionality, in order to show the lower bound of overhead imposed by the overlay network itself. In the LAN configuration, MON increases the end-to-end SAT by a factor of 1.04 to 1.4, while in the ADSL configuration by a factor of 1.1 to 1.7. Though this increase might be considered significant, we believe that is an acceptable "cost" for enhancing IP service availability. Note that the total transfer time is affected by the network link's quality in which MON nodes are installed. For instance, the % difference in average SAT between the LAN configuration and direct access ranges from 4% to 40%. The same stands for the ADSL configuration, however, in scenario 5 there is an increase of 66%.

Regarding the overhead introduced by entry and core MON nodes, the average ranges between 30 and 50 microseconds (see Fig. 7). As we use the same underlying infrastructure for both configurations, there are no differences in the

Table 1. Characteristics of the systems hosting the MON nodes in the test-bed.

Name	CPU Speed	Memory	Other characteristics
L1-N1	2.4 GHz	4Gb	Ubuntu 10.04, Intel Core i5 laptop
L1-N2	2.4 GHz	2Gb	Ubuntu 10.10, virtual machine
L2-N1	2.8 GHz	3Gb	Ubuntu 10.04, Xeon 2xCPU
L2-N2	2.4 GHz	2Gb	Ubuntu 10.10, virtual machine

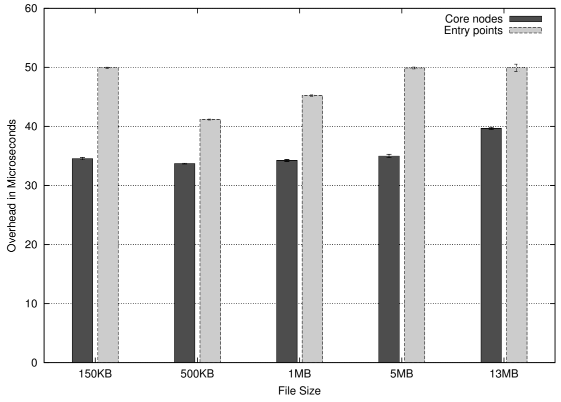

Fig. 7. Average processing time introduced by entry points and core nodes in MON.

delay introduced by each MON node. This fact validates our initial hypothesis regarding the end-to-end SAT fluctuations in the ADSL configuration.

Based on our experimental results, we deduce that MON does not influence the SAT, while the overhead imposed on the provided service by each MON node is negligible. However, it should be noted that the end-to-end delay is affected both from the number of MON layers, and the latency of the underlying network where the MON nodes are located. In the extreme case, we can optimize performance by reducing the system to one level of indirection similar to [29].

5.2 Qualitative Analysis

MON introduces little delay to end user communication with a protected service, however, its effectiveness as a protection mechanism against DoS attacks must be evaluated as well for its ability to identify and punish DoS packets. Thus, we employ a single source attack scenario in which the malicious user generates 1000 HTTP requests per second for a web page of 100 bytes. Regardless of the simplicity of this attack, the main goal of this experiment is to demonstrate MON's efficiency during such an attack, by monitoring the number of requests received by the protected service.

On the one hand, if the service is protected through traditional centralized DoS mechanisms, all the attack requests will arrive at least up to the protected service network causing a congestion either at the edge of the network or in the service itself. On the other hand, if MON is enabled it will forward all the packets belonging to the attack flow toward the service until the DDME is triggered. In the worst case, MON will deliver to the service $Tw \times Number\ of\ Packets\ Per\ Second$, however, the number of packets received by the service depends on the Tw that the DDME is triggered, and the punishment model. Since, the DDM is triggered the attack will be identified and traffic will be throttled. Particularly, under our attack scenario the number of requests received by the service is on average 107 HTTP requests, which corresponds to 323 packets. The majority of these packet are delivered to the service because the DDME had not triggered yet. Using MON we achieve a reduction on the attack traffic receive by the service up to 85%. The number of times a flow exceeds the legitimate threshold affects the punishment delay introduced on a flow exponentially, as described in Sect. 3. So if a malicious user exceeds more than 10 times the threshold the introduced delay reaches to 1200 seconds.

