A Deep ConvNet-Based Countermeasure to Mitigate Link Flooding Attacks Using Software-Defined Networks

Junchi Xing*, Jingjing Cai†, Boyang Zhou‡ and Chunming Wu*
*College of Computer Science and Technology, Zhejiang University
†Polytechnic Institute of Zhejiang University
‡Zhejiang Lab
Email: *{jcxing, wuchunming}@zju.edu.cn, {†netbeanscai, ‡zby_zju}@163.com

***Abstract*—Recently, link flooding attacks (LFA) have been observed as a serious threat for cutting off the Internet connectivity through congesting critical links. A LFA typically utilizes legitimate and low-rate flows, which makes it extremely hard to be detected and, subsequently, to be mitigated. In this paper, we present LF-Shield, that is a deep convolutional neural network (ConvNet) based countermeasure to accurately detect and efficiently mitigate LFAs using software-defined network (SDN) paradigm. LF-Shield can identify malicious bots that launch LFA flows by extracting end-hosts' traffic features and afterwards, classifying the type of end-hosts based on deep ConvNet. Then, LF-Shield mitigates LFAs without affecting legitimate end-hosts through blocking the classified malicious bots and limiting the bandwidths of inactive or newly-accessed end-hosts. A LF-Shield prototype is implemented for evaluating its performance by several experiments. The experimental results demonstrate that LF-Shield can identify malicious bots with an accuracy of 96.4% and mitigate LFAs with the 93.1% reduction in link degradation ratio, with negligible impact on legitimate end-hosts.**

***Index Terms*—Software-defined network, link flooding attack, deep convolutional neural network.**

I. Introduction

Distributed Denial of Service (DDoS) attacks pose a severe threat to the availability of network infrastructures and applications. In recent years, as a highly sophisticated class of DDoS attack, link flooding attack (LFA) has been emerged [1]–[3]. Unlike traditional DDoS attacks that flood the end targets, LFA aims at depleting the bandwidth of critical links so as to disconnect a targeted area [4]. To achieve this, the attacker employs a swarm of bots to send flooding traffic to the bots or decoy servers in the targeted area through critical links. LFA is destructive but is difficult to be detected and mitigated because the attacker typically uses legitimate, low-rate flows with real IP addresses [5], which are indistinguishable from legitimate flows by traditional defense mechanisms (e.g., firewalls or IDSes) deployed at the network perimeter [6].

This work is supported by the National Key Research and Development Program of China (2016YFB0800102, 2017YFB0803205), the Key Research and Development Program of Zhejiang Province (2017C01064, 2018C01088, 2018C03052), and the Major Scientific Project of Zhejiang Lab (2018FD0ZX01). Corresponding Author: Chunming Wu. wuchunming@zju.edu.cn)

Currently, a promising networking paradigm, software-defined networking (SDN) [7] monitors and programs the data plane at a centralized controller, offering potential to defend LFA. It is because that SDN is able to sample the statistics of all the network flows in a holistic and effective manner [8]. Moreover, SDN simplifies the policy enforcement for LFA mitigation. Existing SDN-based solutions [5], [6], [9]–[12] realize the LFA mitigation by re-routing the traffic that congests the links. Nevertheless, the major problem with these solutions is that the end-to-end latency of the legitimate flows in the re-routed traffic will be increased. The root cause is that they fail to identify LFA flows from legitimate ones due to the indistinguishability of individual LFA flows [9], [10] and therefore detour the legitimate flows as well.

Challenges. Therefore, to exploit the benefits of SDN to defend against LFAs, we face the following two challenges to overcome the problem above:

- How to accurately distinguish LFA flows from legitimate ones with a holistic visibility across the network?
- Based on the distinguishing results, how to efficiently mitigate LFA without affecting legitimate flows?

