Denial-of-service test-bed for distributed location proof system

Pedro Teixeira[0000−0002−6632−1275], Samih Eisa[0000−0003−0972−4171], and Miguel L. Pardal[0000−0003−2872−7300]

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
{pedro.a.f.teixeira,miguel.pardal}@tecnico.ulisboa.pt
samih.eisa@inesc-id.pt

Abstract. Location proof systems use many smart devices scattered across different geographic areas to provide witnessed proof of location to enable secure location-based services. However, Denial-of-Service (DoS) attacks can affect the system by slowing or shutting down the smart devices, making them inaccessible to its intended users. These attacks can happen even if the network devices are authenticated and are using encrypted communications. This is because DoS is a distinct class of attack that targets the availability of the system and requires different security solutions that are hard to deploy and test.
In this work we provide a Mininet-based test-bed for an example Internet of Things system. The test-bed can emulate all the network nodes across a city, including sensors, servers and routers, and can generate both regular and DoS traffic. Using it, we can experiment with DoS detection and mitigation techniques before the application is deployed. We have run experiments with simple threshold analysis and also with sophisticated techniques based on Deep Learning. The end-goal is to deliver systems that are protected and stable even during DoS attacks.

Keywords: Security · Location Proof · Denial-of-Service · Distributed Denial-of-Service · Software-Defined Networking · Deep Learning.

1 Introduction

Denial of Service (DoS) attacks can make the network resources unavailable for the intended users, thus the service is interrupted temporarily or indefinitely [2]. Distributed Denial of Service (DDoS) is a coordinated DoS attack, generated by using many compromised hosts [9]. Attacks like DDoS have been performed [4] with ease, i.e. it is easy for the attacker to take over a large number of these devices in order to execute a DDoS attack.

One possible way to improve network security is to use a Software Defined Networking [6] (SDN) architecture. It has potential to enhance network security with the provision of a highly reactive security monitoring, analysis and response system.

In this article we propose a DoS test-bed based on the Mininet [5] network virtualizer. We also present two different solutions for detecting these attacks:

one based on *Thresholds* and the other on *Deep Learning*, explaining the differences between these two approaches, the different cases that they might apply, providing experiments and results from both solutions. In terms of security properties [1], the objective of this work is to assist in the development of solutions to preserve *availability*; the focus is *not* on *integrity* and *confidentiality*.

For the test-bed evaluation, we chose CROSS [8] as an example Internet of Things application with a support network. CROSS is a distributed proof location system created with the intention of promoting smart tourism in the city of Lisbon, Portugal. In CROSS, users are rewarded if they complete itineraries across the city. If the system is made unavailable by a DoS attacks, tourists will not be able to collect their rewards and will be very unsatisfied.

Location proof systems, such as CROSS, are usually deployed with many smart devices to help clients prove their position and prevent location spoofing attacks. These devices are programmed so that they can protect the client's privacy, using methods like anonymization or cryptography. However, they lack security protections against DoS attacks. This work will address this gap with the use of SDN for testing DoS mitigations.

Hence, the key contributions of this article are as follows:

- Present a Solution for detecting and mitigating DoS/DDoS attacks that joins previous work, making each solution compensate where the other lacks.
- Describe a test-bed for DoS/DDoS attacks based on CROSS.

The remainder of the paper is structured as follows:

- Section 2 start by presenting our network management, followed by work on DoS/DDoS detection, ending with previously proposed ways to mitigate these attacks.
- Section 3 presents our proposed solution for dealing against DoS/DDoS attacks.
- Section 4 explains our test-bed in detail.

2 Background & Related work

This section describes previous work with DoS attacks providing detection and mitigation methods.

2.1 Network infrastructure management

Software-defined anything (SDx) is a technology that makes software more "in command" of multi-piece hardware systems allowing for software control of a greater range of devices. SDx includes software-defined networking (SDN), which is the most recognized technology, and its core feature is the separation of the control plane and data plane in the network. SDN realizes flexible control of network traffic and provides a good platform for the innovation of core networks and applications. The devices in SDN are programmable, and thus the networks

themselves are more dynamic, manageable, cost-effective, and adaptable. Making the SDN technology more capable of addressing problems like DoS in location proof systems.

The primary components of SDN are controllers and switches. The controllers in SDN oversee the management of the entire network, and the switches in SDN are responsible for network traffic forwarding based on the instructions deployed through the SDN controllers.

With all this in mind, we can have an idea of how we can simulate the CROSS infrastructure using SDN and how we can configure it, and also the procedures to follow.