Similar is the case of multiple sources DoS attack, as we can consider it as N separate single source DoS attacks. The only difference is the amount of traffic that will be delivered by MON towards to the service until the DDME is triggered, which is $N \times Tw \times Number\ of\ Packet\ Per\ Second$. Note that a malicious user can still spoof an IP address and get a valid MON ticket, but he is not able to affect the communications of other users, as MON distinguishes flows on a per ticket basis. However, to effectively shield a service against these types of DoS attacks, MON nodes should be widely deployed across the Internet. Also, additional IP spoofing protection mechanisms would further fortify the system.

All in all, MON can be used to defend against DoS attacks targeting service network bandwidth or server resources (briefly described in Sect. 2) by exploiting the autonomously operating MON nodes to detect and discard DoS traffic.

6 Related Work

Overlay networks emerged as a solution to overcome the limitations of the Internet architecture, providing new services and applications in a distributed end-to-end fashion. To that end, previous work build on overlay networks as an alternative underlying mechanism to enhance Internet service security and availability. The very first work exploiting overlay networks to enhance service availability is RON [4]. RON introduces an application-level routing and packet forwarding service that enables end-points to monitor the quality of links in the network, and detect path failures in order to choose an alternative path in a few seconds. However, the predominant work exploiting overlay networks to improve network service security is presented in [18]. The SOS architecture routes traffic to the protected service only from a set of pre-determined authorized users, based on protocols such as IPsec [26] and SSL [31]. A similar approach is presented in [3].

Several variations of SOS have been proposed in literature [28,29,20,21,6,5]. [28] extends its functionality to defend against botnets using CAPTCHA [32], while [29] introduces an architecture for enhancing service availability build on multi-path and stateless tokens. In ODON [20], users are verified through credentials to access the overlay network. They establish a session token among end-users, ODON nodes, and the final service used to verify traffic, and provide integrity and confidentiality services. [21] proposes a federated overlay network (FONet) architecture to protect services from DoS attacks. FONet forwards only the traffic originating from the federation to the protected service, and filters the other traffic on the edge of the domains participating in the FONet. [5] introduces an intermediate interface by overlay nodes to hide the location of application sites during DoS attacks. Similarly, this solution allows communication only among confirmed entities; meaning that any packet must be authenticated through the proposed architecture. [6] protects a target using a filter that drops any packet whose source address is not approved. In the case of a DoS attack, rather than processing all arriving packets, the target's filter processes only a subset of received packets, and drops all the remaining.

Phalanx [11] follows a similar approach to SOS, leveraging the power of swarms to combat DoS. A client communicating with a destination sends its packets through a random sequence of entities called "mailboxes". Mailboxes discard user packets that do not include an authentication token, or a solution to a cryptographic puzzle. A protected destination receives packets only from specific mailboxes, which were authorized in a previous communication. The only approach utilizing a distributed intrusion detection system (IDS) is presented in DefCOM [24]. DefCOM nodes collaborate during an attack to spread alerts, and protect legitimate traffic based on local classifier, which limits attack traffic.

Most of the existing overlay solutions, that were discussed above, have been influenced by SOS [18]. These solutions rely on authentication mechanisms, requiring pre-established trust or user interaction, to filter unauthorized traffic and defend against DoS attacks. None of them except SOS [18], WebSOS [28] and [29] operate transparently to end-users, and build on existing network infrastructure. While [24] is the only solution exploiting an IDS mechanism, but assumes that such a protection mechanism already exists.

7 Conclusions and Future Work

In this paper, we proposed and implemented a distributed and transparent to end-users architecture overlay network to counter DoS attacks, that does not require modifications to the underlying network infrastructure. The proposed architecture is based on a multi-layered semistructured overlay network, which is DoS resistant by design, and uses filtering to stop DoS attacks close to their source. We believe that our work makes the use of overlays for DoS protection more feasible compared with previous work.