Our work. In this paper, we propose a deep convolutional neural network (ConvNet) based countermeasure to detect and mitigate LFA in SDN called LF-Shield. Firstly, the statistics of flows that congest links are periodically collected using SDN. Then, we group these flows according to their source IP addresses (end-host addresses), and turn this problem as an end-host type classification problem. To solve this problem, we extract six features of each end-host traffic using the related flow statistics, and classify the end-host type using a deep ConvNet. As for the LFA mitigation, we firstly focus on reducing the false positive rate of the end-host type classification. To approach this, we do not classify the inactive or newly-accessed end-hosts with insufficient number of flows (leading to inaccuracy in their traffic feature extraction), but adopt a temporary *max-min fair bandwidth-limiting* mechanism to throttle their bandwidths. Moreover, we use the programmability of SDN to block the classified malicious bots and make the traffic of classified legitimate end-hosts pass through the

links with congestion relieved.

We implement a prototype of LF-Shield as an application running on the RYU controller [13]. And we perform several experiments for evaluating the performance of our prototype implementation. The experimental results show that malicious bots that send flooding traffic in a LFA is able to be accurately distinguished and LFA can be efficiently mitigated without affecting legitimate end-hosts.

Contributions. This paper makes the following contributions:

- We propose a novel framework with four major modules (network information collector, link congestion monitor, end-host type classifier, and link flooding mitigator) to detect and mitigate LLA using SDN. (Section III)
- We introduce an accurate LFA flows distinguishing method based on deep ConvNet involving the following processes: grouping of flows to be distinguished, end-host traffic feature extraction, and binary classification. (Section IV.A)
- We design a *max-min fair bandwidth-limiting* mechanism to throttle the bandwidths of the inactive or newly-accessed end-hosts, which relieves the congestion of links and reduces the false positive rate of bots blocking. (Section IV.B)
- We implement a prototype of our proposed framework. Based on the prototype, the experiments demonstrates that: the accuracy for classification of malicious bots is 96.4%, the feasibility of congested link mitigation is reflected by the fact that 93.1% of the link degradation ratio is reduced, with negligible impact on legitimate end-hosts is caused. (Section V)

Organization. Section II presents background and preliminaries. Section III introduces the proposed LF-Shield in architecture. Section IV presents the design of LF-Shield in detail. Section V evaluates the performance of LF-Shield. Section VI discusses on related work. Section VII conclude the paper.

II. Background and Preliminaries

A. Link Flooding Attack

Link flooding attack (LFA) [1]–[3] refers to a new type of DDoS attack, as shown in the bottom half of Fig. 2. The attacker has a crowd of bots (or botnet) at his disposal, and seeks to attack a certain area, called the targeted area. Its objective is to cut off Internet connectivity to this area. To this end, the bots are made to send legitimate, low-rate traffic flows towards the decoy servers [1] or bots [2] in the targeted area. The traffic flows pass through the critical links that connect these servers. Hence the targeted links will be congested by the traffic flows.

The most remarkable characteristic of this type of attack is that it uses legitimate and low-rate traffic flows to achieve its devastating impact thus making the attack particularly hard to be detected and mitigated [5], [9].

B. Software-defined Networking

Software-defined Networking (SDN) [7] is a popular network paradigm consisting of a programmable controller and several belonging switches, where the controller communicates with the switches using southbound protocols (e.g., OpenFlow [14]) in order to collect network statistics in central and effective manner and program the forwarding rules for defining network policies. There are clear benefits to be gained from the SDN architecture in terms of innovation in network security [8]. And currently, some research has been carried out to improve network security based on the deployment of SDN, such as [15]–[17]. Similarly, SDN can be used to tackle the LFA [5], [6], [9]–[12], [18], [19].

C. Deep ConvNet

Deep ConvNet (deep convolutional neural network) refers to a convolutional neural network with multiple hidden layers [20]. The input layer receives an input feature set of data to be processed, and output layer generates the classification results (viz., Fig. 1). Due to the powerful capacity of feature abstraction and learning, it is hopeful to integrate the intelligence into attack identification to achieve high accuracy [21], [22]. Therefore, this technology opens a new door to identify LFA flows.

