Computer Communications 148 (2019) 215–239

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

VARMAN: Multi-plane security framework for software defined networks

Prabhakar Krishnan [a,*], Subhasri Duttagupta [b], Krishnashree Achuthan [a]

[a] Center for Cybersecurity Systems and Networks, Amrita Vishwa Vidyapeetham, Amritapuri, India
[b] Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India

ARTICLE INFO

Keywords:
SDN
NFV
SDNFV
IoT
Cloud
Edge networks
DDoS
Botnet
Malware
Network security
Threat analytics
Security
IDS
IPS
NIDS
Machine learning
Deep learning
CICIDS2017

ABSTRACT

In the context of future networking technologies, Software-Defined paradigm offers compelling solutions and advantages for traffic orchestration and shaping, flexible and dynamic routing, programmable control and smart application-driven resource management. But the SDN operation has to confront critical issues and technical vulnerabilities, security problems and threats in the enabling technical architecture itself. To address the critical security problems in SDN enabled data centers, we propose a collaborative "Network Security and Intrusion Detection System(NIDS)" scheme called '***VARMAN***: ad**V**anced multi-pl**A**ne secu**R**ity fra**M**ework for softw**A**re defined **N**etworks'. The SDN security scheme comprises of coarse-grained flow monitoring algorithms on the dataplane for rapid anomaly detection and prediction of network-centric DDoS/botnet attacks. In addition, this is combined with a fine-grained hybrid deep-learning based classifier pipeline on the control plane. It is observed that existing ML-based classifiers improve the accuracy of NIDS, however, at the cost of higher processing power and memory requirement, thus unrealistic for real-time solutions. To address these problems and still achieve accuracy and speed, we designed a hybrid model, combining both deep and shallow learning techniques, that are implemented in an improved SDN stack. The data plane deploys attack prediction and behavioral trigger mechanisms, efficient data filtering, feature selection, and data reduction techniques. To demonstrate the practical feasibility of our security scheme in real modern datacenters, we utilized the popular NSL-KDD dataset, most recent CICIDS2017 dataset, and refined it to a balanced dataset containing a comparable number of normal traffic and malware samples. We further augmented the training by organically generating datasets from lab-simulated and public-network hosted hackathon websites. The results show that VARMAN framework is capable of detecting attacks in real-time with accuracy more than 98% under attack intensities up to 50k packets/second. In a multi-controller interconnected SDN domain, the flow setup time improves by 70% on an average, and controller response time reduces by 40%, without incurring additional latency due to security intelligence processing overhead in SDN stack. The comparisons of VARMAN under similar attack scenarios and test environment, with related recent works that utilized ML-based NIDS, demonstrate that our scheme offers higher accuracy, less than 5% false positive rate for various attack intensities and significant training space/time reduction.

1. Introduction

The rapid proliferation of IoT technologies and Cloud-connected devices are posing challenges in various dimensions such as Availability 24/7, Reliability, high-speed Interconnectivity, Computational resources, and Security. This has led to Cloud networking evolving into heterogeneous Edge networks, distributed Fog infrastructures that got enabled by SDN, NFV, SD-WAN, and multiple access technologies. However, to deal with modern network applications we require efficiently scalable architectures, frameworks and enabling platform that can provide services at scale, reliably and flexibility for handling disparate traffic load. This is where SDN [1] has shown promise, with an architecture (Fig. 1) that maintains a global view, offers programmable interfaces for monitoring or shaping the traversal of flows across the data links. This concept created a paradigm shift in networking design, operation, and management. Furthermore, researchers proposed ideas allowing third parties also to participate in network application design, deployment and dynamic policies at run time.

The pioneering deployment of SDN was undertaken by the Ethane project [2], which implemented a notion of flow-based (instead of packet-based) traffic shaping and network orchestration and thus forming the foundational aspects towards standardizing the protocols in the SDN domain. The management and security of large-scale distributed infrastructure having a myriad of networked devices with heterogeneity and capacity are daunting and that is where SDN comes in, with its global control, monitoring, programmability, and visibility. Although NFV and SDN are still evolving and developing respectively,

* Corresponding author.
E-mail address: kprabhakar@am.amrita.edu (P. Krishnan).

https://doi.org/10.1016/j.comcom.2019.09.014
Received 18 July 2019; Received in revised form 4 September 2019; Accepted 22 September 2019
Available online 26 September 2019

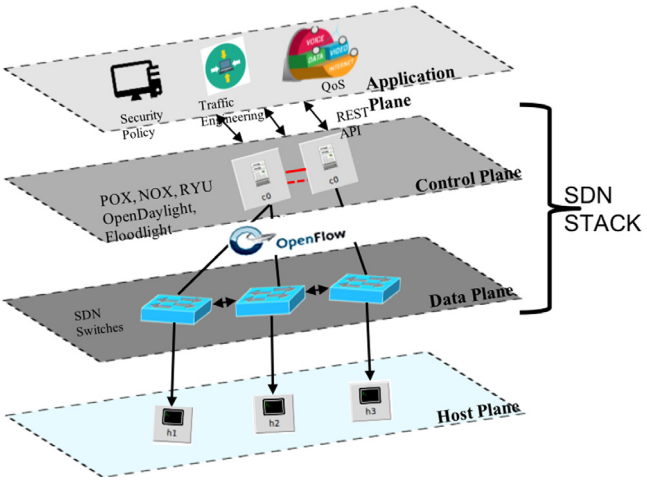

Fig. 1. SDN reference architecture.

both of them advocate open standards and open source innovations. Most of the conventional "anti-DDoS" mechanisms were originally designed based on localized standalone systems with limited attack surface coverage — only to stop volumetric attacks, the most rudimentary of DDoS assaults. While these methods proved efficient with respect to detection and mitigation, they lack the autonomic capabilities. Hence deploying and operating these defense mechanisms in large distributed networks has been challenging and mostly not feasible. SDN is one of the recent enabling technologies that promises dynamic reconfigurations, network orchestration, and efficient traffic engineering through a programmable interface. It centralizes the network operational strategy by separating the packet "*forwarding* (Data Plane)" and the "*routing* (Control plane)" with a control channel *OpenFlow* [1] protocol. The network operations and behavior can be centrally controlled and programmable through applications using a rich set of Northbound protocol and APIs. OpenFlow is a Southbound flows-based control protocol between the controller and switches on the dataplane. This architectural model can be very efficient to conventional networking topologies for traffic orchestration, scalable, multi-path transparent path migration giving a fault-tolerant agile networking fabric for modern datacenters. The modern adversaries craft much more sophisticated planning, scanning, and reconnaissance that leads to more lethal persistent botnet/malware distributed attack strategies, in comparison to attacking legacy networks.

To solve the problems in network intrusion and detecting sophisticated attacks, SDN architecture is more suitable to deploy ML algorithms, than implementing NIDS in the conventional network model, leveraging the higher computational resources in the control plane. To exemplify this notion in an SDN enabled data center, the packet stream are considered as flows and there is an inherent relation to all the packets in any particular flow and the switch in the data plane provides the statistical values of its counter to the control plane. The centralized controller, which has a global view of all the switches in the data center can expose through Northbound "API" to the applications to run ML based analytics on the flow data. ML-based schemes can mine a large trove of time series data and draw insight to detect network intrusions and forecast future network attacks [3]. As the technology advanced to bring multi-core and hardware-accelerated CPU (e.g. scalable Intel Xeon and Phi) many-core co-processors (e.g., GPGPUs and Tensor units), thus opened up the platform for ML-based analytics and deep computational networks [4,5]. The dataset and the training the machine with labeled data, etc. are the core factors that determine the efficiency and practical feasibility of any ML-based NIDS. As the SDN model is inherently designed to centrally collect the global statistics and run time data from all the devices in a network domain, the control plane offers the optimal placement of ML-based Intrusion detection/Anomaly detection applications. Further, based on both historical and real-time data analytics, SDN offers the programmable model through API, for automating the provisioning, services chaining, dynamic security decisions and traffic policies at run time [6].

Klaine et al. [7] provided a survey of Machine Learning-based network classification systems in the context of mobile networks. Fadlullah et al. [8] presented their views of applying ML-based approaches for improving the traffic management and security in modern industrial control networks. Hodo et al. [9] were one of the early proposers of the hybrid scheme in ML algorithms for NIDS problems and described various issues of ML-based IDS in detail.

From most of the significant research works, we observed that existing ML based classifiers improve the accuracy of IDS, however, at the cost of higher processing power and memory requirement, thus unrealistic for real-time solutions. Hence, through this research, we addressed some of the most critical issues of NIDS in SDN enabled deployments. ML Intrusion detection systems have generally been used one class of algorithm either shallow ML or deep learning algorithms to detect attacks. But in recent times, using two or more different techniques to form a hybrid has improved the overall performance. To address these problems and still achieve accuracy and speed, we designed a hybrid model, that combines both deep/shallow learning techniques and implemented in an improved SDN stack.

Evaluating a particular machine learning based technique is primarily dependent on the training datasets and it may lead to diverging results. Therefore, to compare one ML technique over another approach, the datasets utilized to run the training and test, etc. need to be similar and selection of the appropriate dataset is crucial for anomaly detection. The accuracy of a model also depends on how organic, diversified preparation of the training dataset for a particular application. Over the years of advancement of this field, various datasets were generated, validated by practical experiments as a representative data of network traffic and attack environment. Some popular datasets (such as "DARPA98, KDD99, ISC2012, and ADFA13") are enumerated by [10] and several researchers have utilized these datasets for evaluation of their network management and intrusion detection approaches. But ironically, most of these datasets have been outdated and not reliable to be utilized for training ML-based systems. The most common caveats about such outdated datasets are synthetic, lacking coverage of latest network-centric malware/attacks, features, signatures and missing recent patterns/trends. Consequent to unreliable and not-validated datasets, the IDS/IPS, anomaly detection and other network security solutions lack consistency and accuracy when deployed in real networks. We are one of the few researchers to utilize the popular NSL-KDD [11], recent benchmark CICIDS2017 [12] for the SDN deployment and another recently published HogZilla [13] dataset. We refined these datasets further, to extract a normalized dataset (i.e., containing comparable number of normal traffic and malware samples) and thus organically generate realistic datasets for training the intrusion detection systems.

Given the interplay between multiple planes or layers of the network for attack detection and mitigation, we de-couple the problem into three subproblems, namely (a) anomaly detection in data-plane, (b) classification in control-plane and (c) collaborative interface between the layers and downstream services. This work is an attempt to address these problems and decide the best trade-off between switch forwarding rate, reliability of the network and security of the services. To this aim, we propose an ML-based collaborative NIDS scheme called '***VARMAN***: ad**V**anced multi-pl**A**ne machine learning secu**R**ity fra**M**ework for Softw**A**re Defined **N**etworks'. The VARMAN security scheme comprises a coarse-grained behavioral and statistical monitoring on the dataplane for anomaly detection/prediction of network-centric DDoS/botnet attacks. And this is combined with a fine-grained ML-based classification on the control plane.

Our dataplane design comprises of two-tier pipeline that is capable of monitoring and detecting anomalies in flow characteristics,

TCP/IP/OpenFlow protocol behaviors, by reading counter values of the switches. The packets belonging to suspicious and reduced flows are marked and sent to control plane for ML-based classification. Eventually, depending on the classified result, only the control plane and security applications determine the specific Network-Function (NF) or the attack remediation to be executed on the analyzed flow. Once the defense action has been instructed to the dataplane, the switches would load the flow-rule/NF for those flows and until the action has been recalled the flow rules are cached for future packets in that flow. Hence, this VARMAN SDN architecture has an optimal communication performance due to traffic reduction on the control channel and superior classification speed as well due to reduced load on the controller.

The novel hybrid ML-based detection scheme at the control plane is constructed with a non-symmetric stacked autoencoder and an improvised random forest classifier, ie. Combining deep and shallow learning algorithms. We have also deployed efficient data filtering, feature selection, and reduction mechanisms to effectively reduce the data processing overheads on this ML-based NIDS. Most notably, unlike most prior research publications, we present the experiments conducted against the most recent dataset resembling the real-world attack profile and the consistency of accurate detection as well. We further augmented by organically generating datasets from lab-simulated and public-network hosted hackathon websites. The results show that VARMAN framework is capable of detecting attacks in real-time with accuracy more than 98% under attack intensities up to 50k packets/second. In a multi-controller interconnected SDN domain, we observed flow setup time improved by 70% on average, and controller response time reduced by 40%, without incurring additional latency due to security intelligence processing overhead in SDN stack. The comparisons of VARMAN under similar attack scenarios and test environment, with related recent works that utilized ML-based NIDS, have demonstrated that our scheme offers higher accuracy, less than 5% false positive rate for various attack intensities and significant training space/time reduction.

The main contribution summary of our paper:

- Two-tier dataplane pipeline for monitoring/defense
- Multi-stage detection and classification
- Data plane detection, statistical, attack forecasting and prediction, SDN protocol-based trigger, protocols anomaly NF data plane, Light-weight Flow/aware sampling monitoring in switches
- Control plane overload prevention/OpenFlow saturation, as reduced flows to the controller for classification
- Hybrid deep and shallow machine learning-based classification system that has reduced training time and overhead
- A multi-plane security framework that exploits collaborative intelligence between data, control plane and application plane and a security orchestration layer with SDNFV controller
- NF offloading mechanism to dynamically deploy defense actions across the dataplane switches for rapid attack reaction

In our previous articles [14] and [15] we began building a practical SDN-based NIDS [16] by experimenting different design choices, case studies for IoT network, exploiting a simple ML-classifier on the control plane, with limited datasets. The early results with our first version of NIDS showed promise and so, we progressed the research in this direction to exploit the advanced machine learning algorithms. This work extends the functionalities [16] by introducing light-weight detection in the dataplane, optimized ML model and a cross-plane linking layer. We analyze our solution by integrating our security scheme in production-ready SDN stack, in various realistic network scenarios and evaluate it in an experimental testbed. The main difference between traditional frameworks in SDN and VARMAN is that we take the data plane into consideration for fine-grained security monitoring and first level defense, to achieve cross-plane optimization. Further, based on our systematic literature survey in this area, we claim that our work is one of the early adopters of this convergent technologies and ensemble of techniques. i.e., DDoS attack detection, SDNFV protocol analysis, multi-plane attack detection scheme, Security aware SDN framework, Stateful/application-aware dataplane switch fabric, Hybrid-deep/shallow learning in modified ML pipeline, organically deriving datasets from traffic simulation and most recent datasets for training a hardened ML NIDS scheme.

The paper is structured as follows: Section 2 provides the background of the enabling technologies and discusses the related works, Section 3 presents our security proposal and solution model, architecture and operations of various sub-systems, layers, and components of our solution, Section 4 presents the results of our experimental studies, Section 5 discusses the key results and future outlook, and Section 6 concludes this paper.

2. Background and related works

Network-centric attacks and malware such as DDoS/Botnet, have been haunting the IT-enabled enterprises for decades and discussion on these topics is beyond the scope of this article. We already presented an overview of some of the key research works in this software-defined and virtualized networking technologies in the previous section. As discussed in the introduction, there is a scarcity of publications that exploit the benefits of integration/convergence of the advanced technologies and adopt the advanced ML/DL algorithms/datasets for modern network architecture. Here, we discuss some of the advanced solutions relevant to SDN and key challenges in securing SDN and DL/ML-based NIDS solutions.

2.1. State of the art

Most of the commercial solutions from vendors such as Cisco, Juniper have targeted to address virtualizing NF services, micro-services chaining, and network services orchestration. And some of the popular research publications have taken the control-plane based NIDS approaches. On the contrary, we have made significant strides in extending the dataplane functionalities in terms of designing an advanced OF pipeline with stateful security aware monitoring, light-weight statistical features-based anomaly detection mechanisms and dynamically enforceable NF service chain. One of the early works with this approach was from Curtis et al. [17] who presented "DevoFlow", a framework that keeps the traffic within dataplane and thus reducing the load on the control channel. However, this Devoflow did not handle NF service chaining and other key functionalities in the modern virtualized datacenters.

Hu et al. [18] proposed a light-weight "DDoS Flooding Attack Detection and Mitigation System (FADM)" for SDN networks. The real-time attack detection application is implemented in the control plane and utilizes an entropy-based method to extract features and applies SVM for attack classification. M Gurusamy et al. [19] presented an SVM based ML module that dynamically classifies traffic in an IoT network, utilizing linear and non-linear kernel filters (RBFs) and responds with mitigation actions.

Deep Learning (DL) is a method, that advanced from artificial neural network (ANN) methods, exploits low-cost compute power and learns the representation of data at various generalizable levels. Hence DL algorithms are naturally suitable to build IDS/IPS mechanisms in modern data centers. It is proven in the literature that when compared to ML-based classifiers, DL techniques can outperform in model logic/analytics tasks. In the current threat landscape, in order to defend the sophisticated adversaries and zero-day attacks, the DL algorithms such as "Autoencoder, ANN, RNN" are considered effective. Also, some promising research works [20,21] have demonstrated that DL algorithms excelled the other ML-methods for network intrusion detection systems in SDN-enabled networks. We also examined the inherent limitation when we utilize ANNs for SDN security applications, as they commonly fitted for nonlinear problem and limitation of local minimal that results into longer training phase, with respect to the feature set.

Deep belief networks (DBN) with the ANN principle, represent a system with many "Restricted Boltzmann machines (RBM)" stacked into a model. The DBNs are efficient due to the multi-layer training scheme and suit the hierarchical feature detection use cases.

The random forest (RF) classification has wide usage to estimate QoS indicators, network IDS, traffic classification and threat assessment. Research studies from [22,23] show that Random Forest is an efficient and fast method in the context of classification of network attacks. Sharafaldinetal [12] used a Random Forest Regressor to determine the best set of features to detect each attack family and further examined the performance of these features with different algorithms that included "K-Nearest Neighbor (KNN)", Adaboost, "Multi-Layer Perceptron (MLP)", Naïve Bayes, Random Forest (RF), Iterative Dichotomiser 3 (ID3) and Quadratic Discriminant Analysis (QDA).

