# S-Pot: A Smart Honeypot Framework with Dynamic Rule Configuration for SDN

Javier Franco, Ahmet Aris, Leonardo Babun, and A. Selcuk Uluagac
Cyber-Physical Systems Security Lab., Florida International University, Florida, USA
{jfran243, aaris, lbabu002, suluagac}@fiu.edu

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10000682

***Abstract*—Enterprise networks are becoming increasingly heterogeneous where enterprise devices and IoT devices coexist, requiring tools for effective management and security. Software-Defined Networking (SDN) has emerged in response to such needs of modern networks. SDN lacks adequate security features and Intrusion Detection and Protection Systems (IDPS) have been used to protect SDN from attacks. However, they have limited knowledge of zero day attacks. Machine Learning (ML) has become a valuable tool against these limitations and improve (SDN) network security. However, the solutions that solely rely on ML can struggle to discriminate benign traffic from malicious, and suffer from false negatives. To solve these problems and improve security of SDN-based enterprise networks, we propose S-Pot, an open-source smart honeypot framework. S-Pot uses enterprise and IoT honeypots to attract attackers, learns from attacks via ML classifiers, and dynamically configures the rules of SDN. Since honeypots generally receive only malicious traffic, S-Pot can learn from the received malicious traffic and minimize the false positives of an SDN network. In addition, S-Pot can detect the new attacks using ML classifiers, thus can help to minimize the false negatives. Our performance evaluation of S-Pot in detecting attacks using various ML classifiers show that it can detect attacks with 97% accuracy using J48 algorithm. In addition, we evaluated the effectiveness of S-Pot in improving the security of an enterprise SDN testbed network. Our results demonstrate that, compared to the without S-Pot case, S-Pot can improve the security of the SDN networks by detecting attacks with better performance, greater accuracy, effectively generating rules, and dynamically configuring the network.**

***Index Terms*—SDN, Honeypot, IDPS, IoT, Network Security**

## I. Introduction

The digital transformation has been converting all aspects of life in recent years. The ever-growing number of Internet of Things (IoT) devices has exacerbated demand on traditional networks, making it increasingly complex to manage and scale. Enterprise networks in this regard are becoming increasingly heterogeneous where enterprise devices/services and IoT devices coexist. SDN emerged in response to these needs, transforming networking infrastructure, moving the brain from network devices to a centralized software controller [1]. SDN has been playing a crucial role in providing high performance networking around the globe during the Covid-19 pandemic [2]. More and more enterprise networks are moving to SDN [2], and the market size of SDN is expected to reach $59 billion by 2023 [3]. Simultaneously, cyber threats are evolving and increasing in quantity and impact. The cost of cybercrime is expected to reach $10.5 trillion in 2025 [4].



Although SDN has its benefits to ease security operations (e.g., isolation, segmentation, attack mitigation, etc.) in enterprise networks, it can also pose new threat vectors [2]. Intrusion Detection and Protection Systems (IDPS) have been widely used to protect SDN from attacks. However, IDPS solutions have fundamentally limited signature rules, which can leave SDN-powered enterprise networks vulnerable to new threats (zero day attacks). While some IDPSs offer anomaly-based detection, they can have false positives (FP) and false negatives (FN). Machine Learning (ML) has become a valuable tool to overcome these limitations. However, ML-based solutions can also struggle to discriminate benign traffic from malicious traffic, and suffer from FN. In addition, although ML models can be retrained to cope with zero-day attacks, it may not be possible for every enterprise to have the necessary human resources that have the technical know-how to perform retraining. In parallel with IDPS and ML-based solutions, honeypots have been widely used to understand evolving cyber threats and develop effective defenses. Honeypots are mostly used for research purposes and they aim to lure attackers. In addition, they generally receive only malicious traffic [5], [6]. Although there exist various studies on the use of IDPS, ML, and honeypots with research purposes, including various combinations of these tools implemented together in SDN [7]–[14], to the best of our knowledge, *no study considered to benefit from honeypots for production purposes in improving the security of an SDN-based network.*

In this study, we propose S-Pot, an open-source smart honeypot framework that integrates the use of IDPS and ML for securing SDN-based enterprise networks through dynamic rules configuration. S-Pot benefits from honeypots that can simulate both enterprise services and IoT devices in gathering attack information. It employs an IDPS and ML classifiers to detect and learn from the attacks in honeypots. *Unlike research honeypots, it utilizes the obtained attack information from the honeypots for production purposes for the security of SDN networks, dynamically creates new rules for the attacks, and shares them with the SDN-based enterprise network.* Since honeypots (hence S-Pot) generally only receive malicious traffic, S-Pot can benefit from the identified malicious traffic of honeypots and greatly reduce FPs on the real network [5]. Moreover, it can also detect new attacks that target the enterprise network by means of ML classifiers, and thus improve FP performance of the defense solutions in the enterprise network.