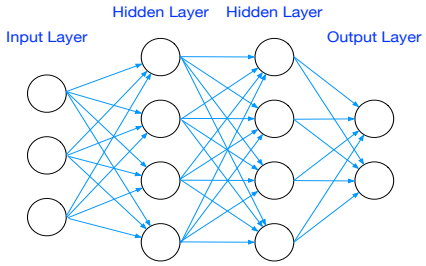

Fig. 1. A deep ConvNet architecture.

III. Architecture

In this section, we briefly introduce the architecture of the proposed LF-Shield (viz., Fig. 2). LF-Shield aims to detect and mitigate LFA flows at per-end-host granularity and is composed of four major modules:

Network Information Collector module is used for periodically collecting current flow statistics from the SDN switches through the OpenFlow [14] protocol, which is the prerequisite of LF-Shield. Subsequently, it sends the flow statistics to the next module for further detection. Moreover, it clusters the flows according to their source IP addresses (we regard each source IP address as an end-host) and stores them into a *End-host Statistics Database* as end-host traffic history information.

Link Congestion Monitor module is responsible for rapidly locating the congested links. Specifically, it accumulates the rates of flows that pass through each link based on the received flow statistics. If the accumulation reaches or exceeds the predefined bandwidth of a link, this means that the link is congested. Then this module triggers the next module.

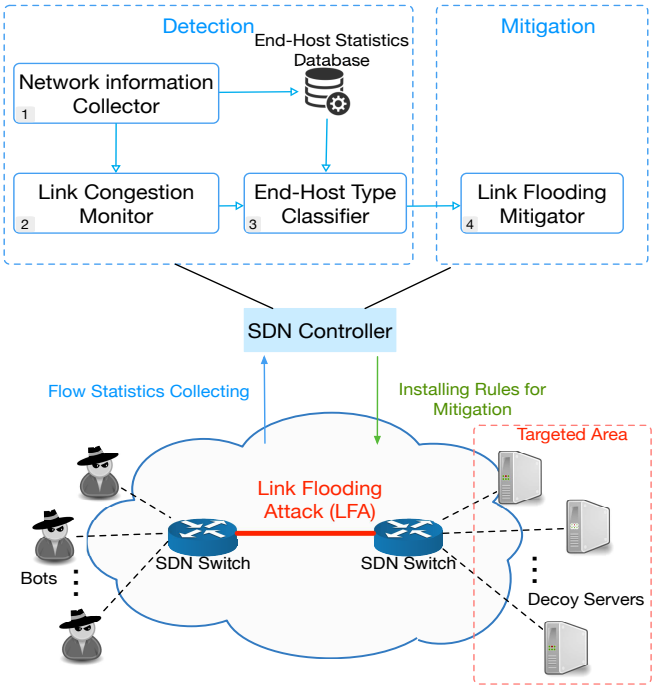

Fig. 2. Architecture of LF-Shield and an example of link flooding attacks.

End-host Type Classifier module is devoted to distinguishing the malicious bots among all the end-hosts that congest the link using deep ConvNet, which is the core module of LF-Shield. Specifically, it extracts six features of the end-hosts' traffic according to the end-host traffic history information from the End-host Statistics Database, and then performs end-host type classification using a pre-trained deep ConvNet. For accuracy, it only processes end-hosts with sufficient number of flows (active end-hosts), and the end-hosts with insufficient number of flows (inactive or newly-accessed end-hosts) will be processed by the following module.

Link Flooding Mitigator module is used for implementing the mitigation function of LF-Shield. First, it blocks all the malicious bots' traffic through installing rules into the adjacent SDN switches of the malicious bots. Second, it adopts a temporary *max-min fair bandwidth-limiting* mechanism to throttle the bandwidths of inactive or newly-accessed end-hosts through installing rules into the adjacent SDN switches of the congested link.

The first three modules and the last module will be detailly described in Section IV.A and IV.B, respectively.

IV. Design

In this section, we present the design of LF-Shield in detail. We look forward to address the challenges regarding (1) accurate flow distinguishing and (2) efficient LFA mitigation without affecting legitimate flows.