Latah et al. [24] presented a comparative analysis of the ML-based approaches for SDN-based NIDS solutions and proposed a novel IDS scheme, utilizing "Principal Components Analysis (PCA)" for reducing features and evaluated their model using NSL_KDD datasets. They demonstrated improved results in terms of accuracy, false-alarm and recall in comparison to conventional Supervised ML algorithms. Miettinen et al. [25] proposed an autonomic threat detection and security framework called "IoT SENTINEL", for SD-IoT networks. They utilized dynamic flow-analysis and filtering of vulnerable devices, with the device profiles, fingerprinting the traffic patterns of devices in the network and used Random Forest as the main classifier. Ajaeiya et al. [26] implemented an ML system to analyze, detect and react to cyber-attacks in an SDN-enabled network. They utilized the Random Forest classifier and trained their NIDS with the "ISOT botnet" datasets, to achieve reasonable accuracy and classification results compared to other traditional methods.

Huang et al. [27] presented their systematic study of adversarial attacks on SDN-based NIDS system. Their deep learning model (DNN) could detect ports-scanning attack based on the OpenFlow protocol messages (PACKET_IN) and flow statistics available at the controller. They simulated different normal datasets and adversarial attacks such as "Jacobian-based Saliency Map Attack (JSMA)" in their experiments with different ML algorithms and compared the performance under the adversarial datasets. Alrawashdeh [28] recommended the DBN technique for classification system utilizing a three-layer model with a hidden layer and applied unsupervised features reducing technique. They claimed improvement over similar proposals (according to their paper) at that time, with an accuracy of about 97.9% and a false alarm of 2.4%. Ihsan H. Abdulqadder et al. [29] presents a robust security scheme to provide fortification against major threats along with user privacy in SDN based 5G network with NFV enabled cloud environment. Entropy analysis is performed at controller with the intention to detect DDoS attack. To alleviate the DDoS attack, Hybrid Fuzzy with Artificial Neural Network (HF-ANN) classifier is employed in sVNF in virtualized cloud, which classifies the network traffic as suspicious and normal, based on packet features.

In this article, we will consider two deep generative models namely Stacked Auto-Encoder (SAE) and Deep Belief Network (DBN), that are employed in SDN by few popular research papers. We will compare and contrast the DBN model with our novel NDAE scheme for unsupervised feature learning, which unlike regular SAE based schemes, provides non-symmetric features reduction and improves classification performance when compared with leading methods such as DBNs.

2.2. Research challenges

We observe a rapid advancement of research in this combination of ML/AI based classification systems and SDN. Due to the flexibility, ease of control through a central controller, with the global view, many creative network security solutions are proposed in this field of SDN. We present an overview of some of the barriers and challenges that ML/SDN faces in deployment. Some of the notable publications [30,31] present a systematic survey and taxonomy of the key security problems, vulnerabilities and also proposed approaches in this cross-cutting SDN/ML and security. The next generation SDN 2.0 will bring in new "East-West side channels", reliability, scalability features in terms of multi-controller protocol standardization, SD-WAN, SD-Clouds will demand more robust schemes from future AI-based proposals. In the context of ML-based-analytics and SDN, we categorize the challenges into two areas (i) Machine Learning centric issues, (ii) SDN architecture inherent vulnerabilities and each of these areas are seeing a significant research evolution.

ML-centric challenges: The key issue in dataset-driven ML system is the optimal selection of features relevant to the network attacks and from the dataset with diverse features, the challenge is to design the efficient feature-reduction algorithm. There is a dearth of realistic tests/benchmarks, validated dataset to fine-tune the classification accuracy, consistency and reliability of the trained machine for real-time use in NIDS. Tam. N Nguyen study [32] inferred that the ML-based security solutions for SDN face issues with (i) A large portion of training data for ML-based SDN security applications is synthetic and is not realistic enough, so hard to find organic training data; Because these datasets were developed by research institutes and made available to the public, ML-based solutions trained on those datasets alone can be out-maneuvered by adversaries who studied the same data. (ii) a semantic gap between initial work and practical real-world deployments; (iii) enormous variability in input data; (iv) measuring and minimizing the cost of errors; and (v) other difficulties in evaluation.

SDN architectural issues: Control channel saturation, Southbound bottleneck, controller overhead, and topology poisoning are some of the major attack vectors on the SDN enabled networks. The adversaries could exploit this vulnerability with "distributed denial-of-service (DDoS)" or *newflow* attack towards the SDN domain. Consequently, the SDN network will crash due to control plane saturation/controller's overload even before the targeted end-servers going down and this effect is contrary to conventional network in which DDoS attacks target the servers or application services [33]. The common approaches using SDN for security applications are based on inspecting packets or flow entries, flow analysis, southbound traffic optimization and scaling control plane capacity [34,35]. The authors [36] proposed a DDoS detection solution called "Overwatch" that comes with cross-plane functionalities [37] and they utilized both coarse and fine-grained classification algorithms in their SDN system. Ying-Dar Lin et al. [38] presented an advanced SDN model, which restricts most of the network-functions on the flows in the dataplane thereby the control channel overhead is reduced. They presented the results of some experiments conducted with network intrusion scenarios and proved that the extended SDN solution is efficient by integrating costly network functions in the dataplane OF pipeline.

This paper [38] proposed an extensible SDN/NFV architecture by deploying the virtualized network functions (VNFs) into the dataplane and presented their experiences with security firewall applications and micro-benchmarks to measure the impact on the SDN critical components (controller, OpenFlow channel). The efficient and reliable functioning of traffic control in SDN enabled domain is dependent on the topology discovery services running in the SDN. But this topology protocol can be poisoned with different attacks and controller need to have intelligence and fine-grained monitoring the flow rule/behaviors to flag any such attacks. In [17], one of the earliest works that tackled this SDN limitation, authors proposed a modification of the OpenFlow model through DevoFlow will allow operators to target only the flows that matter for their management problem. DevoFlow reduces the switch-internal communication between control- and data-planes by (a) reducing the need to transfer statistics for boring flows, and (b) potentially reducing the need to invoke the control-plane for most flow setups. Therefore, it reduces both the intrinsic and implementation overheads of flow-based networking, by reducing the load on the network, the switch, the control-channel, and the central controller.

In recent years, there is an active research redesigning of SDN 2.0 is going on, with the possibility to offload a portion of stateful flow management and controlling functions down to the data plane switches. We have proposed one such extended stateful dataplane design in VARMAN, the notion is to avoid the control channel choking (“switch-to-controller”), communication overhead and a delayed response for *newflow* connections, by adding intelligence to dataplane. The primary goal of this design is to keep the packet inspection and defensive actions within the data plane. If some packets are routed through a different service chain within the data plane or forwarded to a packet-scrubber connected to another port, it is optimal to keep the packets within the switch.

3. Proposed security framework

In this section, we present our multi-Plane NIDS framework for SDN-enabled networks called VARMAN and discuss in detail the design motivations, architectural improvements, key components in each plane of the solution when deployed. Because of the flexibility, programmability and dynamic management of SDN and the possibility of Network-Functions/services deployed as virtual-machines or software, the design and deploying of network-centric applications are made simple with our framework. The SDNFV orchestration plane, having global view and control, makes it the vantage point of the entire network to consolidate statistics, orchestration, real-time monitoring, flow analysis and enforce SLAs and policy decisions as well. VARMAN incorporates a multi-stage granular threat prediction and attack detection scheme.

In the first stage, anomaly detection based on thresholds, rate-limiting and protocol analysis on data gathered from switch counter to filter attack flows. The extended pipeline in the dataplane includes NFV and Intelligent security applications that marks some suspicious flows as ‘malign’ and forwards the packets from those flows to control plane. At the control plane, fine-grained ML-based deep neural network classification algorithms are run over these suspicious flows and the attacks are identified with very high accuracy. Once the attack class is determined, at each plane defensive or remediation functions are pre-deployed for quicker response to the attack and for DDoS/botnet attacks smart topology tracking applications from the defense library are executed to trace the source of these attacks and drop these malign packets at the origin switches. Finally, the SDNFV orchestrator at the cross-plane layer co-ordinates and supervises the entire network with the global view and controls all the entities at each plane. The global topology engines can help in redirecting or remediation of affected controllers, installing alternate flow rules in the switches and overseeing the framework in order to fulfill the application level Quality-of-Services (QoS) and security for multiple SDN enabled network domains. The design and detailed functioning of the various key modules and components at all planes or layers of the SDN enabled network under VARMAN framework is presented in these sub-sections here.

3.1. Architecture overview

In designing the VARMAN architecture, we maintained the layering and abstractions, API across the layers with SDN OpenFlow reference specifications [1], while the extensions and improvements are introduced at all planes of a large interconnected multi-domain cloud networks. The SDNFV architecture (Fig. 2) consists of light-weight monitoring and intelligent data plane with extended NFV functionality; sophisticated ML-analytics driven control plane and a cross-connected orchestration layer. We designed intelligence in every layer, an API as part of the VARMAN framework to develop and deploy most updated defense applications.

Controller: This is a multi-threaded software, that leverages uploaded in-band messages, feature-digests, synopsis to classify attack type. From the suspicious flow stream, in real time, key features are extracted from the packet-metadata and utilizing an efficient hybrid ML-based Classifier system, attacks are classified in specific classes. Once the malware is identified, the control plane modules organize specific action-plan and flow rules for the specific series of switches in the attack path. With its global view of the attack sources, calls the defense-action library to implement specific defense-action in the switches that are in the path or closest to attack source.

Data plane: Therefore, on the data plane, a lightweight statistical Detection system is deployed that utilizes DDoS attack sensors embedded in the switches. This layer consists of OpenFlow (OF) switches, hybrid OF-enabled Edge switches, IoT/Multi-Access-Edge Gateway switches. This layer is managed by the Controller, primarily through OF protocols for data switching, security monitoring and policy enforcement through switch management protocols. Probes/Sensors monitor flows/packet-stream for attacks and execute the corresponding defense/mitigating actions. The switches in the VARMAN dataplane are pre-built and loaded with NFVx and the standard API conforming to ONF [39].

Network Function Virtualization Extension (NFVx): This extension is designed as a sub-system or layer integrated within the data plane switches and Openflow pipeline. NFVx extends the data plane and helps in light-weight and high-speed anomaly detection and threat forecasting functions. The openflow software stack on the switches run augmented Openflow 1.5.1 with customizations to implement the security logic and intelligent functions of the workflow. Based on the mitigation scheme instructed by the security controller and switch management, the switch will dynamically load those *modules* and execute the *action/function* matching to that new-flow/packet or on flows.

Cross-coordinating layer: This global layer connects across all the layers of the network, also interconnected with multiple domains (e.g. SD multi-Clouds) will span control, monitoring, and security at each layer and interface. The orchestrator oversees the various parameters that include security and QoS, controller response time, controller overload indicator, inform the switches connected for re-discover the route, or query switches periodically or whenever *newflow* is detected. It collects data from all security modules across all the planes in the entire network, to build a topology map and network flow reputation database.

In summary, the VARMAN architecture involved modifications on the switch for the extended intelligence and local control. We combined multiple behavioral analysis and anomaly detection techniques to intercept at multiple stages and plane, drop malware/attack packets early in the network path. And with a dynamic co-ordination and situational awareness provided by the cross-plane layer, leveraging all available compute/memory resources on the switches and controller platforms.

3.2. Operation overview

Fig. 3 illustrates the operation of the entire VARMAN multi-plane security architecture with the newly introduced cross-coordination layer.

Once VARMAN starts running, various attack sensor modules keep monitoring every flow on the data plane constantly. If any abnormal flow is captured (i.e., DDoS attack traffic), the specific switch runs attack detection and first level mitigation response. Also notifies control plane with information including an alert message containing the synapse of the features from the packet. In the first stage of the attack detection phase, the data plane executes the following coarse-grained analysis:

1. Statistical anomaly detection using rate limiting thresholds and switch level counter features
2. Attack Prediction/ Forecasting using SDN protocol Triggers and characteristics
3. Anti-spoofing system using a combination of probabilistic Challenge/Response and network packet level filters

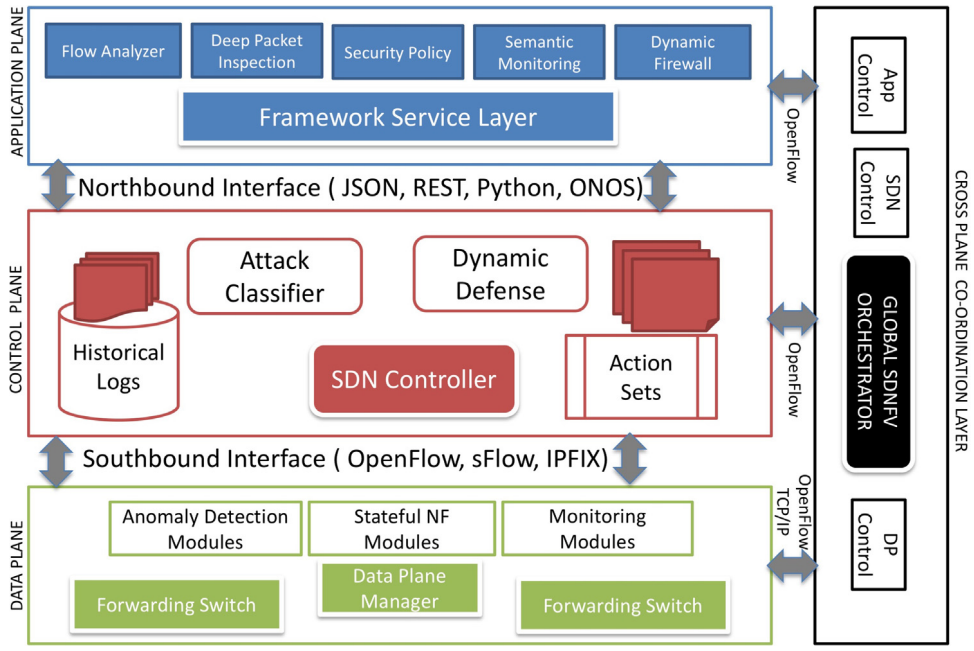

Fig. 2. Multi-plane SDNFV orchestration.

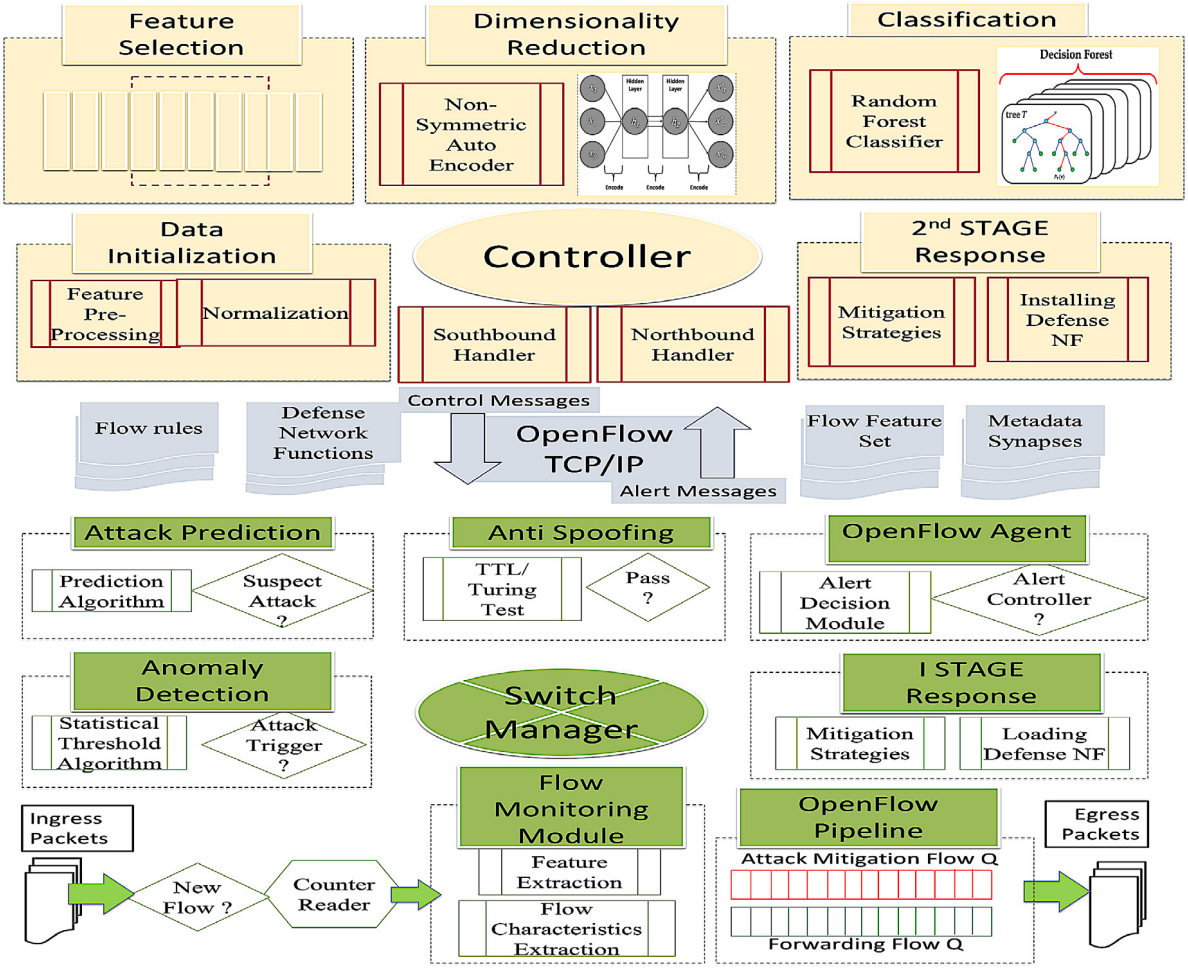

Fig. 3. VARMAN multi-plane security framework.

Once abnormal flows will pass through the threat analytics system that leverages uploaded attack features to find out DDoS attack types and its global perspective to locate the attack sources. In the second stage, the following operations are performed as part of the fine-grained analytics and remediation in the control plane.

1. NDAE based feature selection and dimensionality reduction Algorithm
2. Improved RF algorithm for multi-class classification of the types of attacks
3. Mitigation system to block/defend the attacks
4. Remediation logic to establish an alternate control channel and/or honeypot detection system

Once the particular attack type is identified, the controller invokes the defense actuator library to implement specific defense actuators on the switches that are close to attack. As the remediation phase, the overloaded controller will be reassigned to different controller channel, the loaded actuator will be executed on the specific switch, defending against certain DDoS/Botnet attacks in the first place. After the attack is eliminated, the defense actuators used for defense will be removed from certain switches.