We implemented S-Pot and evaluated its performance in detecting attacks using various ML classifiers. Our evaluations show that S-Pot can detect attacks with 97% accuracy with J48 algorithm. In addition, we did another analysis in evaluating the performance of S-Pot in generating new attack signatures and dynamically configuring the rules of a realistic enterprise SDN testbed network. Our analysis demonstrates that, S-Pot can efficiently generate new rules and dynamically configure rules of the realistic testbed network to block attacks. Our evaluations show that S-Pot can improve the security of an enterprise SDN network compared to the case without S-Pot.

**Contributions:** The contributions of S-Pot are as follows:

- We propose a fully open-source smart honeypot framework that benefits from enterprise and IoT honeypots to dynamically generate new IDPS rules for securing SDN-based hybrid enterprise networks.
- With S-Pot, we demonstrate how enterprises can benefit from honeypots for production purposes to secure their networks against both known and zero-day attacks.

**Organization:** This paper is organized as follows. Section II identifies the related work. Section III provides background information on honeypots, SDN, and IDPS. Section IV defines the problem scope and the threat model. Section V describes S-Pot. Section VI details the implementation of S-Pot, data collection and processing, and evaluates the performance of it. Finally, Section VII concludes the paper.

## II. Related Work

Several studies exist in the literature on securing SDN networks with the use of honeypots, IDPS, or ML tools. The use of SDN is presented by Wang and Wu [7] as a necessity for improving honeynet topology. Sultana et al. [14] surveyed ML methods using SDN and IDS. Valdovinos et al. [15] and Swami et al. [16] focused on SDN vulnerabilities and DDoS attack detection and mitigation for SDN, yet in this case, both ML and honeypot tools are not included. [17], [18], and [19] focused on research with ML and SDN. Du and Wang [8] aimed to be the first study to focus on DDoS attacks on honeypots-based SDN, particularly in Industrial IoT. Molina et al. [20] presented a high interaction IoT honeynet using SDN to protect the devices from DDoS botnet attacks. Kyung et al. [9] presented an SDN-based honeynet architecture to improve detection of fingerprinting attacks and avoid malware propagation with the application of SDN controller. Azab et al. [10] introduced a smart gateway that isolates the northbound and southbound interfaces, and works as an IDS to protect the SDN controller from being compromised. In addition to the mentioned studies, various studies exist that use IDPSs in SDN. Sarica and Angin [11] introduced an SDN-dataset for intrusion detection in IoT. Other studies focused on attack detection in SDN using ML, such as Nanda et al. [12] and Elsayed et al. [13], but do not use IDPS or provide dynamic rules configuration.

**Differences from the existing work:** Despite the wealth of research in SDN security, to the best of our knowledge, *none of the aforementioned studies have benefited from honeypots for production purposes for the security of SDN-based enterprise networks*. Also, unlike prior work, S-Pot integrates the use of honeypots, IDPS, and ML for securing an SDN network. In addition, S-Pot can detect new attacks targeting SDN-based networks, create new rules for the detected attacks, and dynamically configure the SDN-based network.

## III. Background

### A. Software-Defined Networking

SDN allows for creating a more reliable, secure, and flexible network by adding the capacity of a centralized network controller to program and manage the network [21]. SDN reduces the traditional network's limitations by separating the network into three layers: application, control, and infrastructure layers. These are also referred to as application, control, and data planes. The application layer contains the network applications, such as IDPS and firewalls. The SDN controller software is the control layer, which handles the flow of all traffic in the network and enforces policies that can be established by the network administrators. The physical switches in the network compose the infrastructure layer. Communication between the layers is carried out through northbound and southbound Application Programming Interfaces (APIs). Northbound APIs enable communication between the application layer and control layer, whereas Southbound APIs enable communication between the control layer and the infrastructure layer. Although various protocols can be used with SDN to communicate between the controller and the network devices in the infrastructure layer, Open Flow Protocol (OFP) is the most widely used [10]. SDN provides capabilities such as traffic analysis, dynamic rules updating, a global view of the network, and logically centralized network control [18]. While SDN provides control over the network that can be combined with other tools to implement strong security mechanisms, on its own, SDN lacks adequate security features making it vulnerable [22]. Varadharajan et al. [23] categorize the threats in SDN, pointing to how the controller and networking devices such as switches, can be affected by multiple attacks, including denial-of-service (DoS) and distributed DoS (DDoS) attacks.

### B. Honeypots and Honeynets

A honeypot is a decoy that is used to lure attackers and deceive them into thinking they have accessed a real system and to observe and learn from their actions by gathering data about their interaction with the honeypot [6]. The gathered data can be used to develop countermeasures against attacks [24]. Honeypots can vary greatly in the level of interaction that they allow the attacker, ranging from low interaction honeypots that emulate particular services, to high interaction honeypots that emulate entire operating systems (OS). They can be used in a wide variety of applications, for research or production purposes, and can be implemented with physical or virtual resources. Two or more honeypots implemented on a system form a honeynet [6]. Since there are generally no licit causes for which to interact with a honeypot, any traffic is usually

malicious, which allows honeypots to greatly reduce the amount of FP in comparison with anomaly-based IDSs [5].