We evaluated MON using the Apache web server as the protected service. Results shows that it has little effect on user experience, and it can effectively detect and mitigate DoS attacks against the WWW and similar Internet services like FTP and e-mail. However, additional analysis should be done for real time services. For future extensions to MON, we are considering additional protection mechanisms that can be incorporated into our DPM to also identify and prevent other types of DoS attacks.

Acknowledgements

This work was supported by the National Science Foundation through Grant CNS-07-14277. Any opinions, findings, conclusions or recommendations expressed

herein are those of the authors, and do not necessarily reflect those of the US Government or the NSF.

References

1. Abatishchev: Low orbit ion cannon. `http://sourceforge.net/projects/loic/`
2. Abramson, N.: THE ALOHA SYSTEM: another alternative for computer communications. In: AFIPS '70 (Fall): Proceedings of the November 17-19, 1970, fall joint computer conference. pp. 281–285. ACM (1970)
3. Andersen, D.G.: Mayday: Distributed Filtering for Internet Services. In: Proceedings of the 4th Usenix Symposium on Internet Technologies and Systems. Seattle, WA (March 2003)
4. Andersen, D.G., Balakrishnan, H., Kaashoek, M.F., Morris, R.: The case for resilient overlay networks. In: Proceedings of the 8^{th} Workshop on Hot Topics in Operating Systems. pp. 152–. IEEE Computer Society (2001)
5. Beitollahi, H., Deconinck, G.: An overlay protection layer against denial-of-service attacks. In: Proceeding of the IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 1–8 (April 2008)
6. Beitollahi, H., Deconinck, G.: FOSeL: filtering by helping an overlay security layer to mitigate DoS attacks. In: Proceedings of the IEEE International Symposium on Network Computing and Applications. pp. 19–28. IEEE Computer Society (2008)
7. Chee, W.O., Brennan, T.: Slow HTTP POST DoS attacks. OWASP AppSec DC 2010, `http://www.owasp.org/images/4/43/Layer_7_DDOS.pdf` (November 2010)
8. Chellapilla, K., Simard, P.Y.: Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In: Advances in Neural Information Processing Systems (NIPS), vol. 17, pp. 265–272. MIT Press (2005)
9. Cheswick, W.R., Bellovin, S.M., Rubin, A.D.: Firewalls and Internet security: repelling the wily hacker. Addison-Wesley (2003)
10. Cretu-Ciocarlie, G.F., Stavrou, A., Locasto, M.E., Stolfo, S.J.: Adaptive anomaly detection via self-calibration and dynamic updating. In: Proceedings of the 12^{th} International Symposium on Recent Advances in Intrusion Detection (RAID). pp. 41–60 (2009)
11. Dixon, C., Anderson, T., Krishnamurthy, A.: Phalanx: withstanding multimillion-node botnets. In: Proceedings of the 5^{th} USENIX Symposium on Networked Systems Design and Implementation. pp. 45–58. NSDI'08 (2008)
12. Gil, T.M., Poletto, M.: MULTOPS: a data-structure for bandwidth attack detection. In: Proceedings of the 10^{th} USENIX Security Symposium (August 2001)
13. GNU: The GNU multiple precision arithmetic library. `http://gmplib.org/`
14. Hovemeyer, D., Pugh, W.: Finding more null pointer bugs, but not too many. In: Proceedings of the 7^{th} ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering (PASTE). pp. 9–14 (2007)
15. Ioannidis, J., Bellovin, S.M.: Implementing Pushback: Router-based defense against DDoS attacks. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (February 2002)
16. Ioannidis, S., Keromytis, A.D., Bellovin, S.M., Smith, J.M.: Implementing a distributed firewall. In: Proceedings of the 7^th ACM conference on Computer and communications security. pp. 190–199. CCS '00, ACM (2000)
17. Jula, H., Tralamazza, D., Zamfir, C., Candea, G.: Deadlock immunity: enabling systems to defend against deadlocks. In: Proceedings of the 8^{th} USENIX conference on Operating systems design and implementation (OSDI). pp. 295–308 (2008)