A. Flow Distinguishing Based on Deep ConvNet

Given the flows that are congesting a link, our goal is to distinguish the malicious flows from the legitimate. To this end, our idea is to classify the end-hosts to which the flows belong by leveraging the deep ConvNet. Our flow distinguishing process has three steps:

1) Grouping of Flows by End-host. The online flows statistics are collected by *Network Information Collector* module. Flows from each end-host are separated by source IP address, and the flows statistics can be grouped as end-host statistics, which will be stored in the *End-host Statistics Database*. An end-host's properties include: destination IP addresses connected, number of flows sent, duration of flows, number of packets sent, and number of bytes sent. For the accuracy of the distinguishing, we hope to only classify the end-hosts with sufficient number of flows. Thus we set a threshold to limit end-hosts with insufficient number of flows for the distinguishing. And such end-hosts will be directly processed by the LFA mitigation process until their flow numbers reach the threshold.

2) Feature Extraction. We calculate the mean value and standard deviation of the following properties of an end-host, and extract them to construct a feature set. Moreover, we analyze why they are relevant to distinguishing malicious and legitimate end-hosts.

- *Cardinality of Destination IP Address (CDIP):* This indicates the count of distinct destination IP addresses within a 10-second time window. In a LFA, each flow from a bot is low-rate, thus it need to visit numerous decoy servers for achieve its goal, which increases its cardinality [5].
- *Novelty of Destination IP Address (NDIP):* This means the change in the number of distinct destination IP addresses between time windows; new destinations might suggest that the end-host is launching a LFA because the bots often need to find more decoy servers.
- *Duration of Flow (DF):* Legitimate traffic has limited burstiness. In contrast, to flood the links, the bots should make their attack persistent [5]. Hence, each malicious flow seeks to live as long as possible.
- *Inter-flow Interval (II):* Similarly, short inter-flow interval is needed to sustain the attack.
- *Packet Rate (PR):* Bots aims at sending packet at low rate to escape statistic-based detection [1].
- *Packet Size (PS):* The distribution of packet sizes differs significantly between malicious and legitimate flows. Because the bots have the objective to escape per-packet detection and to save resources, but the legitimate end-hosts do not.

3) Binary Classification. The end-host traffic features extracted above are inputted into a deep ConvNet that will output binary labels to indicate whether an end-host is malicious or legitimate.

For time saving, the deep ConvNet is trained offline in advance. In fact, the deep ConvNet learns by measuring the loss function to quantify the prediction deviates from the actual values. The objective of the deep ConvNet is to learn in a way such that the loss function is minimized. The commonly-used *Categorical-Cross Entropy (CE)* function [22] is used as our

Algorithm 1: Max-min Fair Bandwidth Limiting

Input: Upper bandwidth limit of the congested link: B_U; Total bandwidth of all legitimate end-hosts: B_L; Bandwidths of inactive or newly-accessed end-hosts before the limiting: $b_1, b_2, ..., b_N$.

Output: Bandwidths allocated to each end-host: $b_1^{'}, b_2^{'}, ..., b_N^{'}$.

```
B_A ← B_U − B_L; /* Compute the available bandwidth.*/
for i ← 1, N do
    b_i' ← B_A / N; /* Averagely allocate the available
      bandwidth to the end-hosts initially.*/
    Allocation_is_finished_i ← False;
end
for i ← 1, N do
    if b_i' > b_i then
        Allocation_is_finished_i ← True;
        Averagely allocate b_i' − b_i to all the end-hosts
          whose Allocation_is_finished flag is False;
        b_i' ← b_i;
    end
end
```

loss function, which can be expressed as follows:

$$CE = -\sum_{i}^{\mathbb{C}} t_i log(s_i), \tag{1}$$

where t_i and s_i are the actual values and the predictive scores of deep ConvNet for each class i in $\mathbb{C}$.

Moreover, we will introduce the datasets that we use for this offline training and the training results in §V.

B. LFA Mitigation

In order to mitigate LFAs without affecting the legitimate end-hosts, we introduce two types of LFA mitigation mechanisms for different end-hosts.