### C. Intrusion Detection and Protection Systems

Intrusion Detection and Intrusion Protection Systems identify potential threats in a network using signature-based and/or anomaly-based methods. For signature-based detection, the systems capture traffic in a network and compare the packets to a database of known threats. For anomaly-based detection, the systems identify deviations from normal traffic in the network. An IDS detects and alerts of malicious traffic in a network but does not take any action. On the other hand, an IPS not only detects but also executes an action such as restricting access to attackers based on a set of rules. Snort IDPS [25], maintained by Cisco Systems, is one of the most widely used open-source IDPS that can be used with diverse OSs.

## IV. Problem Scope and Threat Model

### A. Problem Scope

We consider an SDN enterprise network containing both enterprise systems and IoT devices for smart buildings. The network has a signature-based IDPS that can detect the known attacks but fail to detect new attacks. To detect new attacks, the network can employ an ML-based anomaly detection system. However, although it may improve the security, it can struggle to discriminate benign traffic from the malicious, thus suffer from FP. In addition, although retraining the ML model is possible, not every enterprise may have the necessary human resources that have the technical know-how to perform this task. For these reasons, despite the security measures, the network can be vulnerable to zero-day attacks that can go undetected, and also suffer from the blocked benign traffic. To improve security of the SDN enterprise network, in this work, we propose S-Pot that benefits from enterprise and IoT honeypots, IDPS, and ML classifiers in detecting attacks targeting the network, and improves security of the network by generating new attack rules and dynamically configuring the rules. S-Pot is deployed in a demilitarized zone (DMZ) under the same domain as the real network. It publicly exposes honeypots to the Internet and aims to protect the real SDN network. S-Pot identifies key features from the gathered attack data and creates a new rule in the IDPS in the S-Pot framework that instructs the controller of S-Pot to drop a packet if it exhibits the same features. This rule is passed over to the IDPS on the real enterprise network, which dynamically instructs the controller on the real network to do the same. This improves the security of the real SDN enterprise network by updating the controller rules to defend against zero-day attacks. When an attacker attempts to send a similar attack to a device in the real network, the controller drops the packet, protecting the real SDN network from the attack.

### B. Threat Model

This work considers DoS, DDoS, scanning, trojan malware, and zero-day attacks targeting SDN-based enterprise networks. DoS, DDoS, and malware attacks have been identified amongst the most common types of cyber attacks [26], while scanning is used in the reconnaissance phase of the attacks. In addition, the protocols targeted by these attacks are commonly used in enterprise and IoT environments and have been identified among the most targeted protocols for attacks in the dark web [27].

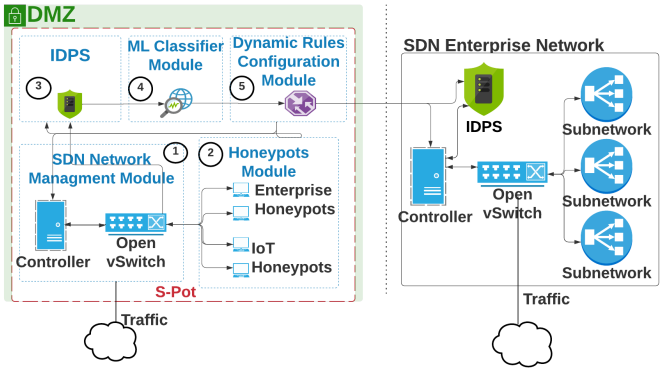


Fig. 1: Architecture of S-Pot.

## V. S-Pot Framework

### A. Overview

Figure 1 presents a general overview of the proposed S-Pot framework. S-Pot consists of the following components: ① The *SDN Network Management module* that includes of a virtual switch for connecting devices to the network and a controller for centralized network monitoring, managing the flow of all traffic in the network. ② The *Honeypot module* is comprised of virtual honeypots to lure attackers. ③ The *IDPS module* captures and logs data, detects potential threats, and distributed new rules. ④ The *ML Classifier module* is where data preprocessing, feature extraction, and classification are carried out. It also feeds the output to the Dynamic Rules Configuration module. ⑤ Finally, in the *Dynamic Rules Configuration module* the new rules are created, distributed, and integrated into the S-Pot network and the real SDN network. This is where the IDPS and controller rules are dynamically changed with the acquired knowledge from S-Pot. S-Pot utilizes SDN to better manage its components. While S-Pot could be implemented without an SDN platform, SDN provides greater features than the use of an IDPS on its own, to be able to dynamically implement new rules in a network based on the learned information. The modules of S-Pot are explained in detail in the following subsection.

### B. S-Pot Modules

*1) SDN Network Management Module:* The controller is the brain of the network. The Open Daylight (ODL) controller Oxygen version [28] was chosen for the proposed framework. The ODL controller offers open-source flexibility, as well as open protocols, centralized network monitoring, and programmable control actions for S-Pot. At the application layer, ODL allows REST API calls to the controller to push down rules to the infrastructure layer. The Open vSwitch (OVS) [29] connects all the devices to the network. S-Pot is able to mirror all honeypot traffic to the IDPS using the OVS.

