Addressing Shortcomings of Existing DDoS Protection Software Using Software-Defined Networking

Lukas Iffländer, Stefan Geißler, Jürgen Walter,
Lukas Beierlieb, Samuel Kounev

{firstname.lastname}@uni-wuerzburg.de
Würzburg University, Würzburg, Germany

Abstract

DDoS attacks are becoming increasingly frequent and violent. A typical type of attack is the TCP SYN flood, inhibiting a server from opening new TCP connections. Current countermeasures to this attack introduce inefficiencies by either reducing computing resources on the service host or creating new network bottlenecks. In this work, we present a novel approach to mitigate TCP SYN flood attacks using software-defined networking. We perform an initial evaluation of a proof-of-concept implementation that exhibits performance measures close to existing countermeasures while circumventing their inefficiencies.

1 Introduction

Network security is essential to provide reliable services on the web. Distributed-Denial-of-Service (DDoS) attacks regularly interrupt services and cause considerable financial damage to service providers. Moreover, attacks are becoming more frequent and complex [8].

A widespread attack is the TCP SYN flood, which exploits the TCP three-way handshake. By sending countless requests with forged IPs, the TCP backlog is filled with half-open connections. Establishing new, benign connections is consequently no longer possible. The existing solutions SYN cookies and SYNPROXY offer ways to protect against these attacks. However, SYN cookies create additional load on the actual service host, and SYNPROXY requires the routing of active connections over the proxy host.

In this work, we present a solution using Software-defined Networking (SDN) and Network Function Virtualization (NFV) to circumvent these limitations. After describing our approach and its prototype implementation, we perform an initial evaluation. Preliminary results show the validity of the proposed mechanism as well as promising results regarding attack mitigation performance.

The remainder of this paper is structured as follows: In Section 2 we describe the TCP SYN flood attack and the already existing defenses. Next, Section 3 describes our approach, accompanied by the corresponding implementation in Section 4. We present preliminary results in Section 5 and provide an overview of the related work in Section 6. Finally, we conclude the paper in Section 7.

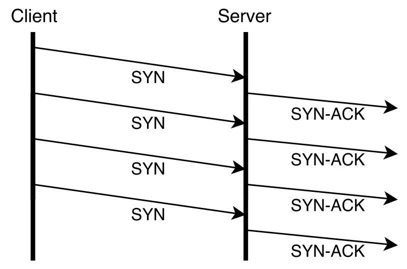

Figure 1: SYN Flood Attack Pattern.

2 SYN Floods and Existing Defenses

TCP uses a three-way handshake to establish connections. A SYN packet by the client is answered with a SYN-ACK packet by the server which replies with an ACK packet. The completion of these steps concludes the establishment of a new connection and is necessary to exchange the Initial Sequence Numbers (ISNs) for server and client.

TCP SYN Floods exploit the fact that the transmission control block (TCB) stores all the information about a connection in a data structure referred to as the backlog. However, the backlog does not only store fully established connections. Instead, half-open connections are stored as well, once the SYN packet is received. This property allows an attacker to send SYN packets with arbitrary source addresses, thus hiding his identity and avoiding connection limits, while still occupying the servers backlog, resulting in legitimate SYN packets not being able to be handled. Figure 1 illustrates this for an attacker that sends SYN packets with spoofed source addresses, to which the server's SYN-ACK responses are transmitted. The Linux kernel provides the two following well-known defense approaches.

SYN Cookies eliminate the need for half-open connections in the backlog. To this end, SYN Cookies

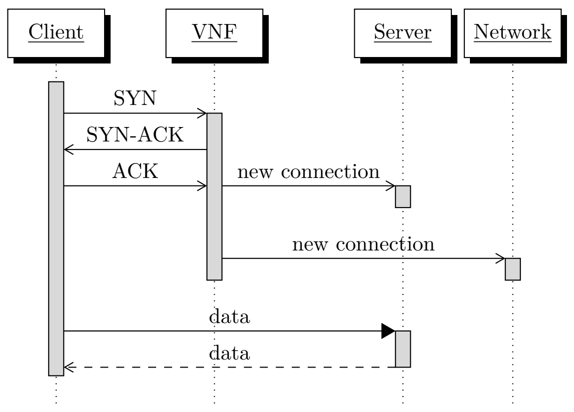

Figure 2: Connection Establishment Process.