2.2 Denial-of-Service detection

To perform DoS attack detection, Yin et al. [12] have proposed an algorithm that calculates the cosine similarity of the vectors of packets. It checks the incoming packets (`packet_in`) in each port of the boundary switches in the SD-IoT [12] and then determines whether a DoS attack has occurred based on the value of the cosine similarity. Although the results of this method were good, compared to previous methods presented by Yin et al., there is a weak point in this solution: the assumption made by the authors that the packets are similar which we cannot assume in a real-world scenario. Despite this weakness the threshold approach can however be used for certain DoS attacks as we will see in the next sections.

Software-Defined Anything and Machine Learning are two approaches that can be combined to deal with DoS attacks, as Ravi et al. [10] have demonstrated by crafting a novel Learning-driven detection mitigation mechanism (LEDEM) mechanism. LEDEM leverages the cloud-SDN architecture as it is open, flexible, programmable, and dynamic. SDN splits the control and data plane in the network. To tackle these issues, there is a necessity to provide intelligence to detect DoS. Keeping this in mind the Ravi et al. have used a semi-supervised [10] ML model, the semi-supervised deep extreme learning machine (SDELM) model, which is a mixture of unsupervised training, where unlabeled data is used, and supervised training, where labeled data is used, to detect DoS.

2.3 Denial-of-Service mitigation

The strategy to mitigate these attacks, in a SDN environment, is to set drop rules for the malicious packets as done in the work of Yin et al. [12] where, after an attack has been detected, it is set on the flow table to drop all the packets originated by the malicious devices. But we need to consider that if an attacker has a large botnet, composed for example of many IoT devices, setting individual rules for the malicious IoT will saturate the limited flow table space in switch and lead to overloading issues in the control plane of the SDN. So, Ravi et al. [10] have proposed a novel mitigation strategy that mitigates DDoS and prevent saturation issues. This strategy is the one that we followed in our own proposal, presented in the next section.

3 AntiDDoSTe

Our solution is called AntiDDoSte[1]and it uses 2 strategies for DoS/DDoS detection and 1 strategy for mitigation.

3.1 DoS detection

The first strategy is the *Threshold* detection approach in which we set a threshold that limits the number of certain packets that a client can send. The procedure applied to TCP-SYN flood attacks in shown in Algorithm 1. First, we set a predefined threshold N which can be 20, for example. Then, when a TCP packet is received in the controller, it is detected if the packet contains a SYN or a ACK flag. If the packet has a SYN flag the counter for that client is increased, if the packet has a ACK flag, the counter for that client is decreased. In the event that the counter reaches the maximum threshold the mitigation strategy is applied.

Algorithm 1: Threshold mitigation approach

```
Result: Increase/Decrease counter from host or mitigate attack
Threshold ← N;
tracker ← {};
if PacketIn.type != TCP then
    return
end
if PacketIn.flags != SYN or ACK then
    return
end
if PacketIn.flags == SYN then
    if PacketIn.scr not in tracker then
        tracker[PacketIn.scr] = 1 #src means packet source IP
    else
        tracker[PacketIn.scr] += 1
        if tracker[PacketIn.scr] == Threshold then
            Apply mitigation strategy
        end
    end
end
if PacketIn.flags == ACK then
    if PacketIn.scr not in tracker then
        tracker[PacketIn.scr] -= 1
    end
end
```

The second detection approach is based on the Deep learning [7] (DL) technology. Deep learning is a branch of Machine learning where its most important advantage is the replacement of the handcrafted features with efficient al-

[1] Neologism resulting from the combination of "antidote" with DDoS.

gorithms for unsupervised or semi-supervised feature learning and hierarchical feature extraction. This technology has been previously applied on traffic classification [11]. Therefore, we deploy a DL model alongside the controller. The controller is responsible for the extraction of packet features, which can be seen in table 1, alongside some examples and their types, and send these features to the controller. Once the model as received the features it analyse them and sends the decision to the controller. Finally, the controller upon receiving the decision applies the mitigation strategy if needed.

Field	Field Example	Field Type
Frame Number	1	Numerical
frame.len	805	Numerical
ip.protocol	tcp	Text
ip.ttl	127	Numerical
tcp.srcport	2090	Numerical
tcp.dstport	443	Numerical
tcp.syn	1	Numerical
tcp.ack	0	Numerical
tcp.rst	0	Numerical
Time	0	Numerical

Table 1. Network traffic fields.

Our novel proposes a way to work against possible weaknesses present in previous work. For example in the case of the threshold detection, it might not be possible to detect every attack since we need to know a weakness present in that attack, which might not exist, making the DL solution more appropriate for those cases. On the other hand, if we only use the DL solution, it can take a considerable time to detect a simpler attack, e.g. TCP-SYN flood, in comparison with the threshold solution, when deployed in a less powerful device. Therefore, what we are purposing is a solution that joins different technologies, in this work DL and thresholds, in order to be able to detect attacks faster than just using a single solution.