3.3. SDN data plane

The data plane is where packets are forwarded; leveraging computing resources on the data plane to determine an attack coarsely and locally is quite reasonable. Therefore, on the data plane, we designed a light-weight monitor mechanism to collect counter values, traffic flows on switches. The data plane switches report network alerts/synapses to the controller after processing and in response to this, the controller installs the "extended flow-table entries" on the dataplane switches. Based on the flow-rule installed by the controller, it also extracts specific fields from the incoming packet stream. First, the data plane switches should be capable of capturing the main features of DDoS attacks. Second, after the reaction strategies are made by the control plane, reaction functionalities should be loaded from the control plane to dataplane dynamically. Finally, these actuators can be executed on the dataplane to filter out attack packets. The features are extracted in the switch hardware counter OF action handlers using the OpenFlow specification 1.5.1 with new extensions. The intelligent security Network Functions (NF) pipeline runs attack forecasting algorithm to detect anomalies in the flows and predict impending attack campaign or an attack under progress. The prediction is further validated by a turning-test NF that will either confirm the attack and trigger immediate action referring to the pre-built defense response rule-set. If results of the coarse-grained detection come out as inconclusive and those flows are marked suspicious for fine-grained analysis at the control plane. The unflagged flows are processed as benign and the usual forwarding process is carried out for the normal traffic. The switch manager coordinates the entire workflow operations including the main functions that run in the switch CPU. Fig. 4 illustrates the switching pipeline of the VARMAN dataplane switch, and dynamically loadable defense application modules and network functions (NFs) deployed in the OF flow-table.

3.3.1. OpenFlow extensions

We chose to implement our SDN stack adopting the OpenFlow specification v1.5.1, that defined three flow match fields for TCP (shown below in Fig. 5). A new OXM field OFPXMT_OFB_TCP_FLAGS has been added to match the flag bits in the TCP header (EXT-109). This field allows to match all flags, such as SYN, ACK and FIN, and may be used to detect the start and the end of TCP connections. Thus, we utilized a total of 13 "match" fields on the flowtables. We also utilized the "Copy-Field action" feature to implement splicing and TCP sequence number rewriting when our switch does the proxying/pinhole routing for authenticated TCP connections. We note that our scheme mainly focuses on TCP in the protocol match (i.e., OFPXMT_OFB_IP_PROTO) as it shows that 99.91% of traffic in data center networks is TCP.

The OpenFlow specification defines the common structure of experimental match fields, messages and actions, then each vendor can customize the format of each structure. We implemented a new vendor type OFPT EXPERIMENTER message and encapsulate custom VARMAN management messages in standard OpenFlow channel from controller to dataplane switches.

3.3.2. Key modules overview

Switch monitor: This module monitors the switch counters, manages the switch hardware ports and configurations, static thresholds, alarms and meters the traffic at the packet level. This mechanism provides an interface through a secure channel (that supports NETCONF, SNMP, Openflow) for the orchestrator and controller for overall framework management.

Flow monitor: Using a novel Flow-Aware sampling method for behavioral threat analysis, we embed the monitoring functions in the dataplane switches. This module classifies short-lived/long-lived malign/benign flows and automatically detect pre-cursors to impending attacks. The sampling rate, packet headers and action-handler (NF) at the switch OpenFlow flow-table can be dynamically updated through the flow-rules ie. Match-action-priority-timeout. This unit extracts network metadata from the transport layer and application headers of incoming packets, take actions based on the extended Match field composed of network events, besides the original OpenFlow Match field.

Feature management: This module collects the switch port/counter values and flow-table entries from TCAM (switch memory) to elucidate the feature values. It generates a consolidated synapse (feature-set) that comprises of meta-data and features/labeled specific for two-stage detection – (i) dataplane anomaly detection in switches and (ii) ML-based Classifier/Analytics pipeline in the controller.

Anti-spoofing: To avoid false-positives when legitimate traffic is marked suspicious, we added a validation logic in the dataplane called '*anti-spoofing*'. This module utilizes conventional NIDS techniques such as *Challenge/Response*, *TTL-Correlation* methods and redesigned the logic to be lightweight for the SDN framework. Hence our attack-detection scheme is different from other methods as it combines anomaly-detection and heuristic behavioral methods to validate the risk of an impending DDoS attack, before triggering the controller. The common attack vector for DoS/distributed botnet attacks is to exploit vulnerabilities in the lower layers of the networking stack (TCP/UDP/IP/ICMP/ARP). These transport layer attacks eventually lead to higher level network protocol services. In SDN networks, such attacks target controllers and *Openflow* channel saturation.

Switch manager: This module monitors the statistics of network events and decides the paths of flows through the dataplane pipeline. Based on the policies installed by the controller, the traffic will be either directed through the I stage response or for further deep-packet-inspection through the Network Function pipeline. The Switch Module can also modify locally the monitoring policies due to the changes in network state. If the table-miss policy is matched, the module stores the packet into a buffer and queries the controller for the policies to enforce and is responsible for forwarding the flows/packets with flagged-network events/alert-messages on the OpenFlow channel to the controller.

3.3.3. Attack prediction

We used a light-weight fine-grained flow monitoring module that feeds the real-time data into an attack prediction module and flag some flows as suspicious or symptoms of impending attacks. Classical time-series models use a time domain method to predict future value from some combination of the past values, usually with a focus on reducing the systematic error in the model to white noise. Some of the standard time-series forecasting methods are moving average

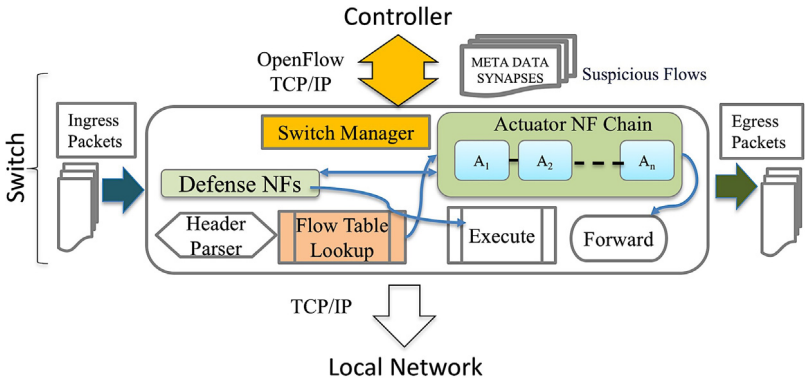

Fig. 4. VARMAN switching pipeline.

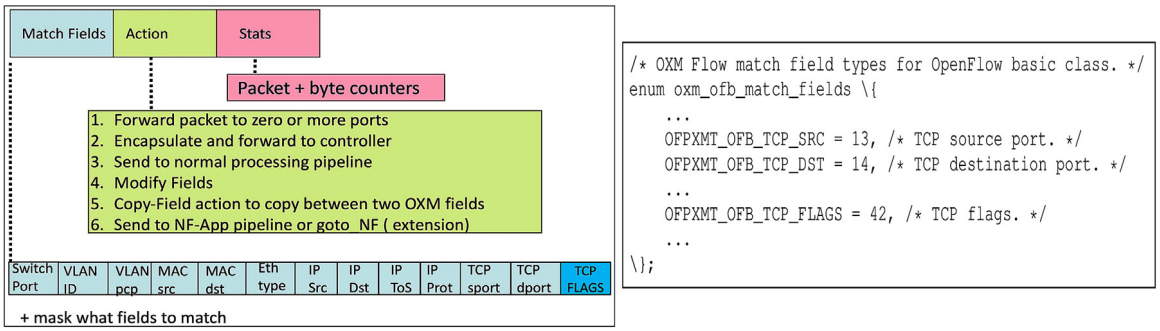

Fig. 5. OpenFlow 1.5.1 flow table extensions.

(MA), Simple Exponential smoothing (SES), exponential moving average (EMA), extrapolation, auto-regressive moving average (ARMA), etc. These are able to forecast intrusions after observing significant changes in traffic volume while attacks are launched. According to our literature analysis, we found SES as an efficient prediction technique for NIDS when compared to other methods. Hence, we utilize an SES based prediction algorithm, leveraging historical state of specific flows and predict the ranges.

When the real-time switch counter values arrive, based on the deviation from the predicted range we mark the flows as benign or suspicious or malign. Application Layer Flooding attacks, targeting higher layers of protocol stack such as DNS/NTP/SNMP amplification and HTTP attacks are very common at present and such attack traffic pattern resembles "flash crowd traffic". As both the Application-level/flashcrowd attacks, can generate legitimate packets in bursts of short intervals, it is not possible to always avoid true/false-positives in detection. These higher-layer/ application level attacks do not usually violate the protocol semantics and so, the protocol metadata/headers encapsulation look normal. However, the attacks operate covertly to exploit the application function vulnerabilities, making the single-feature based detection technique almost impossible to filter those packets. In our scheme, we detect the application layer attack traffic patterns by selecting bilateral flows and feature set for analysis. In unilateral flow, we can count only one side of conversation Request/Response packets. In bilateral flows, we match the Request/Response packets from both sides of the conversations and detect the malicious behavior in the protocol semantics. Using the multiple detection and classification mechanisms across all four planes in our framework, we were able to flag and accurately converge on to a particular class of application attack and take remediation actions. The features values and information are extracted from the switch counters, as flows enter the OpenFlow switch. And we calculate the features as per the formulation given in the Algorithm 1. The pseudocode of this algorithm is shown in Fig. 6. We have adopted the bio-inspired algorithmic approach from ForChaos [40] to construct our attack prediction technique in the NIDS dataplane to detect common network centric and application layer attacks.

3.3.4. Anomaly detection with protocol triggers

In SDN enabled data centers, the DDoS/botnet flooding attacks are detected by periodically polling the flow-tables by a flow-analyzer application on the control plane. This polling-based mechanism causes more monitoring traffic on the OpenFlow channel between control-plane and dataplane, sometimes causing overload during attacks. We propose to use the SDN native protocol monitoring mechanism built around the OpenFlow protocol behavioral analysis. In OpenFlow protocol, an exception message type called "PACKET_IN" is defined for the switches to communicate to the controller whenever the 'matching' action is not specified in its flow-table for a particular flow. The implication is that switches have to forward every *newflow* packet ("flow-table lookup *miss*") to the controller, thus creating a *newflow burst* in the southbound control channel (OpenFlow), consequently overloading the network port and CPU on the controller. The *newflow* attack floods in the SDN network which triggers '*OpenFlow: Packet_IN*' on the dataplane and elucidates '*OpenFlow: Flow-MOD*' from the controller. This *newflow* burst increases the OpenFlow traffic to the controller, creating a *PACKET-IN*/ FLOW_MOD overload on the controller, and thus saturating the control channel. Further, this additional flow processing in the OpenFlow pipeline incurs additional latency, wait cycles for the flow-table update, forwarding delays, and out-of-order packet delivery, connection timeouts, inconsistencies. The pseudocode of the OpenFlow Protocol anomaly detection algorithm is listed in Fig. 7.

We designed an anomaly detection method for SDN OpenFlow protocol messages and deployed an OF-Trigger mechanism in the switches that can detect an abnormal burst of any special/exceptional messages. The *newflow* detection mechanism monitors multiple aspects of the OpenFlow protocol messages, flow-table/counter entries and generates trigger/alerts when any one or more of the monitored parameters exceed the thresholds. Once the protocol anomaly is confirmed as an attack, then those packets that caused the attack are dropped in the

Algorithm # 1 - Attack Prediction at Data Plane Switch

Objective

To detect or predict network Attacks from reading the switch counters

Method

Calculating the following statistical parameters for both scenarios (i.e., anomaly and normal traffic), just by reading the values of the counters in switches and comparing with historical data:

Definitions

In an observed time series consisting of observations A_1, A_2, A_3. . ., for an event, the SES formula F_{t+1} is defined as the next forecast at time t. A_t is the previous actual observation, F_t is the previous forecast, and α is defined as the smoothing constant. The smoothing constant α can take the values $0 \leqslant \alpha \leqslant 1$. By having the α constant value towards 1, more weight is given to the actual observation. If the α constant value is going towards 0, more weight is given to the past observations. Since we have a set of F features at every time interval t, a prediction for each of the features is made based on history.

Input

f // list of m Features
t // Time Series
n //Number of Vectors in the list
w // sampling Window Size
A_{t-1} // Traffic State

Output

Alert (Attack | Not-Attack)

for t = 1 to n **do**
  for f = 1 to m **do**
    $F_t = a\ A_{t-1} + (1\text{-}a)\ F_{t-1}$
    $e_t = |\ F_i - A_i\ |$
    $e_{total_t} = e_{total_t} + \frac{1}{m}\sum_{i=1}^{n} e_i$
    $\lambda_t = \frac{1}{t}\ln|\frac{\Delta e_{total_t}}{\Delta e_0}|$
    if $\lambda_t > 0$
      then Alert (Attack)
    else
      Alert (non-attack);
  end for
end for

Features

Total number of requests
Total number of packets
Average data rate in Megabits
Average packet size
Average time between two requests
Average time between response and first request
Average time between two responses
Total number of parallel requests
Average byte rate of a flow, switch port
Average packet rate of a flow, port, or switch
Average packet rate asymmetry of pair-flow per port

Average Number of Packets in per Flow(APF)

$$APF = \frac{\left(\sum_{j=1}^{FlowNum} packet\_count_j\right)}{FlowNum}$$

$packet\_count_j$ // number of packets in the flow j within a certain time interval T and $FlowNum$ // number of all packets in the interval.

$$\text{Average Number of Bytes in per Flow(ABF)} = \frac{\left(\sum_{j=1}^{FlowNum} byte\_count_j\right)}{FlowNum}$$

$$\text{Rate of Flow Table Entries(RF)} = \frac{FlowNum}{T}$$

$$\text{Percentage of Pair Flows(PPF)} = \frac{2*PairNum}{T}$$

$PairNum$ is the number of convections.

$$\text{Ports Generating Speed(PGS)} = \frac{PortNum}{T}$$

Fig. 6. Attack prediction in data plane.

Algorithm # 2 - OpenFlow Protocol Trigger based Anomaly Detection

Objective

To detect or predict *newflow* Attack to the SDN, from analysing the protocol messages in the SDN control channel.

Method

Calculating these statistical parameters for both scenarios (i.e., anomaly and normal traffic), just by reading the values of the flow tables (Hardware TCAM) and comparing with historical data collected in each switch:

Definitions

At any $Switch_j$, t is a set of time slots that have equal length:

Total $packet\_in$ messages, PI=$\frac{PI^t}{time}$
Total $matched\_packets$,MP=$\frac{MP^t}{time}$
Total $flow\_removed$ messages ,FR=$\frac{FR^t}{time}$
Total $flow\_mod$ messages ,FM=$\frac{FM^t}{time}$
Average controller_response time, CR=$\frac{CR^t}{PI}$

Initialization

$Sampling\_window$ ←threshold
$PacketIn\_Speed\_threshold$ ← MAX_VALUE
$PacketIn\_count$ ← **0**
$t_i = i^{th}$ time slot
T_{pre} ← Record current time at this instant

Input

PI,MP,FR,FM

Output

Alert (Attack | Not-Attack) or Forward_to_controller()

While 1 **do**
  foreach $t_i \in Switch_j$
  \\confirm abnormal burst of PACKET_IN messages sent from switch
  Increment $PacketIn\_count$++
  if ($PacketIn\_count$ **MOD** $Sampling\_window$) ==0
  then
    T_{now} ← Record the time now
    $\Delta T = T_{now} - T_{pre}$
    $PacketIn\_velocity = \frac{Sampling\_window}{\Delta T}$
  else
    forward_to_controller(PACKET_IN message)
  endif
  if ($PacketIn\_velocity > PacketIn\_Speed\_threshold$)
  then
    Alert (Attack)
  else
    forward_to_controller(PACKET_IN message)
  endif

  \\ Check if the OpenFlow control channel is already choked
  if ($Ave(CR_i) < CR_{t_i}$) then
    Alert (Attack)
  \\ Check if more flow tables are removed than number of matching packets
  if ($MP_{t_i} < FR_{t_i}$) then
    Alert (Attack)
  \\ Check if there is abnormal burst of flow_mod messages from controller
  if $Max(FM_{t_i})$ = True, then
    Alert (Attack)
  end for
end while

Fig. 7. Protocol anomaly detection in data plane.

Table 1
Description of CICIDS2017 dataset traffic classes [41].

Traffic type	Size	Description
Benign	2 358 036	Normal behavior
DoS Hulk	231 073	The attacker employs the HULK tool to carry out a denial of service attack on a web server through generating volumes of obfuscated traffic.
Port Scan	158 930	The attacker tries to gather information related to the victim machine.
DDoS	41 835	The attacker uses multiple machines that operate together to attack one victim machines
DoS GoldenEye	10 293	The attacker uses the GoldenEye tool to perform denial of service attack
FTP Patator	7 938	The attacker uses FTP Patator in an attempt to perform a brute force attack to guess the FTP login password
SSH Patator	5 897	The attacker uses SSH Patator in an attempt to perform a brute force attack to guess the SSH login password
DoS Slow Loris	5 796	The attacker uses the Slow Loris tool to perform denial of service attack
DoS Slow HTTP Test	5 499	The attacker exploits the HTTP Get request to exceed the number of HTTP connection allowed on a server, preventing other clients from accessing and giving the attacker the opportunity to open multiple HTTP connections to the same server
Botnet	1 966	The attacker uses trojans to breach the security of several victim machines, taking control of these machines and organizes all machines in the network of Bot that can be exploited and managed remotely by the attacker
Web attack: Brute force	1 507	The attacker tries to obtain privilege information such as password and Personal Identification Number (PIN) using trial-and-error
Web attack: XSS	625	The attacker injects into otherwise benign and trusted website using a web application that sends malicious scripts
Infiltration	36	The attacker uses infiltration methods and tools to infiltrate and gain full unauthorized access to the networked system data
Web attack: SQL injection	21	SQL injection is a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution
Heartbleed	11	The attacker exploits the OpenSSL protocol to insert malicious information into OpenSSL memory, enabling the attacker with unauthorized access to valuable information

dataplane switches and also it prevents future attacks. Hence in the data path, VARMAN SDN incurs only a small *processing overhead* for the *newflow* (first packet in a flow) and it has lower latency as it keeps the packets in the data plane and operates at wire speed.