*2) Honeypots Module:* The production honeypots are the tools used to attract attackers to our network. S-Pot is composed of virtual machines (VM) running on Virtual Box. This framework allows for building high interaction virtual honeypots that can simulate both enterprise services and IoT devices with open-source and closed-source OSs and provides easy scalability.

*3) IDPS Module:* All traffic goes through the IDPS module. The IDPS captures the data packets, normalizes them, and checks the packets against its ruleset database. The processed information and results are sent to output plugins of the IDPS, where options for output can be selected.

*4) ML Classifier Module:* The ML Classifier module obtains the network traffic log of the IDPS module. The first step is *Data Cleaning*. Here, the data is cleaned before proceeding with feature selection. The next step is *Feature Selection*, where selected features are combined to identify the type of attacks. Finally, pre-trained multi-class ML *classifiers* examine the unknown traffic and detect the attacks. For S-Pot, the following classification algorithms were selected: Random Forest, SMO, BayesNet, and J48. These algorithms were selected to include a variety of approaches: BayesNet is probability based, J48 is decision tree based, Random-Forest combines multiple decision trees, and SMO is used in support vector machine (SVM) implementation. We would like to highlight that although we employ classical ML algorithms in this study, S-Pot does not have any limitation to employ more advanced ML algorithms such as deep learning. The ML Classifier outputs results to the Dynamic Rules Configuration Module.

*5) Dynamic Rules Configuration Module:* This module gathers the alert log data from the ML classifications from the S-Pot framework, parses out the data such as source IP, destination IP, Ethernet type, protocol, source and destination ports, type of service, etc., and generates IDPS and SDN rules using these features, which are passed to the SDN flow tables of both S-Pot and the real SDN network. The new rules are also passed to the IDPS on both the S-Pot network and the real SDN network. In the case of new attacks classified as unknown by the classifier, in addition to generating new rules accordingly, the module sends a notification with the log data to security administrators, so that they may review the logs. The security administrators can use this information to develop new rules and manually change the IDPS and SDN rules, which can be imperative for defense against zero day attacks.

## VI. Performance Evaluation

In this section, we explain the implementation and performance evaluation of the S-Pot framework.

### A. Implementation of S-Pot

**SDN Network Management Module Implementation:** Open Daylight controller and Open vSwitch are selected for the SDN Network Management module. The ODL controller manages the flow tables in the OVS, if a packet matches a rule in the OVS flow table, then the rule is applied (e.g. drop packets). Otherwise, the OVS checks with the controller for new rules before forwarding the packets. The OVS interconnects all the modules of our framework. All traffic associated with the honeypots is compared against the OVS rules tables to check if they comply with previously recorded rules and is also forwarded to the IDPS module to be processed and logged. If attack traffic is detected in the IDPS module, a request for a new rule is sent to the Dynamic Rules Configuration module to generate a new rule for the ODL controller via the RESTCONF API, and the OVS flow table is updated to apply the new rule (e.g. drop packets from source).

```
drop tcp any any -> 192.168.100.4 80 (flags: S; msg:"Flood Attack
Detected.";flow:to_server, established; classtype: attempted-dos;
detection_filter: track by_dst, count 18 , seconds 1 ; sid:
10000007; rev:1;)
```

Fig. 2: An example Snort rule to block packets of SYN flood.

**Honeypots Module Implementation:** We employ two enterprise honeypots where each is implemented in a VM with different OSs and services to attract a greater variety of attack types. Debian 10 was selected due to its common use in enterprise servers, and Windows 10 was selected due to their wide use as clients in enterprise environments. Two IoT honeypots are also implemented. IoT Candyjar [30] and Thingpot [31] open-source honeypots were selected because they were both created for IoT, provide full-device emulation, and are scalable. Furthermore, ThingPot was created specifically for DDoS attacks for IoT, and IoT Candyjar applies machine learning to automatically learn the behaviors of IoT devices from the Internet. Telnet, Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), SSH, HTTP(S), FTP, and SMB protocols were selected because they are commonly used in enterprise and IoT environments and have been identified among the most targeted protocols for attacks in the darkweb [27].

**IDPS Module Implementation:** For the IDPS module, we selected Snort [25] because it is one of the most widely used open-source IDPSs. We used ODL controller and the OVS to connect the honeypots and forward all the traffic to the Snort IDPS. Figure 2 demonstrates an example of a Snort IDPS rule. This rule blocks packets from any IP address using any port that is targeting the same destination (e.g., 192.168.100.4) with more than 18 SYN packets within a second. This rule could vary based on the ML outputs.

**ML Classifier Module Implementation:** We utilized Weka [32] for use in our ML-Classifier module. Weka is an open-source tool for ML applications that provides various algorithms. Details with the data collection and training are given in Section VI-B.