3.2 DoS mitigation

Once a device has been stated as malicious by the detection approaches the controller is responsible for applying the mitigation strategy. The controller starts by creating an entry for a VLAN[2], if the VLAN does not exist, with no flow rules so that the packets get drooped, then, the malicious device is put in that VLAN. There is only one VLAN responsible for every malicious device per sub network, and so needing only one entry in the switch flow table for dropping all the packets from the malicious devices [10].

[2] A VLAN is a Virtual Local Area Network. It is a broadcast domain that is partitioned and isolated at the data link layer.

4 Test-bed

In this section we explain our test-bed, starting by first describing the tool we use to emulate CROSS then we mention the hardware components and software components we emulate in our test-bed, ending with a brief overview of the network topology.

4.1 Network Emulator

Mininet [5] is an OpenFlow-based SDN emulator giving researchers an efficient way to test their SDN frameworks and measure their performance and reliability. The Mininet is an open source emulator written in the Python programming language. It is built over the Ubuntu Linux distribution. The elements of Mininet are organized into three main components: the *host*, which sends and receives the packets, the *switch*, which stores all the required rules to forward the packets to their destinations, and a *central controller* which handles the functionality of control and management operations in the network. Mininet supports different types of virtualized hosts, switches and controllers.

4.2 Hardware

This test-bed consists mainly of a server, 10 devices that can be Kiosks, Smart Space Managers (SSM) or Wi-Fi Access Point (AP), 6 OpenFlow-enabled switches and 6 SDN controllers. Quantities and components used in this test-bed are listed in Table 2. The current set-up only use one machine to emulate everything which is enough to demonstrate our solution management over a smaller scale of CROSS network. However, we also expose the equipment that we will use in the tourism devices for our ultimate goal that is deploying AntiDDoSte alongside CROSS.

Device	QTY	Specification
Server	1	Intel Core i7-8750H CPU at 2.20GHz
Kiosk	3	Intel Core i7-8750H CPU at 2.20GHz or Raspberry pi
AP	4	Intel Core i7-8750H CPU at 2.20GHz or ESP32
SM	3	Intel Core i7-8750H CPU at 2.20GHz or Arduino
switch	6	Intel Core i7-8750H CPU at 2.20GHz
controller	1	Intel Core i7-8750H CPU at 2.20GHz

Table 2. Hardware Specifications.

4.3 Software

On the top of the hardware infrastructure we have a Pox controller [3] which is Python-based open-source OpenFlow/SDN. POX is used for faster development

and prototyping of new network applications. The controller comes pre-installed with the Mininet virtual machine. By using them you can turn dumb openflow devices into hub, switch, load balancer, firewall devices. The POX controller allows easy way to run OpenFlow/SDN experiments. POX can be passed different parameters according to real or experimental topologies, thus allowing experiments to be run on real hardware, testbeds or in Mininet emulator. To generate the traffic from each host we use a Python library, called Scapy. It is a powerful interactive packet manipulation program that can forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies.

4.4 Network topology

As we can see from figure 1, where figure 2 is its caption, there are 6 sub-networks, across the city of Lisbon, in the emulated framework, 5 of them are tourism points with 2 devices, per sub-network, for proof location, and the last one is for the CROSS server. Each sub-network has a switch which is connected to every device in the sub-network, working as a router for that sub-network, and a Pox controller which is responsible for setting rules in the switch for forwarding or dropping packets. Note that in Figure 1 it is internet with lower-case "i" since CROSS is a private network.

4.5 Attack scenarios

Some possible attack scenario in our test-bed are:

- An attacker uses a tourism based device, i.e. the Kiosk, Smart Space Mannager and Wi-Fi Access Point, present in one sub network to perform a DoS aiming the application server.
- Perform a DoS from one tourism based device to another in the same sub network, for example in figure 1 in Jerónimos one Kiosk could attack a Wi-Fi Access Point, or in different sub networks, for example a Kiosk in Sé could attack a Kiosk in Jerónimos.
- All the tourism based device in a sub network can be used to perform a DDoS attack to the application server or a tourism based device present in other sub network.
- And it is possible to use all the tourism based devices present in the test bed to perform a DDoS attack to the application server.