3.4. SDN control plane

As a centralized and often high-performance platform, the control plane holds advantages of abundant computing resources and holistic info of the whole network. Thus, on the control plane, key functionalities are developed: DDoS attack classification attack tracking and dynamic defense. All of them are essential for attack reaction as they determine which actuator is to be deployed and on which data plane switch it is to be deployed. According to our overall objective, the heart of the control plane is its intelligence ability to inspect current DDoS attack and reason proper strategies, which means the control plane should be able to (1) classify DDoS attacks and (2) track the attack and (3) make a proper strategy to defend it. DDoS attack traffic is firstly captured by data plane sensors, the Suspicious traffic matching a specific rule is mirrored (by sampling) for traffic feature extraction. The dataplane switches have a mechanism to extract the feature-set from the counters and flow-table in TCAM and forwarded to the controller ML-pipeline for classification. On the controller, a fine-grained classification pipeline leverages this feature-set as input to identify/map the attack type. To guarantee the accuracy and reduce the false-positive rate during classification, an ensemble of advanced machine learning algorithms and organically generated datasets are utilized in this module. The attack Detection Engine uses a Machine Learning (ML) technique to classify the traffic. This one is specifically trained to detect the presence of one or more components of a botnet in the network. Once the particular attack type is identified, the controller invokes the defense actuator library to implement specific defense actuators on the switches that are close to attack. Last but not least, the loaded actuator will be executed on the specific switch, defending against certain DDoS/Botnet attacks in the first place. After the attack is eliminated, the defense actuators used for defense will be removed from certain switches.

3.4.1. Key modules overview

Flow table manager: This is responsible for creating an appropriate monitoring or remediation rule in the flow-table of the virtual switches whenever a flow is detected by the DDoS/Bot Detection Engine to be malicious and involved in a Botnet attack. Such rule blocks the infected host that is identified by its mac address. The installation phase of the blocking rule for a flow considered malicious proceeds in two steps: (i) the mac-address of the infected host is recovered from the flow ID; (ii) drop rule has to be installed in the flow-entry corresponding to the involved virtual switch in the detected malicious traffic.

Dynamic defense: After the attacks are identified, the control plane makes a set of strategies to react. Enabling defense actuators on the data plane dynamically is a key step for executing these defense strategies. When attack types and source locations of a DDoS attack are both determined/traced back, the control plane needs to perform a highly automatic defense reaction immediately. In VARMAN, the control plane reacts by loading specific defense actuators onto directed data plane devices. A set of pre-compiled software modules implementing special NFs related to security monitoring or attack mitigations and security applications are maintained in this dynamically loadable library. This module orchestrates an optimal service-chaining strategy (with NFs) and push it to the dataplane switches for execution and provides an API to administrators for deploying any custom mitigation/QoS specific strategy.

Controller: This module is the central controlling authority for the policies enforced in the dataplane switches and maintains network states according to the network events, statistics, and flows from the data plane. This will handle the OpenFlow protocol normal messages and also special alert messages from the dataplane switches. The controller initializes the flow-tables and policies when the switches across the dataplane attach through the OpenFlow control channel and manage the complete security processing pipeline, classification and II stage attack response actions.

3.4.2. Datasets Specification

Datasets are generated by aggregating the samples from the real networks or simulation environments, that are processed by labeling, the addition of metadata, etc. and made available for researchers to build train the pattern and anomaly detection systems. We have processed and utilized both NSL-KDD [11] (for comparison with related works) and CICIDS2017 [12] (to be in parity with the latest) datasets in our system. NSL-KDD is one of the popular datasets still utilized by many researchers, but in our analysis that dataset does not simulate the traffic condition and security incidents that are happening in today's networks. Hence, we made a further survey and selected the most recent CICIDS2017 datasets (published by "Information Security Center of Excellence (ISCX) of the University of New Brunswick, Canada"). It is one of the unique datasets that include up-to-date attacks and was selected as the most comprehensive IDS benchmark to test and validate the proposed ideas. CICIDS2017 was collected based on real

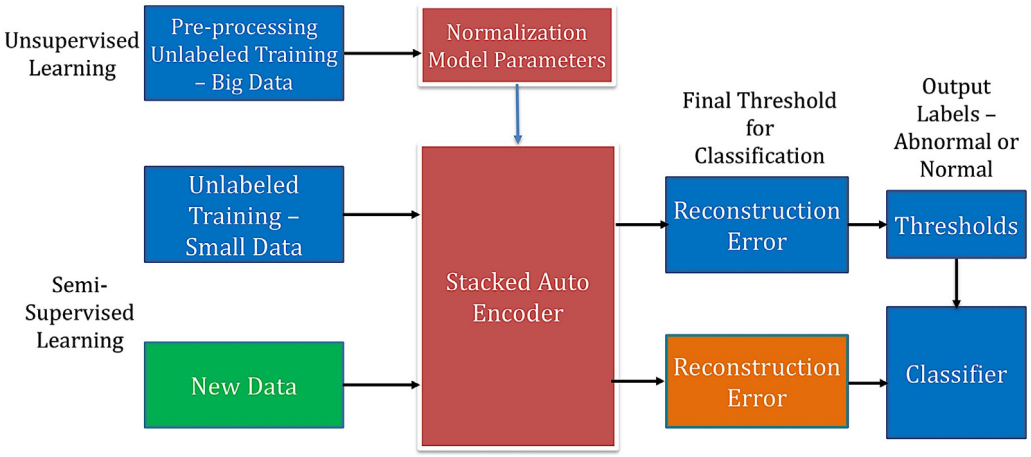

Fig. 8. ML workflow pipeline.

traces of benign and malicious activities of the network traffic. The dataset comprises 2,830,108 records, of which 2,358,036 are benign (83.3%) and 471,454 are malign (16.7%) and represents common attack families/scenarios. Table 1 highlights the characteristics and description of the traffic classes in the CICIDS2017 dataset.

The researchers had used statistical metrics to encapsulate the "*network events*" into features which include.: (1) packet size distribution. (2) packets per flow. (3) payload size. (4) protocol request time distribution. (5) certain payload patterns. CICIDS2017 is a labeled dataset with a total number of 84 features including the last column corresponding to the traffic status (class label). The features were extracted into a CSV file that includes: Flow ID (1), Source IP (2) and Destination IP (4), Timestamp (7) and Label (84). We refined these datasets further, to extract a normalized dataset, i.e., containing a comparable number of normal traffic and malware samples. We further augmented the training dataset based on attacks in SDN, also replaying the DDoS attacks from packet traces to generate SDN *PACKET_IN* packet streams. Consequently, our paper evaluates the performance of machine learning algorithms deployed in our NIDS in realistic networks. Our dataset preprocessing is designed to select the most optimal feature-set to detect the behavior of malware or DDoS attack, that can be gathered from the SDN enabled network.

3.4.3. Feature selection

Feature selection aims to devise the minimal subset of flow features that are enough to detect the behavior of a botnet. This is especially useful for machine learning based behavioral analysis, where too many (useless) features to take into account may confuse the classifier. In this paper, we focused on selected features, taking into account the ones more suitable for DDoS/botnet detection that can be actually collected in a SDN.

In a SDN environment, only statistics like the number of packets, the number of bytes, and the flow duration can be read from the controller (control plane element), which obtains it through the OpenFlow calls to the data plane. However, other useful features can be manually computed from the statistics exposed by the controller. To explain this method, we have listed some representative features from the dataset (Table 2a) that can be directly obtained from the SDN controller through API queries. Table 2b shows the new features we derived in the SDN enabled environment and the corresponding mapping to the features from the dataset. Having direction-specific features is useful to detect particular botnets since, for example, most of spam botnets generate unidirectional flows. Moreover, we consider the minimum, maximum, average and standard deviation of such values. Indeed, bot traffic tends to be uniformly distributed over time but, as a bot progresses through its life-cylce and accomplishes different tasks, the characteristics of the generated traffic varies. The use of such statistic measures helps to capture the behavioral differences between these tasks, so that the overall bot behavior can be detected.

Table 2
(a) Traffic features extracted from SDN controller (b) Derived traffic features.

#	Feature	SDN feature	CICIDS2017 dataset
1	Length of the connection	Duration	Flow duration
2	Maximum expire time of flow	Hard time out	Flow use time
3	Protocol type	Protocol	Protocol
4	Data bytes in bidirectional flow	Bytes count	Bytes
5	Packets in bidirectional flow	Packets	Packets
6	Data bytes from source to destination	Tx packets	Src2dst packets
7	Data bytes from destination to source	Rx packets	dst2src packets
8	Flow permanence time	Idle time out	Flow idle time

#	New feature	CICIDS2017 feature
1	Packet per second from source to destination (PPS)	Packet rate (src2dst)
2	Packet per second from destination to source	Packet rate (dst2src)
3	Inter arrival time (IAT) (min, avg, max, std)	Inter time
4	Inter arrival time from source to destination (min, avg, max, std)	Inter time (src2dst)
5	Inter arrival time from destination to source (min, avg, max, std)	Inter time (dst2src)

3.4.4. ML workflow

The key function blocks in the ML pipeline (Fig. 8) are described here.

Preprocessing: In this step, a preprocessing function is applied to the CICIDS2017 dataset by mapping the IP (Internet Protocol) address to an integer representation. The mapped IP includes the Source IP Address (Src IP) as well as the Destination IP Address (Dst IP). These two are converted to an integer number representation.

Unity-based normalization: In this step, we use the equation give below, to re-scale the features in the dataset based on the minimum and maximum values of each feature. Some features in the original dataset vary between [0,1] while other features vary between [0, ∞]. Therefore, these features are normalized to restrict the range of the values between 0 and 1, which are then processed by the auto-encoder for feature reduction.

$x_i = \frac{(x_i - x_{min})}{x_{max} - x_{min}}$. x_i is feature's value. x_{min} is minimum value. x_{max} is maximum value.

Feature selection and dimensionality reduction: In this step, Information gain of attributes is used to select the important features from the dataset, with the following goals: (i) Reduce dimensions of the dataset (ii) optimize computation and memory usage (iii) improve quality.

We adopted a non-symmetric deep auto-encoder (NDAE) algorithm [42], that has an improved pipeline. There is a fundamental change from a conventional pipeline that is symmetrically stacked

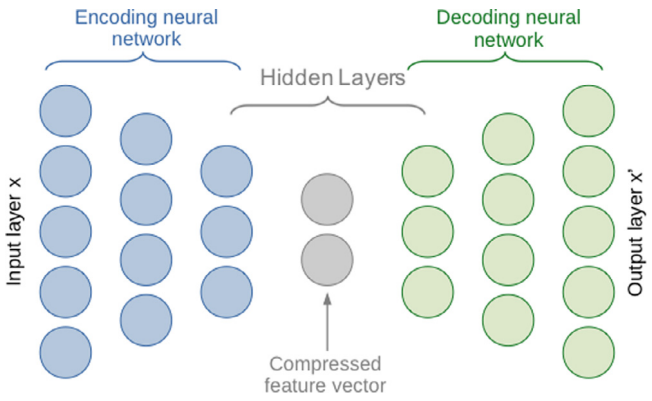

Fig. 9. Conventional symmetric stacked autoencoder.

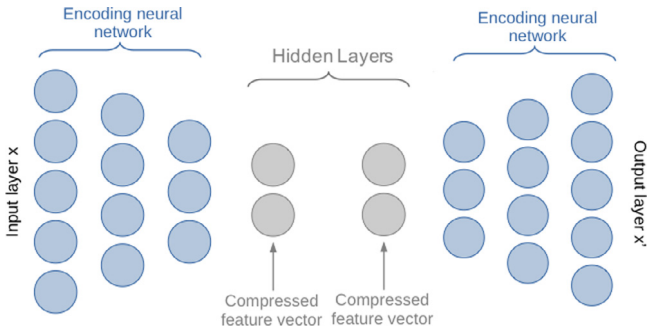

Fig. 10. Non-symmetric stacked autoencoder.

(Fig. 9) to how we have designed the pipeline (Fig. 10). The authors [42] who initially proposed this approach, have proved the rationale for this unconventional design by presenting results of typical network attack classification from popular datasets and this NDAE system utilized less CPU and training time, with higher accuracy. After careful analysis and experimentation with this new autoencoder model, we further improvised with the composition of the hidden-layers, neurons, and feature-set from the more recent dataset for training.

Classification: For shallow learning, we use the Random Forest (RF) as the output classifier. The RF classifier prevents overfit issues and makes the final decision using "majority voting". We improvised the algorithm by optimal input feature-set from the NDAE stack and eventually classifying the network flows into multiple types.

3.4.5. Attack detection

On the controller, there runs a DDoS attack classification module that leverages the extracted traffic features as input to verify the attack type. To guarantee accuracy and reduce the false-positive rate during classification, a machine learning method is utilized in this module. Our classification system uses a hybrid ML-scheme with deep and shallow learning techniques and achieves optimal performance in the context of NIDS. Further, we have made an empirical study to tune some of the hyper-parameters which include — "count of layers, number of neurons per layer, learning rate of the model, regularizers", and others. We selected a subset of 50 and 40 features in the case of CICIDS2017 and NSL-KDD datasets respectively and normalized the values between 0.0 and 1.0.

In the DL workflow, the input feature sets (traffic samples) are reduced by the Autoencoder (NDAE) stack and the resulting encoded representations (features) are fed to the output RF module which learns and classifies the traffic into multiple attack types. After experimenting with various compositions (i.e. "numbers of neurons and hidden layers") we arrived at an optimal model for NIDS application, also factoring in risks of overfitting. Fig. 11 shows the model and ML pipeline in the VARMAN control plane. The algorithms implemented in the ML pipeline are illustrated in Fig. 12.

3.5. Orchestration plane

The main role of this global layer that spans across the entire network, for multi-controller SDN datacenters, interconnected multiple domains (e.g. SD multi-Clouds) will be to provide monitoring and security at each layer and interface, and exchange security and threat intelligence. Our threat analytics engines analyze multiple pieces of information at the network, switching, control plane, protocol, application layers and identify/block even the most sophisticated attacks. The Orchestrator layer monitors the network quality parameters through controller response and overload indicators, switch management units, inform the switches connected to re-discover new routes, dynamically re-deploy network function service-chain and thus orchestrating the network.

Fig. 13 illustrates a large multi-controller/domain SDN network and some key functions are discussed in this section.

3.5.1. Topology management

A "bidirectional selection" scheme is used for creating the initial network topology i.e. "controller-to-switch mapping". The neighborhood and static mapping between any two controllers and switch may be pre-configured by the administrator. When the network is activated, at run time this mapping is managed and dynamically orchestrated by 'flow-table' rule updates from the orchestrator to the corresponding controllers and switches in the network. The protocols such as LLDP also feed real-time information about mapping, discover changes in routing, topologies, and routes traversing multiple controller domains. I.e. Flow rule to forward flows between two or more switches connected to different controllers. In the case of satisfying the global policy and QoS, the controllers and switches maintain the preference list based on control channel (OpenFlow) metrics such as distance, response time, speed, bandwidth, ports.

3.5.2. Control plane monitoring

Generally, to scale and load balance a large domain, multiple controllers are deployed. However, such multiple-controller topology opens up a larger attack surface to breach into the network for DDoS and botnets. The controllers get overloaded processing the flood of malign flows and through one vulnerable controller, the entire network may be compromised due to lateral movement of the malware. So, it is critical to detect the victim controller and devise a dynamic remediation scheme. Fig. 13 shows a multi-controller SDN, in which the switches will sense 'congestion or attack' to the controller from other switches in the data plane, so the 'controller-to-switch' response time and 'control channel' bandwidth is an indicator for attack prediction. (all suspicious packets dropped proactively and other microflows are forwarded). This module is aimed at identifying overloaded controllers and the switches in the attack paths. Dynamically modify the network routes of the paths. Switches will be installed with new flow tables, with modified actions or different forwarding ports, alternate controller or control path.

3.5.3. Data plane switch monitoring

On each switch, a switch monitoring mechanism (SMU) is deployed to provide access and interface for gathering the counter statistics. On this orchestration plane, we deploy the SMU manager that connects with switches to detect the attacked switch and take action to remediate. The SLMU manager gathers statistics such as: *packet_in, matched_packets, flow_removal* and *switch_load*.

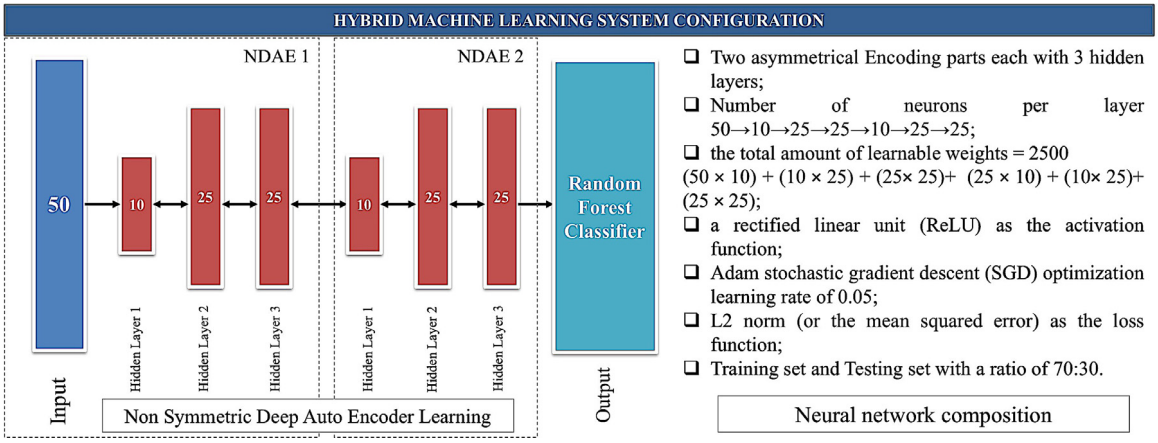

Fig. 11. Hybrid ML system composition.