**Dynamic Rules Configuration Module Implementation:** For this module, we used a script to generate the IDPS and SDN rules and to pass newly created rules from it. The script gathers the alert log data from the ML Classifier module, parses out data from the alert (source and destination IPs, Ethernet type, protocol, source and destination ports, type of service, etc.), and generates rules using these features, which are added to Snort and the SDN flow tables of both the S-Pot framework and the emulated SDN network. The controller is queried by the OVS

every 30 seconds for new registered rules on both networks as well. One advantage when generating new dynamic rules for the SDN controller is that the filter values are similar to the ones used for Snort. This facilitates creating filters for the rule. It is also possible to combine multiple conditions. Rules are pushed to the SDN controller in XML or JSON format. Listing 1 demonstrates an example SDN rule generated by S-Pot to be passed to the SDN controller to drop packets that match the source IP address of x.x.x.x/30 and port 22222, destination IP address of y.y.y.y/24 and port 8080, Ethernet source and destination addresses.

Listing 1: A sample SDN rule generated by S-Pot for the enterprise SDN controller to drop packets from x.x.x.x/30 to y.y.y.y/24

```
{ "flow2": [
    { "id": "2",
      "table_id": 2,
      "flow-name":"flow2",
      "strict": 2,
      "match": {
        "ipv4-source": "x.x.x.x/30",
        "ipv4-destination": "y.y.y.y/24",
        "ip-match": {
          "ip-dscp": 2,
          "ip-protocol": 6,
          "ip-ecn": 2},
        "in-port": "0",
        "tcp-source-port": 22222,
        "tcp-destination-port": 8080,
        "ethernet-match": {
          "ethernet-type": {
            "type": 2048},
          "ethernet-source": {
            "address": "00:02:b2:f2:c3:05"},
          "ethernet-destination": {
            "address": "00:02:c4:a6:b1:03"}}},
      "cookie": 3,
      "instructions": {
        "instruction": [
          { "order": 0,
            "apply-actions": {
              "action": [
                { "order": 0,
                  "drop-action": {} }]}}]}}]}
```

### B. Data Collection and Processing

To test the performance of S-Pot in classifying attacks, we prepared a dataset with TCP SYN Flood DoS, PUSH and ACK Flood DoS, UDP Flood DDoS, scanning, trojan malware, and unknown attacks. In this study, unknown attacks represent zero day attacks and are all the logged packages which did not fit into the other attack classifications that the classifier came across during the training. These were generated using pcaps. Next, we used libpcap [33] to capture network traffic for eight consecutive hours and performed variations of the selected attacks. For this, we used hping3, LOIC, Nmap, and pcaps, respectively. We stored the captured network traffic into JSON files and converted them into CSV format for use in Weka. We chose the following features based on the values of each feature, the statistical relationship, and the redundancy of the features to produce a more accurate model [34]: *inter-arrival time (IAT), packet direction (dir), destination address and port (dst_ap), destination MAC address of Ethernet (eth_dst), source MAC address of Ethernet (eth_src), packet length (pkt_len), protocol type (proto), source address and port (src_ap), capture time (timestamp), transmission control protocol flags (tcp_flags),*

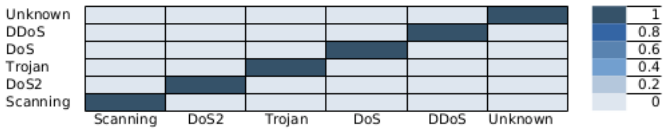


Fig. 3: Confusion Matrix generated from the result of the classification of different types of attacks using J48 algorithm. *transmission control protocol window (tcp_win)*, and *time to live (ttl)*.

TABLE I: Performance evaluation results of S-Pot.