18. Keromytis, A.D., Misra, V., Rubenstein, D.: SOS: secure overlay services. In: Proceedings of the 2002 SIGCOMM conference. pp. 61–72 (August 2002)
19. Krasnyansky, M.: Virtual point-to-point (TUN) and ethernet (TAP) devices. `http://vtun.sourceforge.net/tun/`
20. Kurian, J., Kulkarni, A., Vu, H.T., Sarac, K.: ODON: an On-Demand security overlay for Mission-Critical applications. In: Proceedings of the International Conference on Computer Comm. and Netw. pp. 1–6. IEEE Computer Society (2009)
21. Kurian, J., Saraç, K.: Provider provisioned overlay networks and their utility in dos defense. In: Proceeding of the IEEE GLOBECOM. pp. 474–479 (2007)
22. Mirkovic, J., Dietrich, S., Dittrich, D., Reiher, P.: Internet Denial of Service: Attack and Defense Mechanisms. Prentice Hall, illustrated edition edn. (Jan 2005)
23. Mirkovic, J., Reiher, P.: D-ward: A source-end defense against flooding denial-of-service attacks. IEEE Trans. Dependable Secur. Comput. 2, 216–232 (July 2005)
24. Oikonomou, G., Mirkovic, J., Reiher, P., Robinson, M.: A framework for a collaborative ddos defense. In: Proceedings of the 22^{nd} Annual Computer Security Applications Conference. pp. 33–42. IEEE Computer Society (2006)
25. Patcha, A., Park, J.M.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. Comput. Netw. 51, 3448–3470 (August 2007)
26. S, K., K, S.: Security architecture for the internet protocol. RFC 4301 (Dec 2005)
27. Siris, V.A., Papagalou, F.: Application of anomaly detection algorithms for detecting syn flooding attacks. Comput. Commun. 29, 1433–1442 (May 2006)
28. Stavrou, A., Cook, D.L., Morein, W.G., Keromytis, A.D., Misra, V., Rubenstein, D.: WebSOS: an overlay-based system for protecting web servers from denial of service attacks. Computer Networks 48(5), 781–807 (2005)
29. Stavrou, A., Keromytis, A.D.: Countering dos attacks with stateless multipath overlays. In: Proceedings of the 12th ACM conference on Computer and communications security. pp. 249–259. CCS '05, ACM, New York, NY, USA (2005)
30. Titz, O.: Why tcp over tcp is a bad idea, `http://sites.inka.de/bigred/devel/tcp-tcp.html`
31. Viega, J., Messier, M., Chandra, P.: Network Security with OpenSSL. O'Reilly Media, 1 edn. (Jun 2002)
32. Von Ahn, L., Blum, M., Langford, J.: Telling humans and computers apart automatically. Commun. ACM 47, 56–60 (February 2004)
33. Wang, H., Zhang, D., Shin, K.G.: Change-point monitoring for the detection of dos attacks. IEEE Trans. Dependable Secur. Comput. 1, 193–208 (October 2004)
34. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A content anomaly detector resistant to mimicry attack. In: Proceedings of the 9^{th} International Symposium on Recent Advances in Intrusion Detection (RAID). pp. 226–248 (2006)
35. Wang, X., Feng, D., Lai, X., Yu, H.: Collisions for hash functions md4, md5, haval-128 and ripemd. Cryptology ePrint Archive, Report 2004/199 (2004)
36. Yaar, A., Perrig, A., Song, D.: SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In: Proceedings of the IEEE Symposium on Security and Privacy. pp. 130–143 (2004)
37. Yang, X., Wetherall, D., Anderson, T.: A DoS-limiting network architecture. In: Proceedings of the 2005 conference on applications, technologies, architectures, and protocols for computer comm. pp. 241–252 (2005)