Hybrid Machine Learning Pipeline at Control Plane

Algorithm # Stacked Autoencoder (NDAE)

Input

Weight matrix of the hidden layer: U
Bias vector of the hidden layer: $b1$
Training dataset e
Input vector $x \in R^d$

for all number of training iterations **do**

Train the single layer autoencoder
using back-propagation:
Map it to the latent representations $h_i \in R^{d_i}$
(here d represents the dimension of the vector)
$h_i = \sigma(W_i . h_{i-1} + b_i)$; $i = \overline{1, n}$,
$h_0 = x$, σ is an activation function,
sigmoid function $\sigma(t) = 1/(1 + e^{-t})$,
n is the number of hidden layers.
Calculate output vector $y = \sigma(W_{n+1} . h_n + b_{n+1})$
Estimator of the model $\theta = (W_i, b_i)$ can be obtained by:
Minimise the square reconstruction error over
m training samples $\left(x^{(i)}, y^{(i)}\right)_{i=1}^{m}$,
Calculate $E(\theta) = \sum_{i=1}^{m} \left(x^{(i)}, y^{(i)}\right)^2$

end for

Algorithm # Random Forest Classifier (RF)

Input

f ⊆ F, F is the set of features.,
Number of Trees in Forest B
Training dataset S =(x_1, y_1),… (x_n, y_n)

function RANDOMFOREST(S,F)

$H = \emptyset$
for $i \in 1, \ldots . B$ **do**
$S^{(i)}$ = A bootstrap sample from S
h_i= RANDOMIZEDTREELEARN ($S^{(i)}$, F)
$H = H \cup \{h_i\}$
end for
return ***H***

end function

function RANDOMIZEDTREELEARN (S, F)

At each node:
f = very small subset of F
Split on best feature in f
return The learned tree

end function

Fig. 12. Hybrid ML algorithms.

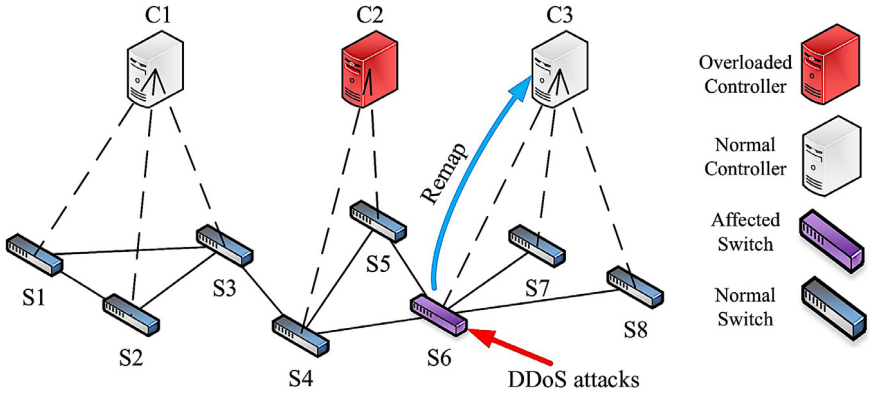

Fig. 13. Multi-controller remediation scheme.

3.5.4. Flow reputation engine

The flows that pass through the VARMAN framework are given a score by this engine which calculates correlation with attack-mitigation historical data and a heuristic with other metrics. It periodically polls all the controllers in the network for interesting flows and connects to switches in the dataplane for flow-table synapses, summary statistics of OpenFlow pipeline. Based on all these factors, the reputation score recalculated for each flow and flow-table rules with (controllerID,switch#, flowID) priority and new action-set is assigned for processing matching packets in that flow.

3.5.5. Attack remediation

The dataplane switches forward certain flows as suspicious and the orchestrator plane installs static flow-rules (controllerID, switch#, flowID) to the switches in the path of those suspicious flows to forward the packets to a special honeynet/honeypot system. Also, in the multi-controller SDN domain, the benign flows can be remapped to an alternate controller through different switches by installing new flow-rule (controllerID, switch#, flowID). The deception techniques allow the VARMAN ML pipeline to do behavioral analysis, label the flows and feed the samples to the training system. Since all the activities in the honeypot will be monitored and logged thoroughly, the generated insights can further help fortify the defending system and mechanisms.

4. Experiments and evaluation

We attempted to recreate a large-scale DDoS/botnet attack scenarios and simulation of network malware campaign in this test environment. In this domain of SDN–NFV Integration for practical applications, formal specification or benchmarks are not published in the open and vendors have not published performance or validation results. The security complexities are quite broad in the distributed environment like 'IoT' computing and many properties of the detection and mitigation systems need to be investigated through practical case studies and validated the security schemes with datasets that mimic the threat landscape in the real data centers. So, deriving from various literature study, we designed an evaluation strategy with security perspectives. The key objectives of our evaluation are:

- To detect and defend largely distributed attacks that happen today.
- Responsiveness, with an acceptable performance hit for legitimate users or applications.
- Effectiveness of the anomaly detection, forecasting techniques in the dataplane
- Accuracy of the classifiers and fine-grained detection algorithms
- performance metrics of the choice of datasets and pre-processing
- Improvements in feature-reduction, training resources and time.
- Performance in multi-domain and multi-controller scenarios

4.1. Experimental setup

The evaluation environment is created as a research testbed to experiment with various design choices and reconfigure different network topologies both with legacy networking and SDN components. We replicated the virtualized datacenter infrastructure with a combination of real equipment, emulation, simulated network devices, hosts and also dataset driven evaluations. We simulated a large network topology similar to SDDC that in an SDN/NFV-enabled Cloud computing datacenter and also hierarchical network topology in software network simulator "Mininet", as depicted in Fig. 14. VARMAN framework components are deployed in all layers of the test network, SDN dataplane consists of standard OpenFlow switches running modified "Open vSwitch" and control plane runs enhanced controller software based on RYU. The NFV pipeline in the OF switches are derived from ETSI standard and implemented as dynamically loadable Linux Netfilter modules. The cross-plane co-ordination layer is implemented as a set of service processes on a central server and the SDN application plane consists of some exemplary security applications adhering to the standard northbound Python-based REST/API.

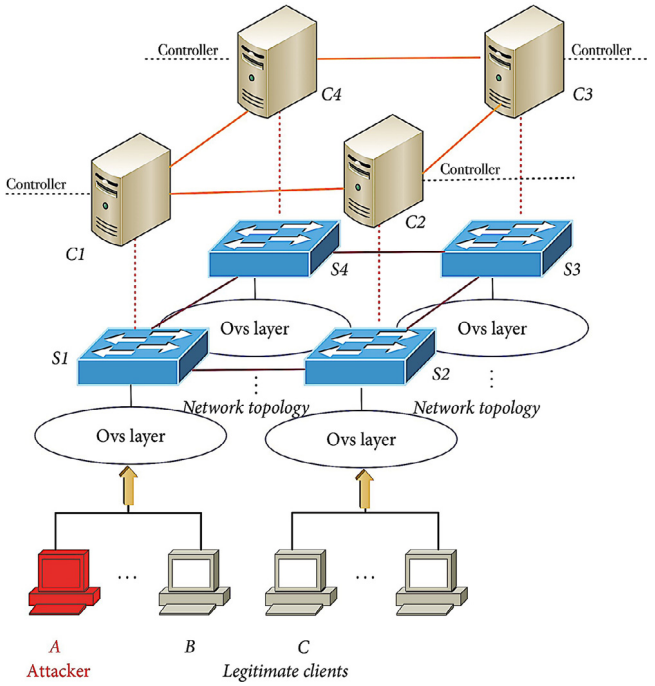

Fig. 14. Testbed network topology.

Table 3
Key metrics.

Metric	Formulation
True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN)	
Precision (number of correct classifications penalized by the number of incorrect classifications)	$\frac{TP}{TP+FP}$
Accuracy (ACC: proportion of the total number of correct classifications)	$\frac{TP+TN}{TP+FP+FN+TN}$
Recall/Detection Rate (DR: number of correct classifications penalized by number of missed	$\frac{TP}{TP+FN}$
False Alarm Rate (FAR: proportion of benign events incorrectly classified as malicious)	$\frac{FP}{TN+FP}$
F-Score (F1: the harmonic mean of precision and recall, derived measurement effectiveness)	$2 * \frac{Precision * Recall}{Precision + Recall}$

4.2. Evaluation metrics

In the field of "pattern recognition", the following terms that relate to classification ("binary or multi-class") are defined to evaluate the overall system performance and efficacy. The formulation of the key metrics is used in all the experiments is given in Table 3. The parameters relevant to these metrics are observed over many cycles, varying datasets and the results are presented in the sub-sections below.

4.3. Experiments

In the following subsections, we present the results from each case study that demonstrates the feasibility of VARMAN solution.

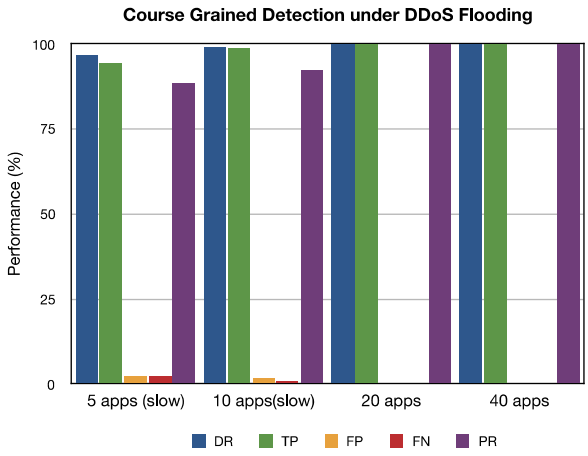

Fig. 15. Attack prediction efficiency.

4.3.1. Dataplane — attack prediction efficiency

In this experiment, we generated various classes of attack traffic to test the detection efficiency of the data plane. The traffic generator from attacker machines flooded the network with packets of different sizes, protocols, varying the inter-packet gap, intensity/rate, varying the ratio of benign/malign traffic and so on. The application traffic simulated client–server protocol behavior (request/response). Experiments are divided according to the type of the attack and the training time. Different types of attacks are constructed — flooding and slow-rate attacks.

- Flooding attacks: the parameters differentiated were the number of applications used and the time the applications was initiated. Two modes of instantiation (i) "constant" in which all of the attackers' applications were initiated at the start. (ii) "increasing", the attackers' applications were gradually initiated. Through this differentiation, we aimed to test if our detection mechanism was able to identify the attack, even when the peak time of the attack was not visible.
- Slow-rate attacks: slow-legitimate connections were integrated in the network traffic as part of its normal behavior. The number of applications was differentiated from 20 to 40. Through slow legitimate connections, we could evaluate if the algorithm is able to identify malicious or normal activity.

The number of threads/application processes per machine were varied gradually in steps of 5 apps, at specific intervals. With this, the experiment validates if the anomaly detection algorithm can differentiate the normal and malware traffic in the midst of mixed traffic conditions.

With reference to the ForChaos algorithm [40], that is used in our VARMAN dataplane detection mechanism, the formulation as : $F_{t+1} = \alpha A_t + (1-\alpha)F_t$, (Defined in Fig. 6) varying the sampling/prediction intervals (window size) and α. The prediction logic is scheduled at periodic intervals based on the traffic conditions and the rate/proportion of detected attacks in the network. Iterating the experiment, we arrived at an optimal interval of 30 seconds and 0.1 for α. The training time was 1200 seconds and attack window lasted for 300 seconds.

Observations from Fig. 15:

- ***Detection Accuracy (DR)*:** The detection rate was higher for DoS attacks compared to slow-speed attack, as the features have a higher variation due to higher proportion of attack-packets in the network. The lower detection rates of 96.8, 99.1 are measured for slow-rate experiments and up to 100% accuracy is measured for fast-flooding experiments (DoS apps).
- Our algorithm did not generate any false positives after the training time in any of the Flooding experiments regardless of the attack time. This is due to the nature of the attack itself. Flooding attacks make "noise" and their behavior is more obvious than the Slow-Rate attacks and that is reflected in the features.
- ***False positives/negatives***: as expected some FP/FNs were measured (1.45% FP/2.23% FN) when exposed to slow-speed and low-proportion traffic in the network. In general, this is one limitation of the statistical method-based detection mechanisms, as the features show fewer variations. The accuracy decreases drastically at lower attack rates because the entropy difference in low volume attacks is almost non-detectable. But in VARMAN, Suspicious flows are not marked based on threshold alone, as we do the Anti-Spoofing/Turing test (Challenge–Response) with the incoming new-flows.
- VARMAN showed consistency and acceptable rate (96.8% to 99.1% for slow-rate and 100% for flooding) at all traffic intensity level (flooding and slow-rate attacks) compared to other traffic-entropy based techniques (which is inefficient at slow-rate attacks) and ML/statistical-detection methods (require more features, monitoring overheads, computational and space complexity, which are not suitable for real-time attack detection) from the literature.

4.3.2. Dataplane — statistical anomaly detection

In this experiment, we present the effectiveness of the statistical detection algorithms based on the features captured from the switch counters in the data plane. We utilize about 16 features and create multi-variate tuple-set and perform anomaly detection based on traffic flow, SDN protocol behaviors and so on. We fire a bunch of apps that generate various types of packet stream (symmetric flows, asymmetric flows, new connection, DoS attacks such as ICMP, SYN flooding and link flood attacks and so on) from various attack machines and normal application traffic from benign user machines in the network. From the graphs depicted here, we can see each graph plots two features elicited from the time-series data. These features are chosen as representative results measured from VARMAN data plane switches. In all the plots, the attacks start at around 9 s into the experiment and our detection mechanism responds immediately, eventually bringing back to normalcy at around 18 s.

Key Observations:

- Fig. 16a: there is a spike in "byte rate" during the attack period (>14k bps), "asymmetric flow" raises exponentially during the attack period than during the normal period.
- Fig. 16b: ratio of "symmetric flows" sharply decreases (<0.2), and newflows (single packet/smaller packet) traffic increases by 90%.
- Fig. 16c: When DoS attack happens in an SDN-enabled network, there is a burst of OpenFlow 'Packet_IN" messages that are forwarded by the dataplane/switches over the control channel to the controller, as these newflows have no flow-rules/route in the OF switches. Consequently, the controller installs flow-rules to the corresponding switches through "*Flow_MOD*" messages over the same southbound/control channel. However, these OF messages is reduced considerably, since the VARMAN dataplane flow-monitoring mechanism detects these OF packet burst and drops these flows at the switch itself, avoiding the control channel saturation.
- Fig. 16d: shows the results when selecting 10–12 features, contributes to maximum detection accuracy in the anomaly detection and prediction algorithms on dataplane. Also, our experiments confirmed that at about 16 features the accuracy tapers down and only these top features are selected to be forwarded for training VARMAN's NDAE ML classifier on control plane as input, which needs to be transformed to less dimensional space. Brief description of these features is provided in Figs. 6 and 7 of Section 3.3.

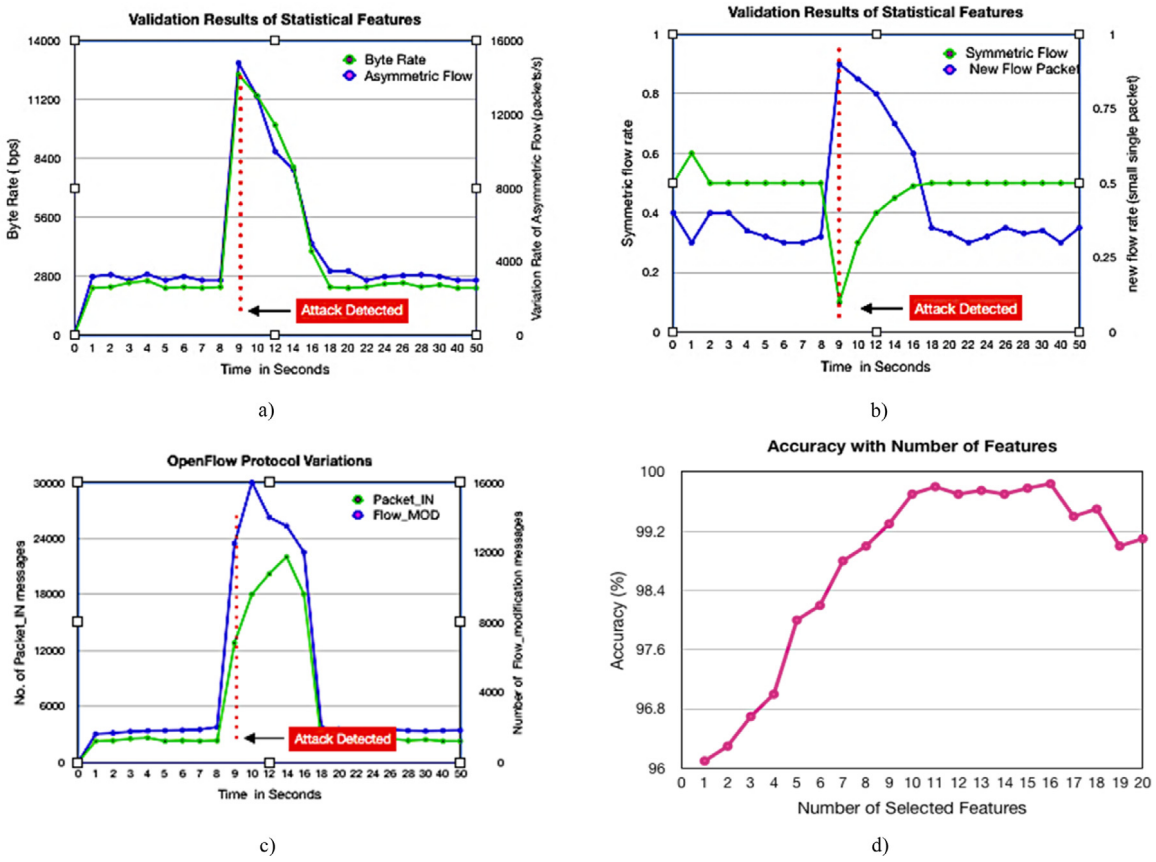

Fig. 16. Feature behavior during attack - (a) Byte rate & asymmetric flow (b) New-flow & symmetric flows (c) OpenFlow Packet_In & Flow_Mod (d) Accuracy variation with feature selection.

Although the complexity of our algorithms is not high, we have experimented with reducing the features to check whether we can achieve the same detection rate. The sub-optimal reduction of features could make the detection algorithm more sensitive towards changes in the network that are not necessarily malicious. We infer that: (a) the features that are monitored are the optimally right ones and capable of predicting the attacks accurately at the dataplane based on coarse-grained security mechanism, (b) the VARMAN dataplane security/remediation scheme detects and drops the "newflow" attack packets and prevents control plane/southbound channel saturation vulnerability in the SDN architecture.