| Model | TPR | FPR | Prec. | Recall | F1 | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| RF | 0.971 | 0.02 | 0.965 | 0.971 | 0.968 | 0.995 | Scanning |
| | 0.945 | 0.016 | 0.925 | 0.945 | 0.935 | 0.975 | DoS2 |
| | 0.947 | 0.001 | 0.973 | 0.947 | 0.96 | 0.976 | Trojan |
| | 0.95 | 0.011 | 0.963 | 0.95 | 0.957 | 0.996 | DDoS |
| | 0.996 | 0.004 | 0.979 | 0.996 | 0.987 | 0.998 | DoS |
| | 0.881 | 0.003 | 0.95 | 0.881 | 0.914 | 0.986 | Unknown |
| | 0.958 | 0.013 | 0.959 | 0.958 | 0.958 | 0.991 | Avg. |
| J48 | 0.988 | 0.014 | 0.975 | 0.988 | 0.981 | 0.986 | Scanning |
| | 0.931 | 0.007 | 0.968 | 0.931 | 0.949 | 0.965 | DoS2 |
| | 0.951 | 0 | 1 | 0.951 | 0.975 | 0.965 | Trojan |
| | 0.969 | 0.012 | 0.961 | 0.969 | 0.965 | 0.987 | DDoS |
| | 0.991 | 0.005 | 0.97 | 0.991 | 0.981 | 0.994 | DoS |
| | 0.936 | 0.003 | 0.962 | 0.936 | 0.949 | 0.987 | Unknown |
| | 0.97 | 0.01 | 0.97 | 0.97 | 0.969 | 0.983 | Avg. |
| BayesNet | 0.942 | 0.011 | 0.979 | 0.942 | 0.96 | 0.994 | Scanning |
| | 0.927 | 0.017 | 0.921 | 0.927 | 0.924 | 0.986 | DoS2 |
| | 0.976 | 0.004 | 0.87 | 0.976 | 0.92 | 0.976 | Trojan |
| | 0.935 | 0.018 | 0.94 | 0.935 | 0.937 | 0.99 | DDoS |
| | 0.996 | 0.004 | 0.974 | 0.996 | 0.985 | 0.997 | DoS |
| | 0.927 | 0.013 | 0.835 | 0.927 | 0.878 | 0.992 | Unknown |
| | 0.945 | 0.013 | 0.947 | 0.945 | 0.945 | 0.991 | Avg. |
| SMO | 0.968 | 0.022 | 0.961 | 0.968 | 0.964 | 0.985 | Scanning |
| | 0.934 | 0.017 | 0.922 | 0.934 | 0.928 | 0.96 | DoS2 |
| | 0.953 | 0 | 1 | 0.953 | 0.976 | 0.979 | Trojan |
| | 0.943 | 0.013 | 0.958 | 0.943 | 0.95 | 0.978 | DDoS |
| | 0.996 | 0.004 | 0.974 | 0.996 | 0.985 | 0.996 | DoS |
| | 0.899 | 0.004 | 0.942 | 0.899 | 0.92 | 0.995 | Unknown |
| | 0.955 | 0.015 | 0.955 | 0.955 | 0.955 | 0.981 | Avg. |

In the *Data Cleaning* step, the collected data was cleaned by extracting only the selected features and forming the feature vectors. We eliminated the features that are not relevant to the classification of the attacks (e.g., rule revision, generator id, eth_type, or where columns have missing information (e.g., class, service). In the *Feature Selection* step, we combined the selected features such as seconds, the source IP address, and the packet length or seconds and packet number to identify the type of attack. Seconds, which represent the inter-arrival time between each continuous packet, have been determined to be helpful to identify different types of attacks in networks [35]. We began by labeling the known network traffic generated each as a different class. Class categories are presented as DoS, DoS2, DDoS, scanning, trojan malware, and unknown, where DoS represents TCP SYN Flood DoS attacks, DoS2 represents PUSH and ACK Flood DoS attacks, and DDoS represents UDP Flood DDoS attacks. DoS attacks were from a real static IP address flooding the target, while DoS2 was a spoofed IP address to simulate a trusted source. To identify these attacks, the selected features were used in *Classifiers* component of the ML Classifier module to build ML models using Random Forest, SMO, BayesNet, and J48 algorithms. Feature combinations were selected for the identification of

each attack type. DoS, DoS2, and DDoS attacks were identified through feature combinations of identical IAT, src_ap, dir, and packet length, consecutive values of timestamp, pktnum, and src_ap, and identical values of pktlen, and tcp_win. When the traffic is benign, it was observed that these features change uniformly. Also, while maintaining the same dest_ap, the src_ap port number increases by one when it is under DoS attacks. For the scanning class, dst_ap, TTL, eth_src, and tcp_flags were considered to detect active scanning. For the Trojan malware, Snort detected the packets as malicious based on direction flowing to the server, dest_ap, pkt_len repeated in groups of three, and proto targeting HTTP. Finally, *Classifier* was trained using a multi-class classification of labeled signatures from known attack types by applying a supervised approach.

### C. S-Pot Classification Accuracy

In this section, we evaluate the accuracy of S-Pot in detecting the attacks. Table I presents the results considering several accuracy metrics such as True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall, F1, and ROC Area. The BayesNet model showed the best performance for identifying scanning, achieving 97.9% precision. For identifying trojan malware attacks J48 and SMO showed the best performance with 100% precision. The metrics proved the most accurate in regards to DoS attacks, with RF achieving 97.9% precision. In regards to DDoS attacks, RF demonstrated the best performance with 96.3% precision, whereas J48 proved to have the highest precision in regards to DoS2 attacks with 96.8% as well as unknown attacks with 96.2%. J48 provided the best performance overall, with the highest average true-positive rate of 97%, lowest average false-positive rate of 1%, and highest average precision of 97%. The obtained results show that the J48 model is the most suitable classifier to be used in the ML Classifier module of S-Pot. Figure 3 displays the confusion matrix obtained as a result of the classification of different attack types and the J48 algorithm.

### D. Performance Evaluation of the SDN Enterprise Network with S-Pot vs. without S-Pot

To compare the security of an enterprise network with and without S-Pot, we built an enterprise SDN testbed network that includes the components tabulated in Table II. In this evaluation, we simulated TCP SYN Flood DoS, PUSH and ACK Flood DoS, UDP Flood DDoS, scanning, and trojan malware attacks using the aforementioned tools and applied each type of attack 10 times.