5 Conclusion

In this work we have proposed a test-bed for DoS attack detection and mitigation. As an example, we used CROSS, a location proof systems, supporting a smart tourism application. Our test-bed uses SDN so we can do a deep inspection of the packets from each device in the network, as well as, being able to set

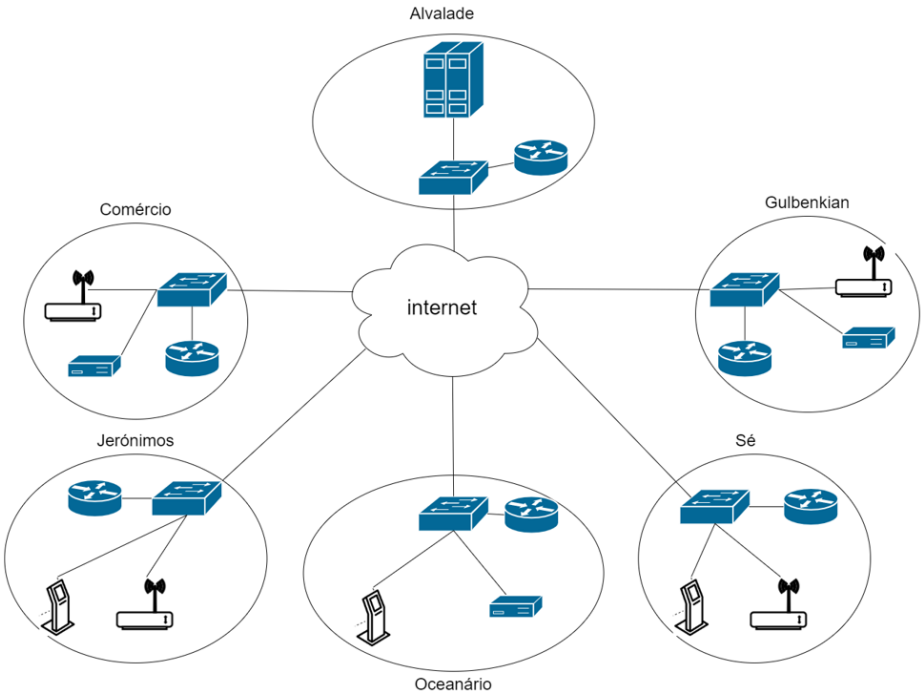

Fig. 1. Emulated CROSS topology.

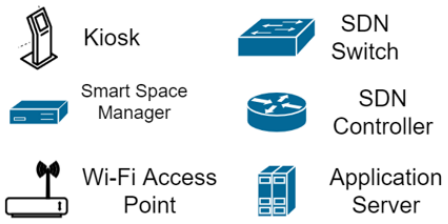

Fig. 2. Emulated CROSS topology caption.

flow rules. We also proposed two different solutions for detecting DoS attacks one based on Thresholds and another based on Deep Leaning, conducting different experiments for each approach.

For future work we aim to use our solution to demonstrate the test-bed efficiency for dealing with DoS/DDoS attacks. And, use real devices for our test-bed.

Acknowledgements

This work was supported by national funds through Fundaç ao para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID) and through project with reference PTDC/CCI-COM/31440/2017 (SureThing).

References

1. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing **1**(1), 11–33 (Jan 2004). https://doi.org/10.1109/TDSC.2004.2
2. Chen, W., Ding, D., Dong, H., Wei, G.: Distributed resilient filtering for power systems subject to denial-of-service attacks. IEEE Transactions on Systems, Man, and Cybernetics: Systems **49**(8), 1688–1697 (2019)
3. Kaur, S., Singh, J., Ghumman, N.S.: Network programmability using pox controller. In: ICCCS International Conference on Communication, Computing & Systems, IEEE. vol. 138, p. 70. sn (2014)
4. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: Ddos in the iot: Mirai and other botnets. Computer **50**(7), 80–84 (2017)
5. Lantz, B., Heller, B., McKeown, N.: A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. pp. 1–6 (2010)
6. Larry L. Peterson, C.C., Brian O'Connor, T.V., Davie, B.: Software-Defined Networks: A Systems Approach (2020)
7. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
8. Maia, G.A., Claro, R.L., Pardal, M.L.: Cross city: Wi-fi location proofs for smart tourism. In: International Conference on Ad-Hoc Networks and Wireless. pp. 241–253. Springer (2020)
9. Matta, V., Di Mauro, M., Longo, M.: DDoS attacks with randomized traffic innovation: Botnet identification challenges and strategies. IEEE Transactions on Information Forensics and Security **12**(8), 1844–1859 (2017)
10. Ravi, N., Shalinie, S.M.: Learning-driven detection and mitigation of DDoS attack in iot via sdn-cloud architecture. IEEE Internet of Things Journal **7**(4), 3559–3570 (2020)
11. Wang, Z.: The applications of deep learning on traffic identification. BlackHat USA **24**(11), 1–10 (2015)
12. Yin, D., Zhang, L., Yang, K.: A DDoS attack detection and mitigation with software-defined internet of things framework. IEEE Access **6**, 24694–24705 (2018)