4.3.3. Attack mitigation and remediation performance

VARMAN system has defense schemes to detect various classes of malicious traffic, malware, and network-centric attacks. The cross-plane design intercepts the attack flows at different levels in the pipeline and hence the response times vary for different attack classes. Through this experiment, we present the efficacy of VARMAN in five broad attack scenarios (that cover more than 90% of network attack vectors) and they exercise different modules/mechanisms and NFs in the framework. Of course, VARMAN's defense capabilities go beyond the scenarios summarized here (Table 4), but they are given for illustrating the overall operation under diverse traffic conditions.
Test Cases Description:

- *Scanning*: Nmap and Bonesi tools are used to generate scans from single/multiple machines on the network. This attack is quickly detected at the dataplane (Level 1) with the attack prediction mechanism (Section 3.3.3) and mitigated by dropping the packets in the switches.
- *High-volume DoS*: DoS Flooding with TCP-SYN, ICMP-Echo traffic using "Low Orbit Ion Cannon tool [LOIC]" is generated from a single attacker flooding the victim network, utilizing the entire bandwidth. As the attacker generates requests from different ports, the source port entropy increases during a DoS attack. This attack pattern is easily detected due to volumetric and asymmetric nature of traffic, by the detection mechanism (Section 3.3.3) and attack packets dropped in the switch (level 2).
- *DDoS*: In DDoS attack, different attackers target the same service in the victim, each attacker sends a smaller number of requests to the victim network. But since it is coming from a massive number of attackers, entropy for source-IP would be higher. During attacks, all traffic converge to the same service in the victim network. This attack pattern is marked suspicious at the dataplane (Level 2) due to volumetric and asymmetric nature of traffic and further classified at the control plane DDoS/botnet tracker modules (Section 3.4.1) and remediation actions are installed to the switches either to drop the malicious packets or detoured.
- *Slow-rate*: Slow-Rate attacks exploit the target machines that adhere to network protocol semantics and standards, especially with connection-oriented TCP. The rogue machines exploit this protocol to initiate connections with the peer hosts and disappear. In some cases, the attacker will continue to generate half-open connections to induce the victim servers to maintain connection-state tables and eventually exhaust the resources. These victim machines and the services running on them become unresponsive and crash in most cases. Using the dataplane transport-layer anomaly detection mechanism (Section 3.3.4), intelligent NFs and flow-monitoring rules on the upstream switches, we mark these flows as suspicious and forward summary synapses to the

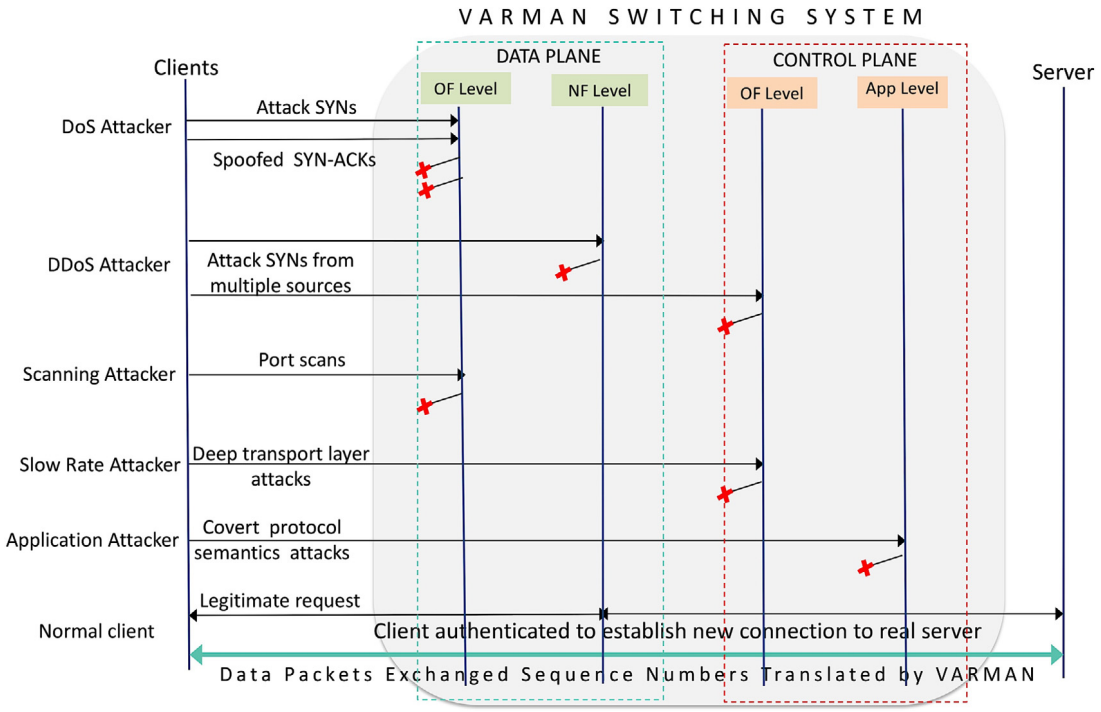

Fig. 17. Sequence diagram of various attack types and defense levels in VARMAN stack.

Table 4
Description of attack scenarios.

Attack type	Attack identification	Fields of interest	Detection method	Protection method
Scanning	Increase in Attacker, Host A, ratio to target addresses	IP Address Port	Level 1 (data-plane) Level 2 (NFV)	Block/Drop
DoS	Volume of traffic flows from/to a single IP exceeds a threshold	IP Address TTL	Level 1 (data-plane) Level 2 (NFV)	Block/Drop
DDoS	Volume of traffic from multiple IPs targeting exceeds a threshold	IP Address TTL	Level 1 (data-plane) Level 2 (NFV) Level 3-(control-plane)	Block/Drop
Slow rate	The slow-rate attack opens a great number of connections and initiates requests that never complete them.	IP Address Port	Level 1 (data-plane) Level 2 (NFV) Level 3-(control-plane)	Block/Drop
App layer	Correlation/asymmetric volume between Request/Response packets	Port Protocol	Level 3- (control-plane) Level 4- (application)	Block/Drop/Remediate

controller. On further analysis at the control plane through the ML pipeline, we classify these slow-rate attacks and mitigate or remediate to a honeypot system.

- *Application layer flooding*: Application Layer Flooding attacks, targeting higher layers of protocol stack such as DNS/NTP/SNMP amplification and HTTP attacks are very common at present and such attack traffic pattern resembles "flash crowd traffic". The distinguishing pattern between Application-level/flash attacks is very bleak due to their short bursts of packets. We used Poisson distribution for the traffic generator, replayed the 'collected traffic trace files' using *fprobe* tool and targeted HTTP application services. The target victim machine is running web services and client machines are simulated with 100 seconds of browser apps spraying all kinds of HTTP/SQL/DNS protocol packets to the webpages on the simulated website. In our scheme, we detect the application layer attack traffic patterns by selecting bilateral flows and feature set for analysis. In bilateral flows, we match the Request/Response packets from both sides of the conversations and detect the malicious behavior in the protocol semantics.

We measure the time period spent by the malicious flows in the VARMAN pipeline, passing through the levels of detection before they

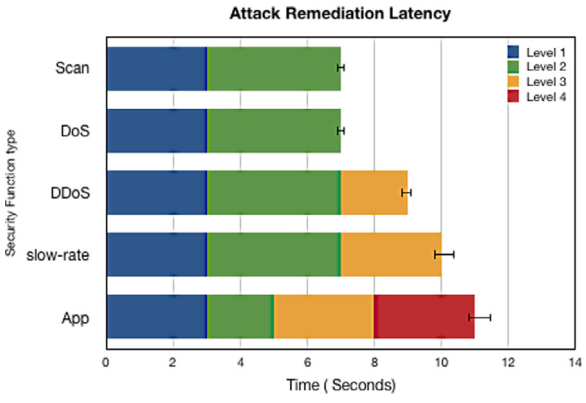

Fig. 18. Detection of various attacks.

are mitigated (Scan, DoS, DDoS, Slow-rate, application-level) as illustrated by Fig. 17.

Fig. 18 illustrates the detection times for each class of attacks and the depth in the VARMAN detection pipeline at which they are detected

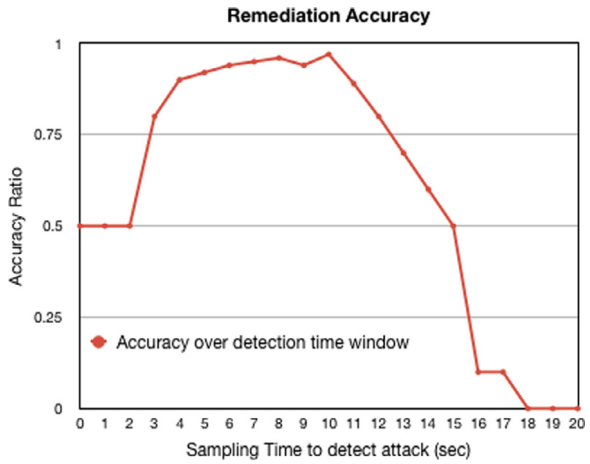

Fig. 19. Detection accuracy with sampling variation.

and mitigated/remediated. Based on the layer (dataplane/controller), level (1–4), and classification overhead (Anomaly detection/ML pipeline), the attack remediation speed varies. The application attacks measured the highest latency to detect and mitigate or remediate, as it incurs overhead across all levels of the analytics pipeline and it takes more time to run the application, protocol correlation/analytics. Most of the entropy-based detection techniques are expected to fail as there is a minimal change in meta-data network protocol headers and pure ML-based classification-techniques achieve just marginal accuracy against application-centric attacks as opposed to transport layer attacks. However, our collaborative cross-plane mechanisms ensure accuracy (Fig. 19).

4.3.4. Controller performance

Network load optimization: (Fig. 20a) We measure the traffic load to the network-port (NIC) on the controller, in two cases — VARMAN SDN and Classic OVS SDN. The *newflow* attack packets are fired in the SDN network which triggers "*OpenFlow: Packet_IN*" on the dataplane and elucidates '*OpenFlow: Flow-MOD'* from the controller. This *newflow* burst increases the OpenFlow traffic (*PACKET-IN)* on the control channel, overloading the controller, and thus saturating the control plane. With our VARMAN stack, we observe that even in the presence of high-intensity attacks, the controller network port's bandwidth is restored to normal, after mitigating the source of attacks. As the attack rate reaches ~2000 pps(packets-per-second), there is a traffic spike in the network due to "Packet-IN" burst, and VARMAN detected the attack, blocked and hence the controller's network port throughput is restored to normal operation. But with Classic OVS SDN, the network load keeps increasing.

Controller CPU overhead: Fig. 20b shows the CPU utilization on the SDN controller, with OVS SDN and VARMAN stack loaded in the test network. As the attack rate reaches ~2000 pps, due to the spike in network traffic (*Packet-INs)* the controller CPU gets busy processing the stream of OpenFlow messages and responding them with *Packet_OUT/Flow_MOD* messages to the southbound dataplane switches. VARMAN SDN detected this flooding attack at the dataplane switch level itself and blocked and hence the volume of OF protocol messages interrupting the SDN controller CPU is much lesser. With just a marginal increase in CPU cycles, the CPU utilization is restored to normal operation. But with OVS SDN, the CPU load on the controller keeps increasing to a saturation point (at 120 seconds), becomes unresponsive and the entire network is paralyzed.

Controller response: (Fig. 20c) We measure the response times/delays from the controller for OpenFlow requests from the dataplane, under different attack rates in the network. As the attack intensity keeps increasing, VARMAN's attack-detection/mitigation mechanisms drop those malicious flows and further some flows are detoured to another controller. This remediation and load-balancing function is executed by the orchestrator layer to prevent control plane failures in the network and optimally utilizing the control plane resources as well. In the case of OVS SDN, it improves due to saving in context switching and packet buffer management, as the datapath is separated. But still, the effect of the attack is pronounced at higher intensities due to *Packet_IN burst* and processing overhead at the controller. The extended dataplane in VARMAN scenario prevents these attack-generated-Packet_IN floods at the dataplane itself and drops those packets cheaply. With the attack traffic eliminated from the control-channel, this enables the SDN stack to process the benign flows with normal overhead and hence the new sessions are established ~2.5 times faster, the response time of VARMAN reduces by about 40% compared to OVS.

4.3.5. Multi-controller remediation

In this experiment, we evaluate the multi-controller remediation performance when the network is under DDoS flooding attack from multiple hosts targeting a single switch-to-controller path. The network topology for this test is illustrated in Fig. 14. The Attacker machine and the attack path through the switch S1 and SDN controller C1 are marked in red.

Test Scenarios: 1. Normal without attack 2. Attacks with No Scheme or defense 3. Attacks with a simple Controller Distribution Scheme (CDS-OVS) which load balances the network workload by distributing the flow-tables to multiple controllers in the network. and 4. Attacks with VARMAN scheme enabled.

Observations: We measured the CPU utilization in all four controllers under these 4 scenarios. (Fig. 20d)

- *NORMAL*: with no attacks, the controllers are processing their own switches workload as configured.
- *NO SCHEME*: attacks are generated from some nodes in the network connected to the switch/controller C1. With no defense capability under this scenario, the C1 controller gets a spike of Openflow packets to 95% load.
- *CDS-OVS*: During attacks, this scheme distributes the workload across the controllers and balances the control plane traffic and so the workload reduces from 95% to 65%, additional 30% on C1 is spread across C2, C3, and C4. But this scheme fails under strict routing flow-rules and redistribution of flows incurs additional latency on the switches and the flows.
- *VARMAN*: the attack detection mechanisms will kick in to drop those attacks packets/flows. Hence the workload is reduced down to 65% with that 5% processing overhead for the controller to run the detection logic. The CPU utilization on the attacked controller C1 also reduced and other controllers in the network are not affected at all.

The key result is that Controller Distribution Scheme is able to bring down the saturation effect on the targeted controller C1 by just re-distributing the workload (which includes the flows from the attacker machine as well), but due to lack of security mechanism to detect malicious flows, it unnecessarily processed the malign flows as well. In the case of VARMAN, due to the presence of attack detection and remediation scheme based on a heuristic algorithm, the malign flows are dropped at the switch S1 and the workload on other controllers are unaffected as well. Due to cross-plane attack classification overhead in the VARMAN pipeline, the controller incurs minimal CPU cycles and we conclude that VARMAN performs better in multi-controller SDN architectures.

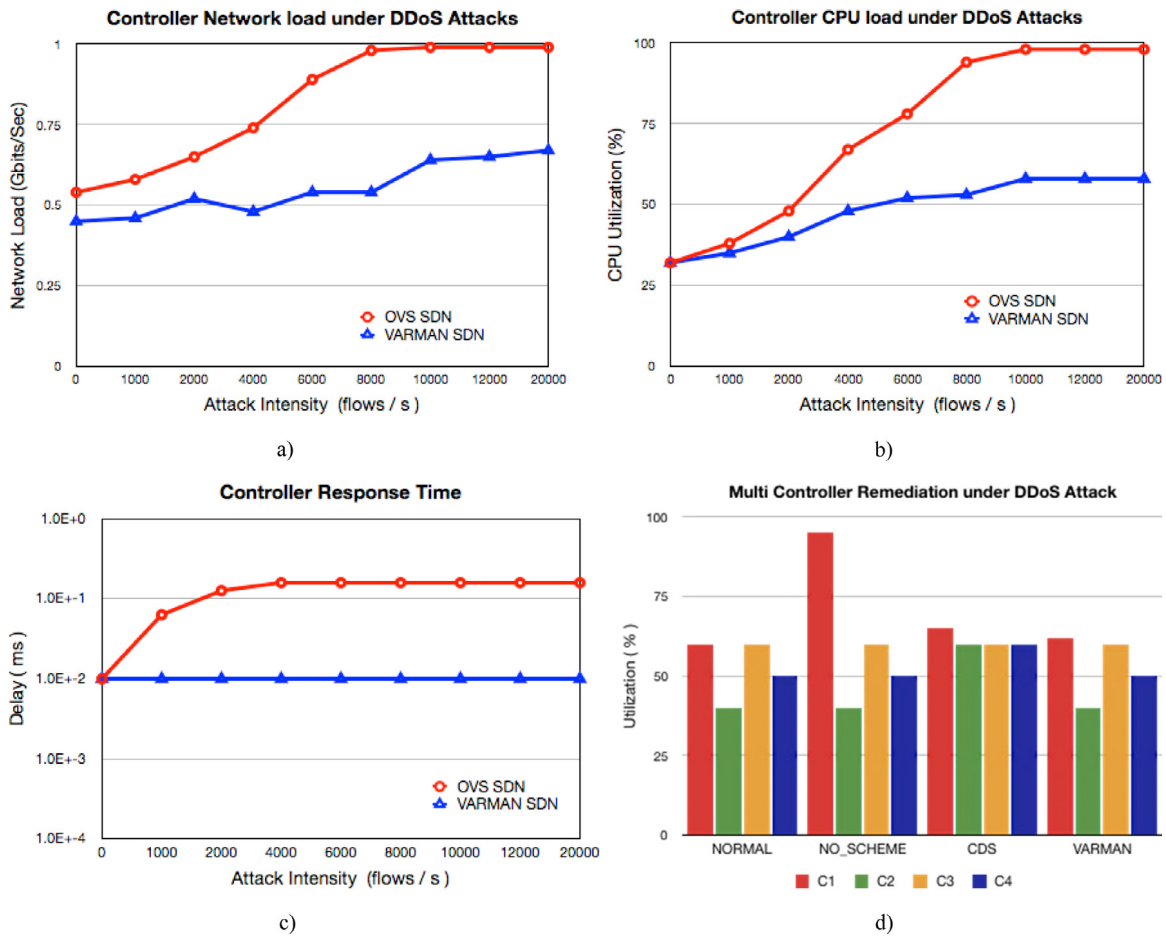

Fig. 20. (a) Network load optimization (b) CPU overload avoidance (c) Response time efficiency (d) Multi-controller remediation.