We began by identifying the number of attacks that were effectively blocked by the use of Snort IDPS alone in the SDN network, without S-Pot. Our analysis showed that the Snort IDPS detected and blocked the trojan malware, and flagged the TCP SYN Flood DoS, PUSH and ACK Flood DoS attacks. However, it did not block the TCP SYN Flood, and PUSH and ACK Flood attacks. A rule is required to be added to Snort to block these attacks. On the other hand, Snort did not detect the UDP Flood DDoS or the scanning attacks.

TABLE II: Components of the SDN testbed network.

| Component | OS | Services |
|---|---|---|
| Enterprise Honeypot 1 | Debian 10 | SSH, HTTP/HTTPS, DNS |
| Enterprise Honeypot 2 | Windows 10 | SMB, FTP, SSH, DHCP |
| IoT Honeypot 1 | Full device emulation | SSH, Telnet |
| IoT Honeypot 2 | Full device emulation | SSH, Telnet |
| Snort IDPS | Ubuntu 20 | IDPS |
| ODL | Ubuntu 18 | Controller |
| OVS | Ubuntu 18 | Virtual Switch |

As the second step of our evaluation, we applied the same attacks (i.e., UDP Flood DDoS, TCP SYN Flood DoS, PUSH and ACK Flood DoS, and scanning) which were not detected by the IDPS of the testbed network on different honeypots of S-Pot, which in turn were analyzed by the IDPS module of S-Pot. Once again, the IDPS module of S-Pot was able to detect and block only the trojan malware. The TCP SYN Flood DoS and PUSH and ACK Flood DoS attacks were only detected but not blocked, and the UDP Flood DDoS and scanning attacks were not detected. For the attacks that were not detected by the IDPS module, the ML Classifier module of S-Pot obtained the traffic logs from the IDPS, and detected the attacks with 100% accuracy. Following the detection process, new signatures for the detected attacks were generated by the Dynamic Rules Configuration module and fed to the IDPS of S-Pot successfully. In addition, the new rules were sent to the IDPS of the SDN testbed network. Once this process was completed, as the last step of the evaluation, we applied the same attacks against the hosts on the testbed network again. At this point, the IDPS of the network was able to detect all of the attacks, and the packets were dropped. Our evaluation with simulated attack instances shows that S-Pot can detect new attacks and generate new rules to block attacks with 40% improvement over legacy systems without S-Pot, based on the applied attacks, thus effectively improving the security of an SDN network. It is important to note that while our analysis only focused on S-Pot effectiveness against these five types of attacks, this is expandable to all kinds of attacks as new attack data is captured by the honeypot module and the ML Classifier module is also extendable.

## VII. Conclusion

In this paper, we proposed S-Pot, a novel smart honeypot framework that aims to improve security of SDN networks. S-Pot benefits from honeypots that can simulate both enterprise services and IoT devices in gathering attack information and employs an IDPS and ML classifiers to detect and learn from the attacks in honeypots. Unlike research honeypots, it utilizes the obtained attack information from the honeypots for production purposes for the security of SDN-based networks, dynamically creates new rules in real-time for the attacks, and shares them with the SDN-based network. We implemented S-Pot and evaluated the attack detection performance of it with respect to various ML algorithms. Our results showed that J48 algorithm provides the best accuracy for S-Pot, with 97% precision. We also created a realistic enterprise SDN testbed network and tested the security of it with and without S-Pot. Our evaluations demonstrated that S-Pot can improve the

security of the SDN network and can effectively add a newly created rule to the flow tables on both the S-Pot and the realistic enterprise testbed network. These new rules can effectively block attacks based on the acquired knowledge from our S-Pot framework, improving network security.

## ACKNOWLEDGEMENTS

This work was partially supported by the U.S. National Science Foundation (Award: NSF-CAREER CNS-1453647, NSF-1663051, and NSF-2219920) and a Microsoft Research Grant. The views are those of the authors only.

## REFERENCES

[1] F. A. Lopes, M. Santos, R. Fidalgo, and S. Fernandes, "A software engineering perspective on sdn programmability," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1255–1272, 2016.

[2] The President's National Security Telecommunications Advisory Committee, "Nstac report to the president on software-defined networking," 2020. [Online]. Available: https://www.cisa.gov/sites/default/files/publications/NSTAC%20SDN%20Report%20%288-12-20%29.pdf

[3] Market Research Future, "Software defined networking (sdn) market size usd 59 billion by 2023 growing at massive cagr of 42.41%," https://www.globenewswire.com/news-release/2019/04/04/1797303/0/en/Software-Defined-Networking-SDN-Market-Size-USD-59-Billion-by-2023-Growing-at-Massive-CAGR-of-42-41.html, [Online; accessed 12-June-2021].

[4] S. Morgan, "Cybercrime to cost the world $10.5 trillion annually by 2025," https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/, [Online; accessed 12-June-2021].

[5] D. Watson, "Honeynets: a tool for counterintelligence in online security," *Network Security*, vol. 2007, no. 1, pp. 4–8, 2007.