1) Malicious bot Blocking. We directly block the malicious bots obtained by the deep ConvNet aforementioned with an adjustable blocking span. To achieve this, OpenFlow rules matching the source IP addresses of the malicious bots with a `DROP` instruction will be installed into the adjacent SDN switches of the malicious bots. And the blocking span lies in the lifespan of the rules.

2) Inactive or Newly-accessed End-host Bandwidth-limiting. We use a bandwidth-limiting mechanism to throttle these end-hosts with insufficient number of flows. Such bandwidth-limiting can be implemented by the *Meter Table* component of SDN switches, which is defined by the OpenFlow protocol.

However, it is necessary to determine the limited bandwidth allocated to each end-host. For this purpose, we adopt a *max-min fair bandwidth-limiting* algorithm (viz. *Algorithm 1*) [23], so as to prevent malicious bots from obtaining more bandwidth resources than other legitimate end-hosts.

Then, the rules with bandwidth-limiting instruction in *Meter Table* will be installed into the adjacent SDN switches of the congested link.

V. Evaluation

In this section, we perform several experiments to evaluate the performance of LF-Shield.

Implementation. We implement a LF-Shield prototype, including the four modules aforementioned. All of them are implemented as applications on the RYU controller [13] in Python. In particular, the deep ConvNet in the *End-host Type Classifier* module is constructed using the Keras library [24]. And we use MySQL [25] as the *End-host Statistics Database* linked to our modules. Moreover, the time scale of the *Network Information Collector* module gathering flow statistics is set to 2 seconds, and the minimum flow number threshold to determine whether an end-host will be processed by the *End-host Type Classifier* is set to 100.

Dataset. We offline train the deep ConvNet in the *End-host Type Classifier* module based on the dataset as follows. *Traffic from legitimate end-host:* The CTU-13 [26] is a well-known public benchmark dataset for botnet research. Considering that it includes no LFA flows, we only use its legitimate flows and extract more than 7,000 samples of legitimate end-host's feature set, which are labeled to negative. *Traffic from malicious bot launching LFA:* Because LFA has no public data set until now [9], we simulate LFA based on paper [1] to build this dataset. Specifically, we aim to flood targeted links whose bandwidth increasing from 10 Mbps to 500 Mbps with a step of 10 Mbps. And we use bots to send flows (at a 4 Kbps per-flow rate) to a varying number of decoy servers (increasing from 20 to 100 with a step of 2). As a result, 1,960 samples of malicious bot's feature set are extracted, which are labeled to positive.

Experiment setup. For our experiments, we run our prototype and experiments on a computer equipped with Intel 4-core Xeon E5-2609 v2 2.5GHz processor and 16GB of memory.

A. Accuracy: Malicious Bot Classification

To evaluate the accuracy of our prototype, we split up the dataset into training set and test set according to the proportion of 90%/10%, 80%/20%, and 70%/30%, feed the the deep ConvNet with the training sets, and use the test sets to examine it. The evaluation metrics mainly focus on (a) *end-host classification rate* to indicate the ratio of the end-hosts that are correctly classified and (b) *false-positive rate* meaning the fraction of legitimate end-hosts classified to be malicious among the total amount of legitimate end-hosts. In addition, we simultaneously measure the effect of each feature in the feature set by feeding the deep ConvNet with a certain feature discarded.

Fig. 3 reports the accuracy of our prototype. When 90% data are used for training with the whole feature set, the end-host classification rate can reach 96.4% while the false-positive rate drops under 1.8%. Moreover, we can get that the accuracy will be reduced when training without the *Cardinality*

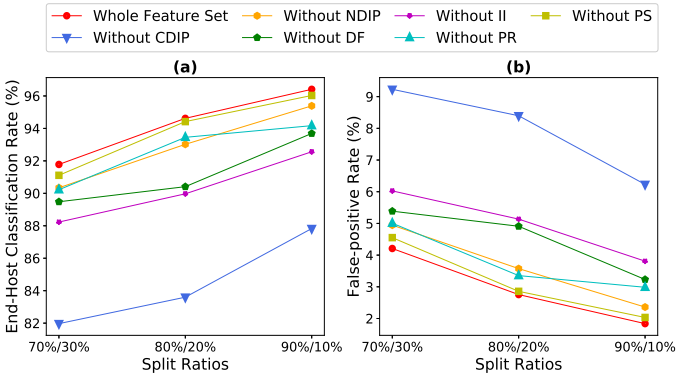

Fig. 3. Accuracy measure of LF-Shield including (a) end-host classification rate and (b) false-positive rate.

of Destination IP Address (CDIP) feature. Thus *CDIP* seems to be the dominated feature that influences the accuracy.