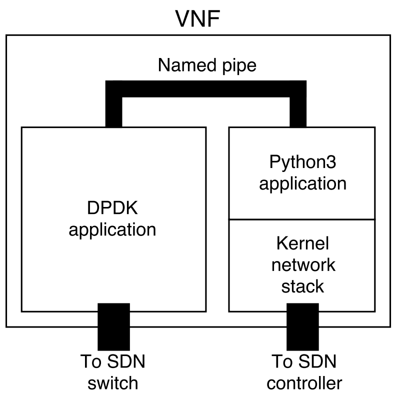

Figure 3: Architecture of VNF Protection Module.

encode the required information (client ISN, maximum segment size, ...) into the SYN-ACK packet. After receiving the ACK response, SYN cookies decode this information from the received packet. To prevent spoofing ACK responses the connection properties are hashed together with a secret number and a counter increasing with time. On reception of an ACK packet, the server calculates the hashes for the last values of the time counter. Only if the packet matches, a TCB addition to the backlog occurs. SYN cookies, therefore, prevent the backlog from overflowing but requires additional computing resources for the hashing. Thus, using SYN cookies diverts resources from the actual service running on the host.

SYNPROXY acts as a middleman between clients and a server. Clients negotiate a connection with the SYNPROXY instead of the server. If an ACK packet successfully opens a connection, the SYNPROXY negotiates a new TCP connection with the server. Since server and client are extremely unlikely (1 in 2^{32}) to have the same ISN, all further traffic has to travel via the SYNPROXY where sequence and acknowledge numbers are modified to match the numbers of the other party. The existing SYNPROXY solution has two major drawbacks. First, the detour creates a significant overhead which can make the proxy a bottleneck network-wise. Second, the approach is not stateless since the proxy has to store open connections.

3 Approach

In our approach, we improve the ideas behind SYN Cookies and SYN PROXY using Software-defined Networking (SDN) and Network Function Virtualization (NFV). SDN separates the data plane of a network from the control plane, allowing for application-specific traffic routing. NFV describes the realization of network functions as software running on top of COTS (commercial off-the-shelf) hardware, as opposed to dedicated, specialized hardware components.

Connection establishment, as usual, starts off with a client sending a SYN packet to the server. Instead of delivering it to the server, the network per default forwards the packet to our new VNF, which then replies with a SYN-ACK packet using the same methodology used in SYN cookies. This packet looks like it originates from the server. If the client then replies with a correct ACK message the VNF establishes the connection to the server. The VNF encodes the ISN for the server to use in the payload of the SYN packet. The connection establishment completes with a SYN-ACK from the server and an ACK from the VNF. Next, the network is reconfigured to no longer send traffic for the established connection to the VNF but instead to the server. Finally, the client and the server can directly exchange data. Figure 2 depicts this connection establishment process.

Using this approach has many advantages compared to existing solutions. Separation of service host and protection is possible, similar to SYNPROXY, but established connections no longer have to pass through the additional machine but are instead sent directly to the server, eliminating the proxy as a possible network bottleneck. The VNF itself is stateless. This property in combination with the separation of the service and the protective environment allows for the addition of more VNFs if required. Thus, the protection system becomes independently scalable without the need to modify the server, as would be the case with SYN cookies.

4 Implementation

In this work, we have realized a prototype of this system. Figure 3 shows the architecture of our prototype implementation. Incoming traffic is analyzed by an Intel DPDK[1] application, which also handles the connection establishment with clients and the server. This application reports newly established connections to a Python application responsible for deriving the required REST request for the SDN controller to modify the network accordingly. The request is then handed to the kernel network stack and sent to the server.

[1]`https://www.dpdk.org/`

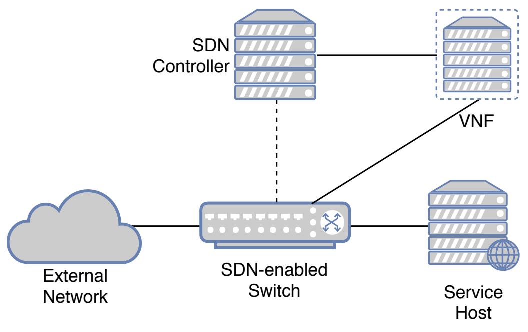

Figure 4: Test Bed Network Topology.