4.3.6. Hybrid machine learning classification performance (fine grained detection)

In this section, we make comparison with related recent works (using equivalent datasets and utilizing ML-based NIDS) for various attack intensities and training time reduction. We evaluate the classification efficiency of VARMAN control plane ML-based system with another DBN based NIDS system [28] that also uses the classical stacked Autoencoder neural network and unmodified RF classifier. So, we attempted to draw a direct comparison between the VARMAN and DBN model.

Comparison with NSL-KDD and CICIDS2017 datasets: The composition of datasets is selected with a heuristic method, totally 100,000 samples from each of the datasets, with a proportion of 60% benign/40% malign traffic samples. In NSL-KDD (Five types of attacks, 8000 samples/attack-class, and 60 000 benign) and from CICIDS2017 (8 types of attacks, 5000 samples/attack-class and 60000 benign). We ran the datasets, with variations of the number of labeled samples and the results given in Tables 5 and 6, show that semi-supervised VARMAN system performed better than fully-supervised DBN system, with similar proportion and with only 10% labeled samples.

Training Time and Memory Optimizations: Time and space efficiency are key considerations for the classifier in the NIDS. The results from the observations are plotted in the single graph Fig. 21, with y-axes denoting the training time and memory consumption during the controlled training iterations done with both models. The non-symmetric stacked neural network model utilized in our VARMAN system achieves a considerable saving in training time of 94.96%. So, we conclude that VARMAN model maintaining higher accuracy compared to the DBN model, even with different hidden-layer compositions. The heuristic we chose for feature selection in the hybrid-ML training network, has optimized the memory usage for data passing through the VARMAN ML workflow, with an average saving of 90.8% when compared with DBN.

Table 5
Performance with NSL-KDD dataset.

Percentage of labeled samples	Varman Hybrid ML			Ref DBN		
	ACC%	DR	FAR	ACC%	DR	FAR
1%	97.20	0.9520	0.009	95.50	0.9230	0.016
5%	98.80	0.9850	0.008	97.40	0.9720	0.013
10%	99.04	0.9940	0.008	98.02	0.9800	0.016
20%	99.56	0.9960	0.003	98.74	0.9878	0.010
100%	99.60	0.9964	0.004	99.50	0.9945	0.004

Table 6
Performance with CICIDS2017.

Percentage of labeled samples	Varman Hybrid ML			Ref DBN		
	ACC%	DR	FAR	ACC%	DR	FAR
1%	96.68	0.9624	0.030	92.65	0.9235	0.102
5%	98.22	0.9888	0.012	95.35	0.9590	0.061
10%	98.74	0.9800	0.010	97.44	0.9822	0.042
20%	99.02	0.9928	0.009	97.68	0.9896	0.034
100%	99.24	0.9978	0.008	98.24	0.9900	0.021

ML Classification performance: We evaluated our detection algorithms using the standard methods with precision–recall (PR) and receiver operating characteristic (ROC) curves, compared against the DBN model. The comparison of four key indicators of detection features — Precision, Accuracy, Recall, and F1 is shown in Fig. 22. The related results

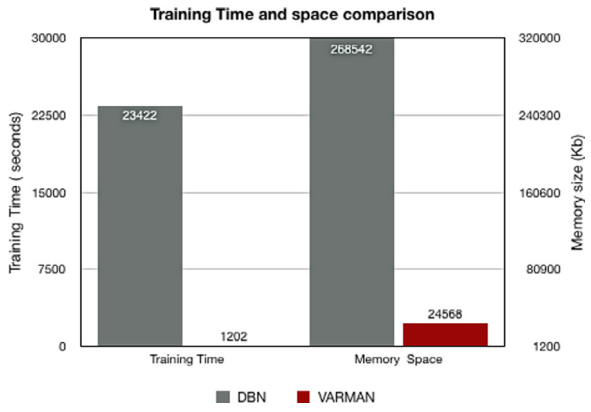

Fig. 21. Training efficiency.

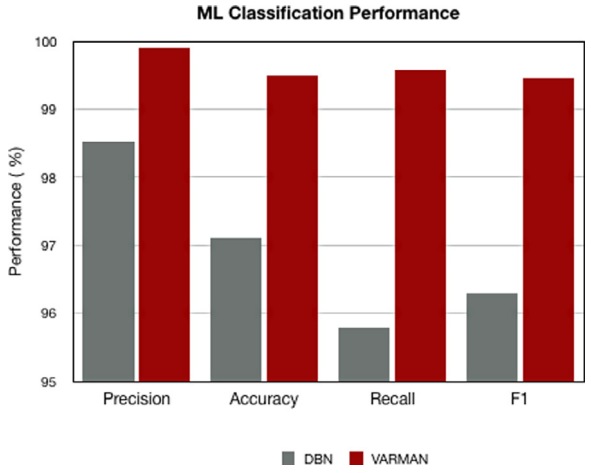

Fig. 22. Classification performance.

Table 7
Performance with generated dataset.

Traffic type	Dataset count	Correct classification	Incorrect classification	Accuracy (%)
Benign	1,672,234	1,624,074	48,160	97.12
HTTP flood	1,828,545	1,777,894	50,651	97.23
Slow Read	456,567	446,933	9 634	97.89
Port Scan	823,456	820,739	2 717	99.67
DoS Flood	2,445,678	2,407,770	37,908	98.45

in terms of ROC is illustrated in Fig. 23 and our proposed model gives superior performance with the highest area under the curve (AUC) value 0.9985 in comparison to DBN model 0.9356. The results show that, when compared to DBN model our hybrid-NDAE classification model in VARMAN, improves on all axes of classification, including the accuracy and recall rate in the simulation environment.

4.3.7. Classification performance of generated dataset

We ran experiments with organically derived datasets by extracting some key features from the CICIDS2017 and traces gathered from hackathon websites and honeypots, recreating the malware traffic in our lab network. Our argument was that with such processed and organic datasets we could train an ML-based NIDS system to be more accurate in real life attack environment and resilient to adversarial attacks. We further evaluated the applicability in real networks, by simulating varied traffic mix in the network which include: "HTTP Flood, Slow Read, Slow Write, Port scan, DoS flood, and DDoS/botnet" in conjunction with benign application traffic. The complete lifecycle

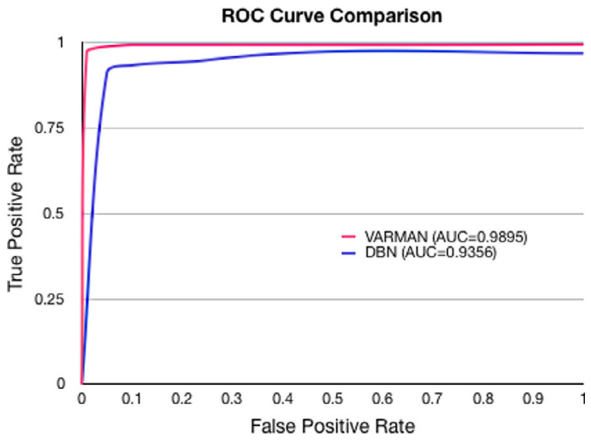

Fig. 23. ROC performance.

of both normal and malicious connections from connection setup to the kill-chain, are captured in trace-files and processed through a data analytics engine such as Splunk, Elasticsearch. We observed similarities in connection behavior/characteristics between the connection setup phase and with slow-rate attack traffic, which might lead to lower detection accuracy. Therefore, in most ML-based NIDS, some of the HTTP slow-rate attacks are misclassified as benign traffic, whereas our NIDS adapts and learns with the traffic intensity, to continuously improve the detection accuracy. Table 7 lists out the results of this experiment, which shows very good classification accuracy and proves the efficacy of the training heuristic we employed in the ML workflow.

5. Discussions and future work

5.1. Evaluation results

In this research work, we conducted experiments under different scenarios, to evaluate the working of our security applications under advanced network attack and large-scale attack scenarios. We measured and compared our architecture over two axes: (i) Efficacy and accuracy coarse-grained detection, fine-grained hybrid ML classification model and light-weight dynamic defense mechanisms and (ii) Performance and processing overhead at multiple levels in our defense pipeline. With a novel scheme in ML feature-reduction, we transformed the CICIDS2017 dataset from 81 features to 10 features while maintaining high accuracy in multi-class and binary class classification using the Random Forest classifier. The evaluation result of the VARMAN as a NIDS solution is quite promising and proved the feasibility of ML-based anomaly detection approaches in the SDN.

We summarize the evaluation of the solutions to the three sub-problems called out in the beginning, namely (a) anomaly detection in data-plane, (b) classification in control-plane and (c) collaborative interface between the layers and downstream services.

(a) Performance of extended data plane architecture: We proposed various extensions to the conventional dataplane including the stateful, security-aware application specific mechanisms, DPI and NFV in the OpenFlow pipeline.

- The notion of a multi-layered dataplane that caches select flow-rules and performs most of the security processing has given tremendous improvement over the conventional SDN scheme. Hence in our data path, we incur smaller *overhead* for *newflows*, as we keep the packets in the dataplane which operates at wire speed. With this key improvement, our SDN achieves significantly higher throughput than classical OVS SDN and other work. This redesign tremendously favors applications comprising short-lived TCP flows and Section 4.3.1 supports this claim.

- Fine-grained distributed network traffic monitoring using SDN is an important capability for effective network management and defense. Using novel Flow-Aware sampling, we embed the light-weight NFs in the dataplane switches.
- Our Feature selection and Trigger mechanisms are devised to extract an optimal subset of counter values/features from the switches, for statistical anomaly detection and classification. The anti-spoofing validation and dynamic flow-reputation scoring mechanisms ensure the dataplane to reduce the misjudgment rate of DDoS attack detection. These optimizations considerably reduced the data flow into the fine-grained ML classification pipeline at the controller. (Section 4.3.2)
- By classifying flows into suspected or normal synapses sets, with multi-stage detection approach, the controller has a relatively lesser workload in runtime. The results from the experiments in Section 4.3.4 proves this claim.
- Algorithm 1 in Section 3.3.3 presented the prediction technique based on the ForChaos [40] with the optimal set of features extracted at real time and this is generally efficient as a black-box deployment in legacy networks. Algorithm 2 in Section 3.3.4 presented the anomaly detection technique monitoring the stats counter values of the various protocol headers, in the packet stream on the switch and this algorithm is specifically designed as a white-box firewall solution in SDN enabled network. So, overall both the algorithms are useful in different deployment and traffic environment, to offer an holistic solution.

(b) Efficiency of fine-grained classification in control plane: The autoencoder-based classification model is not necessarily better than other machine learning approaches, but these results demonstrate the great feasibility of leveraging machine learning hybrid approach, with shallow and deep learning algorithms, to serve VARMAN as a DDoS attack classifier, which is the main purpose of the evaluation above.

- It is observed that our DDoS attack classifier has fairly good accuracy for detecting a single type of DDoS attacks, reaching about 96%. The detection accuracy for mixed attacks is lower but still reaches around 83%. More specifically, we demonstrate the precision, recall and f- measure for various classes of traffic in Section 4.3.7. Except for the mixed traffic of SYN and UDP flood as well as SYN and ICMP flood, the 3 parameters for classification of all type of traffic reach above 90%, which is quite acceptable in classifying DDoS attacks in the real network.
- The results of our Autoencoder (AE) dimensionality reduction approach are given in Section 4.3.6, the observed accuracy for Random Forest is significantly improved compared to LDA, QDA and the Bayesian Network Classifiers.
- Furthermore, what stands out in our experiments with various ML algorithms, is the increase of the resulting accuracy for NDAE approach for the reduced dimensionality from 81 to 10 features. Unlike features selection techniques where the set of features made by feature selection is a subset of the original set of features that can be identified precisely, AE generated new features pattern with reduced dimensions. Here, the stacked AE reconstructed a new and reduced feature representation pattern that reflects the original data with minimum error. As exemplified by the obtained results, our ML approach is able to preserve important information in CICIDS2017, while efficiently reducing the features dimensions in the used dataset, as well as presenting a reasonable visualization model of the data. Features such as Subflow Fwd Bytes, Flow Duration, Flow Inter arrival time (IAT), Packet per second from source to destination (PPS), PSH Flag Count, SYN Flag Count, Average Packet Size, Total Len Fwd Pck, Active Mean and Min, ACK Flag Count, and Init_Win_bytes_fwd are observed to be the discriminating features embedded in CICIDS2017.

(c) Efficiency of attack remediation: The multi-layer monitoring of network events helps to inform the controller and the switches connected to re-discover new routes, dynamically re-deploy network function service-chain and thus mitigating distributed botnet attacks upstream closer to the origin. The application layer and slow-rate attacks require co-operation between the application servers attached to the downstream network and the SDN controllers and the dataplane switches upstream in the host networks.

- The detection times for each class of attacks and the depth in the VARMAN detection pipeline at which they are detected and mitigated/remediated are measured with micro-benchmarks. Based on the layer (dataplane/controller), level (1–4), and classification overhead (Anomaly detection/ML pipeline), the attack remediation speed varies. The application attack shows the highest latency to detect and mitigate or remediate, as it incurs overhead across all levels of the analytics pipeline and it takes more time to run the application, protocol correlation/analytics. (Section 4.3.3)
- We evaluate the multi-controller remediation performance when the network is under DDoS flooding attack from multiple hosts targeting single downstream application server. In VARMAN, due to the presence of attack detection and remediation scheme based on a heuristic algorithm, the malign flows are dropped at the upstream switches and the workload on the attacked controllers and also to other benign controllers are unaffected as well. (Section 4.3.5)

5.2. Comparison with selected works

In our research, we have addressed the cyber-attack holistically on all dimensions, more detailed analytics and classification in multiple planes, to propose an advanced Intrusion Detection system. Here we give an overview of the improvements in our solution, in comparison to selected Extended-SDN and ML-based SDN solutions.

5.2.1. Towards extended SDN architecture

- **Entropy vs. Statistical**: Some of the entropy or statistical-based attack detection methods, utilized only one feature/attribute of the flow i.e. Source IP/ Destination IP to calculate entropy or determine as an anomaly. They have not fully harnessed this feature extraction technique for historical analysis or classification. The accuracy of the entropy-based approaches [29,43] decreases drastically at lower attack rates because the entropy difference in low volume attacks is almost non-detectable, which is common in IoT based port scanning and DDoS attacks. But in VARMAN, suspicious flows are not marked based on threshold alone, as we do the Statistical/multi-feature anomaly detection in conjunction with Anti-Spoofing/Turing test (Challenge–Response) with the incoming new-flows. (Section 4.3.1)
- **Feature selection**: When compared to other IDS schemes [44,45], which selected about 4 to 6 features for classification, but we have utilized about 10 to 16 features and create multi-variate classification and perform anomaly detection based on traffic flow, SDN protocol behaviors and so on. Although this increased the complexity, our experiments proved that the features that are monitored to determine anomalies and detect attacks are the right ones and effective in terms of accuracy and predicting the attacks at the dataplane based on coarse-grained security mechanism. On the contrary, other proposals [46] classified with limited feature-set which slipped certain attacks through the IDS and made the downstream services unresponsive to normal users as well. (Section 4.3.2)

- **Coverage of attacks**: Most of the proposals in the NIDS addressed just the DDoS flooding attacks [47,48] and limited only to traffic anomaly detection methods [49]. But our attack detection mechanisms both in dataplane and control plane classification, we designed scheme to detect transport layer and application level attacks. Also, both in the traffic simulation and datasets we used in VARMAN, covered diverse and deep protocol attack patterns. Section 4.3.3 proves this claim.
- **Datapath optimization**: Ref [36] Overwatch, their defense actuators are deployed in the data plane, our forwarding latency is much reduced compared to their datapath, due to optimization with DPDK and kernel memory mapped Netfilter model in VARMAN SDN stack. Their primary attack classifier algorithm runs on the control plane only and hence prone to control plane saturation and overload. Our approach is optimized since we have extended the core switching layer in the data plane, to execute first stage attack detection and DDoS prediction functions, stateful functions (cached flow-rules/action set) instructed by the controller, on the switches. This has enabled improvements in our system by further reducing the control channel traffic and CPU processing overhead at the controller. This scheme also contributes to the speed of detection in the dataplane as it is critical not to create a larger bump in the wire-speed.
- **Control plane saturation avoidance**: In SD-Anti-DDoS [50], The majority workload of their defense system is taken by the control plane as they used the classical SDN stack and their DDoS detection mechanism is designed around the Packet-In trigger and flow analysis. Due to our stateful/security-aware SDN dataplane, light-weight computational functions are offloaded to the switches for in-line processing in the switches. Since VARMAN data plane is designed for high capacity core switches or IoT Gateways at the Edge network, it can monitor and secure multiple network segments. We can tackle the challenges and intricacies for extensive scale-up in very large-scale networks. (Section 4.3.4)
- **Light-weight detection**: In [38], the authors proved that the extended SDN Architecture is a feasible approach and extended the OpenFlow specification for supporting NFV modules in SDN. As the Application payload analysis is too expensive to be performed on the switch, it is deployed as a virtualized network function in a software module on another machine connected to the dataplane, so this redirection of flows causes additional round-trip delays even though this latency is incurred within the same dataplane. Compared to this design, in our SDN architecture, the anomaly and coarse-grained attack detection, first level mitigation NFs are integrated within the switch fabric and hence the performance hit is very minimal for extending the dataplane in VARMAN. Moreover, in their proposal, the classification functions involve matching the TCP/IP and application headers with patterns specified by the controller, which needs high-speed storage, optimizing search mechanism and additional CPU cycles. They have not considered other light-weight statistical switch counter-based traffic classification methods in their work. In VARMAN, we do not have such complex pattern-matching logic and used the statistical features-based detection effectively.