B. Feasibility: Relieving Congestion on Congested Link

LFA congests a link and consequently cuts the legitimate flows passing through it. We use *link degradation ratio* [1] as a metric, which is the fraction of legitimate flows cut by LFA over the number of all legitimate flows passing through the link, to indicate the effect of LFA. To validate the feasibility of LF-Shield in defending LFA, we aim to show that LF-Shield can effectively reduce the *link degradation ratio* of a congested link. To this end, we use Mininet [27] to generate a software-based network with OpenFlow switches (Open vSwitch kernel switches) and hosts, which is under control of a RYU controller with our LF-Shield prototype. Then, we use the different number of hosts as bots and decoy servers to perform LFAs on a link with bandwidth uniformly distributed in [100, 500] Mbps. Also, we use 50 hosts as legitimate end-hosts sending flows according to the *Traffic from legitimate end-host* dataset aforementioned. The flows sent by the hosts are generated by the Scapy [28] tool. Moreover, the bots and decoy severs are deployed like Fig. 2, and the legitimate end-hosts are deployed at the bots side.

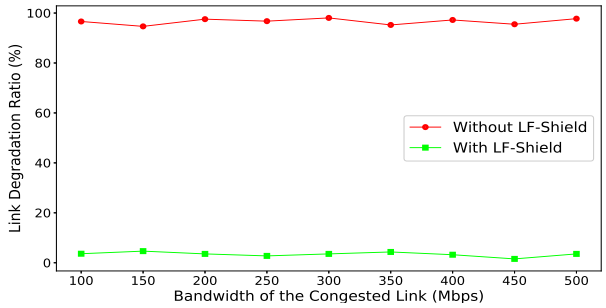

Fig. 4. Link degradation ratio of a congested link with and without LF-Shield.

Fig. 4 reports the feasibility of our prototype. It illustrates the *link degradation ratio* of a congested link without using any defense technology and using our prototype. We can get that the *link degradation ratio* is averagely reduced by 93.1% after using our prototype.

C. Impact on Legitimate End-hosts: Round-trip Time of Request

To evaluate the impact introduced by LF-Shield on legitimate end-hosts, we use the same experiment environment as before but with a targeted link whose bandwidth is 200 Mbps. And we investigate the change in *round-trip time (RTT)* of the requests during the mitigation. To this end, we employ two hosts to be a web server and a client using the webpy library [29]. And this client sends a http request to the server once per second, through a link flooded by bots. For each request, we record the timestamps when it was sent (T_{start}) and when the corresponding response was received (T_{end}), and *RTT*=$T_{end} - T_{start}$.

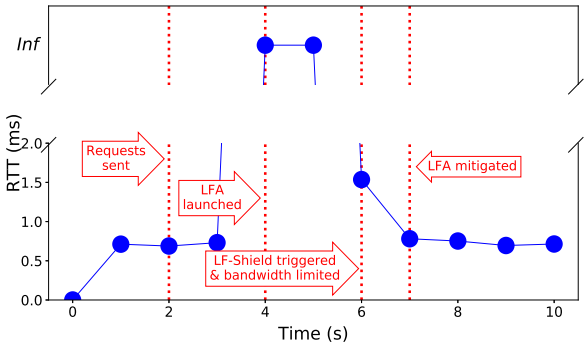

Fig. 5. RTT of requests sent by a legitimate end-host every 1 seconds, passing a link under LFA, with LF-Shield.