5 Preliminary Results

We have performed a preliminary evaluation to validate the feasibility of our approach. Figure 4 describes the test bed including an SDN-capable switch between the external network and our server, the service host. We connect our DDoS Protection VNF to this gateway switch, which is configured by a regular SDN controller.

Table 1 shows the fraction of established TCP connections for different protection settings while being unattacked and during a SYN flood attack using *hping3*. At each experiment, we try to establish 50 TCP connections with the server which was repeated 50 times. Without protection, the attack completely disables the server, blocking new connections. In contrast, our prototype allows establishing 60% connections on average. While this is still short of the 70% connections possible using SYN cookies, these are promising results for a first prototype.

6 Related Work

Related research can be divided into SDN-based DDoS protection and other security mechanisms [3, 4, 5], and NFV to improve network security [1, 2, 6].

Mousavi and St-Hilaire [4] evaluate the impact of DDoS attacks on controller resources. Their proposed protection mechanism is based on the entropy of destination addresses and can detect attacks within the first 500 arriving packets. Lim et al. [3] propose a controller application to detect DDoS attacks similar to our proposal. However, their centralized and controller application based approach puts additional, potentially malicious, load towards a crucial element of every software-based network. Wang et al. [5] evaluate the usability of SDN to improve DDoS protection in cloud environments. DaMask is an anomaly-based thread detection system that has been shown to have a detection rate of up to 89%. However, the application of DaMask is limited to cloud environments and requires changes to existing infrastructures.

Efforts such as FRESCO [1] and AvantGuard [2] aim at providing frameworks for the rapid design of security mechanisms. VFence [6] is a scalable agent-based approach to detecting and mitigating DDoS attacks in softwarized networks using TCP connection establishment spoofing, similarly to our approach. However, VFence, similar to SYNPROXY, has two limitations: i) the traffic is always transported over the VNFs and ii) the VNFs are stateful.

	no attack	SYN flood attack		
protection mode	all modes	without protection	SYN cookies	VNF (new)
connectivity	100%	0%	70%	60%

Table 1: Fraction of Established TCP Connections.

7 Conclusion

DDoS attacks are becoming increasingly frequent and violent. In this work, we presented a new approach to protect against TCP SYN flood attacks using SDN and NFV. Our SDN-enabled network temporarily forwards traffic to a security VNF that handles the TCP handshake and subsequently triggers flow reconfiguration in order to forward traffic to the actual server. We briefly describe the VNF implementation and show promising results with successful connection rates similar to existing solutions while circumventing inefficiencies of related approaches. In the future, we plan to extend our prototype into a ready-to-use solution and integrate it into our attack-aware function management system [7].

References

[1] S. W. Shin et al. “Fresco: Modular composable security services for software-defined networks”. In: *20th Annual Network & Distributed System Security Symposium*. NDSS. 2013.

[2] S. Shin et al. “Avant-guard: Scalable and vigilant switch flow management in software-defined networks”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM. 2013, pp. 413–424.

[3] S. Lim et al. “A SDN-oriented DDoS blocking scheme for botnet-based attacks”. In: *Ubiquitous and Future Networks (ICUFN), 2014 Sixth International Conf on*. IEEE. 2014, pp. 63–68.

[4] S. M. Mousavi and M. St-Hilaire. “Early detection of DDoS attacks against SDN controllers”. In: *Computing, Networking and Communications (ICNC), 2015 International Conference on*. IEEE. 2015, pp. 77–81.

[5] B. Wang et al. “DDoS attack protection in the era of cloud computing and software-defined networking”. In: *Computer Networks* 81 (2015), pp. 308–319.

[6] A. Jakaria et al. “Vfence: A defense against distributed denial of service attacks using network function virtualization”. In: *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*. Vol. 2. IEEE. 2016, pp. 431–436.

[7] L. Iffländer et al. “The Vision of Self-aware Reordering of Security Network Function Chains”. (Vision Paper). In: *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*. ICPE ’18. Berlin, Germany: ACM, 2018, pp. 1–4.

[8] A. Networks. *Insight into the Global Threat Landscape*. [Online; accessed 30. Aug. 2018]. July 2018.