[6] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2021.

[7] H. Wang and B. Wu, "Sdn-based hybrid honeypot for attack capture," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, pp. 1602–1606.

[8] M. Du and K. Wang, "An sdn-enabled pseudo-honeypot strategy for distributed denial of service attacks in industrial internet of things," *IEEE Tran. on Industrial Informatics*, vol. 16, no. 1, pp. 648–657, 2020.

[9] S. Kyung, W. Han, N. Tiwari, V. H. Dixit, L. Srinivas, Z. Zhao, A. Doupé, and G. Ahn, "Honeyproxy: Design and implementation of next-generation honeynet via sdn," in *2017 IEEE Conference on Communications and Network Security (CNS)*, 2017, pp. 1–9.

[10] M. Azab, A. Hamdy, and A. Mansour, "Repoxy: Replication proxy for trustworthy sdn controller operation," in *17th IEEE Int. Conference On Trust, Security And Privacy In Computing And Communications*, 2018.

[11] A. Kaan Sarica and P. Angin, "A novel sdn dataset for intrusion detection in iot networks," in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–5.

[12] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in sdn using machine learning approach," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016, pp. 167–172.

[13] M. S. Elsayed, N. A. Le-Khac, S. Dev, and A. D. Jurcut, "Machine-learning techniques for detecting attacks in sdn," in *IEEE 7th Int. Conf. on Computer Science and Network Technology*, 2019, pp. 277–281.

[14] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches," *P2P Networking and Applications*, vol. 12, 2019.

[15] I. A. Valdovinos, J. A. Pérez-Díaz, K.-K. R. Choo, and J. F. Botero, "Emerging ddos attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions," *Journal of Network and Computer Applications*, vol. 187, p. 103093, 2021.

[16] R. Swami, M. Dave, and V. Ranga, "Software-defined networking-based ddos defense mechanisms," *ACM Comput. Surv.*, vol. 52, Apr. 2019.

[17] A. R. Mohammed, S. A. Mohammed, and S. Shirmohammadi, "Machine learning and deep learning based traffic classification and prediction in software defined networking," in *2019 IEEE International Symposium on Measurements Networking (M N)*, 2019, pp. 1–6.

[18] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 393–430, 2019.

[19] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A survey of networking applications applying the software defined networking concept based on machine learning," *IEEE Access*, vol. 7, 2019.

[20] A. Molina Zarca, J. B. Bernabe, A. Skarmeta, and J. M. A. Calero, "Virtual iot honeynets to mitigate cyberattacks in sdn/nfv-enabled iot networks," *IEEE Journal on Selected Areas in Communications*, 2020.

[21] H. Polat and O. Polat, "The effects of dos attacks on odl and pox sdn controllers," in *2017 8th International Conference on Information Technology (ICIT)*, 2017, pp. 554–558.

[22] D. Tatang, F. Quinkert, J. Frank, C. Röpke, and T. Holz, "Sdn-guard: Protecting sdn controllers against sdn rootkits," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 297–302.

[23] V. Varadharajan, K. Karmakar, U. Tupakula, and M. Hitchens, "A policy-based security architecture for software-defined networks," *IEEE Tran. on Information Forensics and Security*, vol. 14, no. 4, 2019.

[24] W. Fan, Z. Du, D. Fernandez, and V. A. Villagra, "Enabling an anatomic view to investigate honeypot systems: A survey," *IEEE Systems Journal*, vol. 12, no. 4, p. 3906–3919, Dec 2018.

[25] Cisco, "Snort - network intrusion detection and prevention system," https://www.snort.org, [Online; accessed 17-May-2021].

[26] ——, "What are the most common cyber attacks?" https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html, [Online; accessed 2-Jun-2021].

[27] L. Metongnon and R. Sadre, "Beyond telnet: Prevalence of iot protocols in telescope and honeypot measurements," in *2018 WTMC*, Aug. 2018, pp. 21–26.

[28] The Linux Foundation, "Open Daylight," https://www.opendaylight.org, [Online; accessed 13-May-2021].

[29] ——, "Open vSwitch," https://www.openvswitch.org, [Online; accessed 13-May-2021].

[30] T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang, "Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices." in *Black Hat 2017*, 2017.

[31] M. Wang, "Thingpot," https://github.com/Mengmengada/ThingPot, 2017, [Online; accessed 14-May-2020].

[32] M. L. Group, "Weka," https://www.cs.waikato.ac.nz/ml/weka/, [Online; accessed 10-May-2021].

[33] T. . libpcap, "Tcpdump & libpcap," hhttps://www.tcpdump.org/, [Online; accessed 17-May-2021].

[34] T. R. N and R. Gupta, "Feature selection techniques and its importance in machine learning: A survey," in *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science*, 2020.

[35] O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, "Change-point cloud ddos detection using packet inter-arrival time," in *2016 8th Computer Science and Electronic Engineering (CEEC)*, 2016, pp. 204–209.