Fig. 5 reports the *RTT* of the requests. From $t = 1s$ to $t = 3s$, we can observe that the *RTT* stays around 0.7 ms, which can be regarded as the baseline of the normal *RTT*. At $t = 4s$, the bots begin launching LFA, thus the requests can not be responded, reflected by the infinite *RTT* in this figure. At $t = 6s$, the *RTT* falls back to an acceptable range and is slightly more than the normal. It is because that our prototype has detected the attack and blocked the bots, and starts to limit the bandwidth of the client. From $t = 7s$, the *RTT* backs to normal. It is because that the legitimate end-host has reached a sufficient number of flows and is classified as legitimate by our prototype, thus without any bandwidth limiting. In summary, LF-Shield incurs a negligible impact on the client during its mitigation.

VI. Related Work

In this section, we discuss existing works on detecting and preventing LFAs that can benefit from SDN, which can be generally classified into reactive and proactive approaches.

Reactive approaches. Woodpecker [9], [10] presents a scheme to defense against LFA based on the incrementally deployed SDN and uses the centralized traffic engineering to make the traffic balanced and eliminate the congested links. SPIFFY [5] presents a new traffic engineering technique based

on SDN whereby one can virtually increase the bandwidth by routing around the congested links. A SDN-based moving target defense mechanism proposed in [11] leverages traffic engineering dynamically to reroute traffic on the suspected target links as long as it is congested. LFADefender [6] presents a multiple optional paths rerouting method to temporarily mitigate links congestion caused by LFAs. The approach introduced in [12] uses online traffic engineering both to detect bots and to balance the load. Unlike the above works, our approach directly detects the traffic congesting the links and accordingly mitigates the attacks, instead of re-routing the traffic on the link which might incur latency to legitimate flows.

Proactive approaches. HoneyNet [18] uses SDN to create a virtual network topology and exposes the fake topology to attackers, so as to make it difficult to locate the target links to flood. Linkbait [19] is an active link obfuscation mechanism by rerouting probing flows to obfuscate links, so that target links are hidden from adversaries and some bait links are misjudged as target links by adversaries in a SDN. These approaches can be orthogonal to our approach in LFA defending.

VII. Conclusion

Link flooding attacks targeting at the critical links can cause significant harm like link congestion and targeted network area disconnection. In this paper, we presented LF-Shield, which is a deep ConvNet-based countermeasure to detect and mitigate LFAs in SDN. LF-Shield consists of four main modules, that are used for collecting flow statistics, detecting congested links, distinguishing malicious bots based on deep ConvNet, and mitigating LFA, respectively. We implemented a LF-Shield prototype, and thoroughly evaluated its performance. Our experiment results demonstrated that LF-Shield can detect LFAs with a high accuracy and mitigate LFAs with negligible impact on legitimate end-hosts.

References

[1] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, 2013, pp. 127–141. [Online]. Available: https://doi.org/10.1109/SP.2013.19

[2] A. Studer and A. Perrig, "The coremelt attack," in *2009, 14th European Symposium on Research in Computer Security*, 2009, pp. 37–52. [Online]. Available: https://doi.org/10.1007/978-3-642-04444-1_3

[3] P. Bright. Can a ddos break the internet? sure just not all of it. [Online]. Available: https://arstechnica.com/information-technology/2013/04/can-a-ddos-break-the-internet-sure-just-not-all-of-it/

[4] C. Liaskos, V. Kotronis, and X. A. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *35th Annual IEEE International Conference on Computer Communications, INFOCOM*, 2016, pp. 1–9. [Online]. Available: https://doi.org/10.1109/INFOCOM.2016.7524507

[5] M. S. Kang, V. D. Gligor, and V. Sekar, "SPIFFY: inducing cost-detectability tradeoffs for persistent link-flooding attacks," in *23rd Annual Network and Distributed System Security Symposium, NDSS*, 2016. [Online]. Available: https://www.comp.nus.edu.sg/ kangms/papers/spiffy.pdf

[6] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, and F. Yu, "Detecting and mitigating target link-flooding attacks using sdn," *IEEE Transactions on Dependable and Secure Computing*, 2018. [Online]. Available: doi.ieeecomputersociety.org/10.1109/TDSC.2018.2822275

[7] D. Kreutz, F. M. V. Ramos, P. J. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015. [Online]. Available: https://doi.org/10.1109/JPROC.2014.2371999

[8] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 623–654, 2016. [Online]. Available: https://doi.org/10.1109/COMST.2015.2453114

[9] L. Wang, Q. Li, Y. Jiang, X. Jia, and J. Wu, "Woodpecker: Detecting and mitigating link-flooding attacks via SDN," *Computer Networks*, vol. 147, pp. 1–13, 2018. [Online]. Available: https://doi.org/10.1016/j.comnet.2018.09.021

[10] L. Wang, Q. Li, Y. Jiang, and J. Wu, "Towards mitigating link flooding attack via incremental SDN deployment," in *IEEE Symposium on Computers and Communication, ISCC 2016, Messina, Italy, June 27-30, 2016*, 2016, pp. 397–402. [Online]. Available: https://doi.org/10.1109/ISCC.2016.7543772

[11] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using sdn-based moving target defense," in *41st IEEE Conference on Local Computer Networks, LCN 2016, Dubai, United Arab Emirates, November 7-10, 2016*, 2016, pp. 627–630. [Online]. Available: https://doi.org/10.1109/LCN.2016.108

[12] D. Gkounis, V. Kotronis, and X. A. Dimitropoulos, "Towards defeating the crossfire attack using SDN," *CoRR*, vol. abs/1412.2013, 2014. [Online]. Available: http://arxiv.org/abs/1412.2013

[13] "Ryu." [Online]. Available: https://osrg.github.io/ryu/

[14] "Openflow switch specification." [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[15] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic ddos defense," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, 2015, pp. 817–832. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/fayaz

[16] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: detecting security attacks in software-defined networks," in *22nd Annual Network and Distributed System Security Symposium, NDSS*, 2015. [Online]. Available: https://www.ndss-symposium.org/ndss2015/sphinx-detecting-security-attacks-software-defined-networks

[17] S. Gao, Z. Peng, B. Xiao, A. Hu, and K. Ren, "Flooddefender: Protecting data and control plane resources under sdn-aimed dos attacks," in *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017*, 2017, pp. 1–9. [Online]. Available: https://doi.org/10.1109/INFOCOM.2017.8057009

[18] J. Kim and S. Shin, "Software-defined honeynet: Towards mitigating link flooding attacks," in *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN Workshops 2017, Denver, CO, USA, June 26-29, 2017*, 2017, pp. 99–100. [Online]. Available: https://doi.org/10.1109/DSN-W.2017.10

[19] W. Qian, X. Feng, Z. Man, Z. Wang, L. Qi, and Z. Li, "Linkbait: Active link obfuscation to thwart link-flooding attacks," *CoRR*, vol. abs/1703.09521, 2017. [Online]. Available: https://arxiv.org/abs/1703.09521

[20] M. A. Nielsen. Neural networks and deep learning. [Online]. Available: http://neuralnetworksanddeeplearning.com/

[21] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *CoRR*, vol. abs/1901.03407, 2019. [Online]. Available: http://arxiv.org/abs/1901.03407

[22] Z. Chen, W. Zhang, Z. Xie, X. Xu, and D. Chen, "Recurrent neural networks for automatic replay spoofing attack detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 2052–2056. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8462644

[23] "Rate adaptation, congestion control and fairness: A tutorial." [Online]. Available: http://ica1www.epfl.ch/PS_files/LEB3132.pdf

[24] F. Chollet. Keras. [Online]. Available: https://github.com/fchollet/keras/

[25] Mysql. [Online]. Available: https://www.mysql.com/

[26] Ctu-13-dataset. [Online]. Available: http://neuralnetworksanddeeplearning.com/

[27] Mininet. [Online]. Available: http://mininet.org/

[28] "Scapy." [Online]. Available: https://scapy.net/

[29] webpy. [Online]. Available: https://github.com/webpy/webpy/