5.2.2. Efficiency of attack detection/classification

- **Prediction efficiency — Training time and window size:** Compared to other prediction algorithms [46], we have achieved higher accuracy with lesser training time and smaller sampling window interval. The prediction algorithm achieved a consistent detection rate ranging between 96% and 98%, with under 900s training time and 30s sampling window as opposed to other comparable schemes that needed more than 3000s training/ 60s window to achieve similar prediction accuracies. This improvement is due to hybrid behavioral and statistical anomaly detection in the dataplane.
- **False Positives (FP) and False Negatives (FN)**: With the combination of dataplane algorithms, VARMAN generates FPs only in slow-rate attacks with the greatest percentage being 1.45%. The ForChaos [40] reported FP rates about 5.34% for slow-rate attacks and other studies test their algorithm against Flooding types of DDoS only, in which our algorithm does not generate any FPs. Our algorithm generates a FN rate 2.23% while other studies [40] report FN results ranging from 5.6% up to 12.5%. The choice of acceptable rates and metric depends on the business objective. For example: for SPAM filter use-case (positive class is "spam"), false negatives (spam goes to the inbox) are more acceptable than false positives (non-spam is caught by the spam filter) and for IDS/IPS user-case (positive class is "attack") false positives (normal packets that are flagged as possible attack) are more acceptable than false negatives (attacks that are not detected). From the observations of experiments in Section 4.3.1, it is reasonable to infer that VARMAN showed consistency and acceptable rate (96.8% to 99.1% for slow-rate and 100% for flooding) at all traffic intensity level (flooding and slow-rate attacks) compared to other works, in the context of Intrusion Detection solution for network security.
- **Principal Component Analysis (PCA) vs. Stacked Auto Encoder (SAE)** : Ref [12], They combined all CICIDS2017's files together and fed them through the AE and PCA units for a compressed and lower dimensional representation of all the fused data. The execution time (time to build the model) was 74.39 seconds. This is while the execution time for our proposed system using Random Forest is 21.52 seconds with a comparable computing configuration in our testbed. Furthermore, our proposed intrusion detection system targets a combined detection process of all the attack families.
- **SAE and Softmax vs. SAE and Random Forest**: ML Intrusion detection systems have generally used one class of algorithm either shallow or deep learning algorithms for training the ML system to detect and classify attacks. But in recent times, using two or more different techniques to form a hybrid has improved the overall performance. The stacked *Autoencoder* in conjunction with output *softmax* classifier is employed in prior works such as OverWatch [36] and SD-anti-ddos [50], which are less efficient than Random Forest classification. Niyaz et al. [51] presented a comparative study of SDN-based IDS with different combinations, one with "Stacked Auto-Encoder (SAE)" for feature extractor/softmax classifier (binary/multi-class) that outperformed the ANN classifiers.
- **Symmetric SAE vs. Asymmetric SAE**: After systematic analysis, we selected this NDAE with a non-symmetric stacked pipeline. So, there is a fundamental change from the conventional pipeline that is symmetrically stacked and improved by asymmetric stacking of two Autoencoder layers in conjunction with Random Forest (RF) as the output classifier. This hybrid scheme (deep/shallow learning layers) in our design, has contributed to arrive at a resource optimal and fairly accurate classification system. (Section 4.3.6)
- **Deep Belief Network (DBN) vs. Hybrid NDAE**: In [28], the authors proposed a mainstream DBN model-based NIDS. Our experiments show that our VARMAN ML-model under similar SAE architecture, trained with the same dataset size, achieves better accuracy even with reduced labeled sample size. (with only 25% labeled samples). Our evaluations show higher accuracy compared to their work. The non-symmetric stacked neural network model utilized in the VARMAN system achieves a considerable saving in training time of 94.96%. The heuristic we chose for feature selection in the hybrid-ML training network, has optimized the memory usage for data passing through the VARMAN ML workflow, with an average saving of 90.8% when compared with DBN. (Section 4.3.7)

- **Old vs. Recent Datasets**: NSL-KDD, KDD-99 are some of the old datasets still utilized by researchers, but in our analysis these datasets have been outdated and do not simulate the traffic condition/security incidents that are happening in today's networks. We are one of the few researchers to utilize multiple datasets — popular NSL-KDD [11], recent benchmark CICIDS2017 [12] and HogZilla [13] dataset. We refined these datasets further, to extract a normalized dataset (i.e., containing a comparable number of normal traffic and malware samples) and thus organically generate realistic datasets for training the intrusion detection systems. We compared the results of these schemes [51,52] in terms of accuracy and recall — and our system shows much better results in the evaluation under similar datasets and simulations. (Section 4.3.6)
- **CICIDS2017 Related proposals**: Ref. [53], Each classifier was trained to detect a certain attack category using selected features and only a small portion of the CICIDS2017 dataset instances were used to evaluate their system. Conversely, in this paper, we use all the instances of the CICIDS2017 dataset. This work SU-IDS [54] did a comparison between NSL-KDD/CICIDS2017 dataset and produced good classification results. With similar comparison experiment with our VARMAN scheme, due to efficient feature selection and NDAE/RF hybrid classifier, the results show the superiority of our solution in terms of PR (Precision/Recall) and ROC curves, for much larger attack vector space.

5.3. Validity of our solution

Although this study has successfully demonstrated the significance of the feature dimensionality reduction techniques which led to better results in terms of several performance metrics as well classification speeds for an IDS, it has certain limitations and challenges which are summarized as follows.

Control channel failure: In an SDN model, the controller(s) on the control plane is connected to the dataplane switches through the secure control channel called *OpenFlow* channel. This channel is the usual point-of-target for the attackers and the entire operation of IoT-Fog Infrastructure/applications can be impacted if this channel is compromised. The NFV adoption in virtualized data-centers especially gained traction in telecoms, however, there are problems related to micro-services chaining and the isolation of virtualized functions in hypervisor-based infrastructure.

Fault tolerance: The key aspect of fault tolerance in our system is the ability of the multi-level and cross-plane scheme, to detect a large set of well-known attacks. Our models have been trained to detect the 14 up-to-date and well-known type of attacks. Moreover, the deployment of distributed intrusion detection systems in the network can enable fault tolerance.

Datasets and fine tuning the deep learning system: The preparation of the datasets with latest attacks, fine tuning the density of neurons in each layer and the count of layers for optimal training (runtime memory and CPU cycles) have resulted into an improved classification system. The ML/DL scheme used in our NIDS is scalable to work with larger and complex datasets. As the primary goal of our ML-based NIDS is to address the demands of fine-grained traffic pattern detection, the experiments under most recent datasets (CICIDS2017, Hogzilla2018) have shown great potential to face large scale modern network
attacks.

Adaption to non-stationary traffic/non-linear models: The AE has the ability to represent models that are linear and nonlinear. Moreover, once the model is trained, it can be used for non-stationary traffic. We intend to further extend our work in the future with an online NIDS and observatory.

Model resilience: Section 4.3.6, the achieved FP rate is negligible (about 0.001), which may reflect the built-in attack resiliency in the fine-grained ML Hybrid classifier. Moreover, our models were trained (Section 4.3.7) in an offline manner, not offering any chance for dataset corruption/manipulation and with organically generated datasets. This ensures that an adversary cannot inject misclassified instances during the training phase. On the contrary, such a case could occur with online-trained models. Therefore, it is essential for the machine learning system employed in intrusion detection to be resilient to adversarial attacks.

Hardening the ML-based solution for SDN security: It is a common trend that cybersecurity and malware analytics system employ ML/DL based AI algorithms to analyze correlations, patterns in the traffic data and detect/classify the attacks. Because the datasets are developed by researchers and made them publicly available, ML-based systems trained with these datasets can be "out-maneuvered by adversaries" who have access to study the same datasets. As these methods are exploited more widely, the sophisticated cyber-criminals devise adversarial ML attacks to breach these NIDS. Hence the new strategies have to be determined with organic datasets, proper heuristics, the trade-off between classification accuracy/hardening of the ML scheme.

5.4. Future work

Though the computational power and memory speeds are increasing, still it is a challenge to build and maintain the complex software-based implementations from getting overwhelmed in-line or opening up new malware/bugs/vulnerabilities. It is important to exploit advanced technologies such as co-processors/GPGPUs/software-acceleration in the kernel and thereby build a customizable data plane that handles packet/flows at wire speed. To this end, we already advanced our research with the stateful processing in dataplane switches, application offloading and location awareness into the dataplane layers and more powerful control layer functions into our SDN based solution. Additional features along with the existing features given to the training phase can also improve the detection accuracies and this will also be considered in the future. Most of the contemporary anti-DDoS solutions detect the attacks with the data collected in one phase of the attack life-cycle or kill chain. But we hypothesize that correlating traffic data from multiple phases/stages of the botnet cycle will offer more insight and help design more efficient attack prediction or forecasting systems. Given that SDN data centers, are prone to DDoS attacks due to flow-based network traffic control, the solutions require flexible switching fabric to deploy stateful and light-weight logic, that can be programmed at runtime into new generation whitebox switches. Formal methods, for validating the security parameters, correctness, scope, and coverage as per the use-case specifications in the SDN environment, have to be developed.

6. Conclusions

In this article, we presented a highly sophisticated intrusion detection and mitigation framework for the security of emerging softwarized networks and SDNFV enabled data centers, by exploiting the most advanced deep-learning and efficient classification algorithms. We examined the emerging field of SDN, outlined various intrusion detections mechanisms using ML/DL approaches and emphasized SDN technology as a platform using ML/DL approaches to detect vulnerabilities and monitor networks. We proposed a novel hybrid machine learning based classification model constructed from both shallow and deep learning algorithms. The AI-based multi-plane intrusion detection is implemented leveraging SDN's status monitoring as well as traffic capturing under a global view. It integrates and coordinates multiple stages of attack detection including feature selection and flow classification, anomaly detection to detect novel intrusions with a self-learning ability. Most notably, unlike most previous works, we have evaluated the capabilities of our model based on the most recent CICIDS2017 and Hogzilla

benchmarks which have datasets resembling the real-world attack profile, revealing a consistent level of classification accuracy. We have also compared our hybrid detection model against the main stream techniques. These comparisons have demonstrated that our model offers up to a 5% improvement in accuracy and training time reduction of up to 98.81%. Our framework is extensible and generic, hence can be incorporated with many advanced classification algorithms.

Compared with standard OpenFlow-based SDN, the extended SDN design in VARMAN causes little controller overhead for providing NFV modules. The evaluation demonstrates that the redesigned SDN is efficient and the additional processing in the data plane adds minimal overhead to the standard OpenFlow pipeline on the switches. We also studied the controller overhead (amount of traffic from the data plane to the control plane) and the redirection ratio (amount of traffic going to DPI and the other NFV modules) through a series of microbenchmarks and drew valuable insight to fine tune the architecture. In the standard SDN architecture, since the data plane cannot extract the events, all the TCP packets, which account for 77.23% of the input traffic, will be redirected to the controller for event extraction. However, in our VARMAN SDN stack, due to added intelligence and DPI capabilities the dataplane switches can extract the events, and only about 1.2% of the input traffic are forwarded and processed for the security policy-miss or suspicious cases. During attack conditions, the overall throughput for the end-hosts is improved from a few Mb/s to more than 100 Mb/s and new connection latencies are reduced because of the short transmission path. Moreover, only the necessary traffic will be redirected to the NFV modules and the redirection ratios vary with the composition of traffic types and the required NFV modules.

As defense functionalities are spread across multiple layers, this enables security against DDoS attacks on different levels. Based on the movement of threats and malware trajectories, the defense NFs may be dynamically pushed closest to the origin of attack to realize a moving-target-defense system. In the future, the proposed framework can be a reference design for different application domains namely 5G, Smart Grids and Industry 4.0 networks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the office of Dean-Research at Amrita Vishwa Vidyapeetham, Amritapuri, India and the Visveswaraya Ph.D. fellowship from the Government of India.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: Enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. (2008).

[2] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. McK-eown, S. Shenker, Ethane: Taking control of the enter- prise, ACM SIGCOMM Comput. Commun. Rev. 37 (4) (2007) 112, http://dx.doi.org/10.1145/1282427.1282382.

[3] A.L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Commun. Surv. Tutor. 18 (2) (2017) 1153–1176.

[4] M. Wang, Y. Cui, X. Wang, S. Xiao, J. Jiang, Machine learning for networking: Workflow, advances and opportunities, IEEE Netw. 32 (2) (2018) 92–99.

[5] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L.A. Yau, Y. Elkhatib, A. Hussain, A. Al-Fuqaha, Unsupervised machine learning for networking: Techniques, applications and research challenges, arXiv preprint arXiv:1709.06599, 2017.

[6] G. Xu, Y. Mu, J. Liu, Inclusion of artificial intelligence in communication networks and services, ITU J. (1) (2017) 1–6.

[7] P.V. Klaine, M.A. Imran, O. Onireti, R.D. Souza, A survey of machine learning techniques applied to self organizing cellular networks, IEEE Commun. Surv. Tutor. PP (99) (2017) 1.

[8] Z.M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, K. Mizutani, State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2432–2455.

[9] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, R. Atkinson, Shallow and deep networks intrusion detection system: A taxonomy and survey, arXiv preprint arXiv:1701.02145, 2017.

[10] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, Andreas Hotho, A Survey of Network-based Intrusion Detection Data Sets arXiv:1903.02460v1 [cs.CR], 2019.

[11] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, Nsl-Kdd Dataset. http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html, 2012.

[12] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, CICIDS2017: Toward Generating a new intrusion detection dataset and intrusion traffic characterization, in: International Conference on Information Systems Security and Privacy (ICISSP), 2018, pp. 108–116.

[13] P.A.A. Resende, A.C. Drummond, The hogzilla dataset. http://ids-hogzilla.org/dataset, 2018.

[14] P. Krishnan, J.S. Najeem, K Achuthan, SDN framework for securing iot networks, in: N. Kumar, A Thakre (Eds.), International Conference on Ubiquitous Communications and Network Computing UBICNET 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer, Cham, 2018, pp. 116–129.

[15] Krishnan Prabhakar, Achuthan Krishnashree, Managing network functions in stateful application aware SDN, in: 6th International Symposium on Security in Computing and Communications (2018), Springer Communications in Computer and Information Science Series(CCIS), ISSN: 1865:0929.

[16] P. Krishnan, K. Achuthan, CloudSDN: Enabling SDN framework for security and threat analytics in cloud networks, UBICNET 2019, LNICST 276 (2019) 151–172, http://dx.doi.org/10.1007/978-3-030-20615-4_12.

[17] A.R. Curtis, et al., DevoFlow: Scaling flow management for high-perfor- mance networks, in: ACM SIGCOMM, 2011.

[18] D. Hu, P. Hong, Y. Chen, FADM: Ddos flooding attack detection and mitigation system in software-defined networking, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.

[19] S.S. Bhunia, M. Gurusamy, Dynamic attack detection and mitigation in iot using SDN, in: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), IEEE, 2017, pp. 1–6.

[20] T. Tang, S.A.R. Zaidi, D. McLernon, L. Mhamdi, M. Ghogho, Deep recurrent neural network for intrusion detection in SDN-based networks, in: Proc. IEEE NetSoft'18, Montreal, Canada, 2018.

[21] T.A. Tuan, L. Mhamdi, D. Mclernon, S.A.R. Zaidi, M. Ghogho, Deep learning approach for network intrusion detection in software defined networking, Int Conf Wirel Netw Mob Commun, http://dx.doi.org/10.1109/WINCOM.2016.7777224, 2016.

[22] S. Choudhury, A. Bhowal, Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection, in: 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM).

[23] M. Anbar, et al., Comparative performance analysis of classification algorithms for intrusion detection system, 14th Annual Conference on Privacy, Security and Trust (PST), 2016.

[24] M. Latah, L. Toker, Towards an efficient anomaly-based intrusion detection for software-defined networks, IET Netw. (2018).

[25] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.R. Sadeghi, S. Tarkoma, 'IoT Sentinel: Automated device-type identification for security enforcement in IoT'. in: Proc. of IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, USA 2017, pp. 2177-2184.

[26] G.A. Ajaeiya, N. Adalian, I.H. Elhajj, A. Kayssi, A. Chehab, Flow-based intrusion detection system for SDN, in: Proc. of 2017 IEEE Symposium on Computers and Communications (ISCC); 3-6 2017; Heraklion, IEEE, New York, 2017, pp. 787–793.

[27] C.H. Huang, T.H. Lee, L. Chang, J.R. Lin, G. Horng, 'Adversarial attacks on SDN-based deep learning IDS system'. in: Proc. of International Conference on Mobile and Wireless Technology (ICMWT 2018), Hong Kong, China, 2018, pp. 181-191.

[28] K. Alrawashdeh, C. Purdy, Toward an online anomaly intrusion detection system based on deep learning, in: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, Anaheim, California, USA, 2016, pp. 195–200.

[29] Ihsan H. Abdulqadder, et al., Deployment of robust security scheme in SDN based 5G network over NFV enabled cloud environment, 2018 IEEE Trans. Emerg. Top. Comput. http://dx.doi.org/10.1109/TETC.2018.2879714.

[30] F. Junfeng Xie, Richard Yu, Tao Huan, Renchao Xie, Jiang Liu, Chenmeng Wang, Yunjie Liu, A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges, IEEE Commun. Surv. Tutor. http://dx.doi.org/10.1109/COMST.2018.2866942.

[31] Nasrin Sultana, Naveen Chilamkurti, Wei Peng, Rabei Alhadad, Survey on SDN based network intrusion detection system using machine learning approaches, Peer-to-Peer Netw. Appl. (2018) http://dx.doi.org/10.1007/s12083-017-0630-0.

[32] Tam n. Nguyen, The Challenges in ML-based Security for SDN, in: 2018 2nd Cyber Security in Networking Conference (CSNet).
[33] Yan Qiao, et al., Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges, IEEE Commun. Surv. Tutor. 18 (1) (2016) 602–622.
[34] Gkountis Christos, et al., Lightweight algorithm for protecting SDN controller against ddos attacks, in: Ifip Wireless and Mobile NETWORKING Conference, IEEE, 2018, pp. 1–6.
[35] Wang, et al., Sguard:a lightweight SDN safe-guard architecture for DoS attacks, China Commun. 14 (6) (2017) 113–125.
[36] BiaoHan, et al., OverWatch: A cross-plane DDoS attack defense framework with collaborative intelligence in SDN, Hindawi Secur. Commun. Netw. Volume 2018, https://doi.org/10.1155/2018/9649643.
[37] X. Yang, et al., SDN-based DDoS Attack Detection with Cross-Plane Collaboration and Lightweight Flow Monitoring, GLOBECOM, 2017.
[38] Ying-Dar Lin, Po-Ching Lin, Chih-Hung Yeh, Yao-Chun Wang, An extended SDN architecture for network function virtualization with a case study on intrusion prevention, IEEE Netw. (2015).
[39] ONF: https://www.opennetworking.org/.
[40] Andria Procopiou, Nikos Komninos, Christos Douligeris, ForChaos: Real time application DDoS detection using forecasting and chaos theory in smart home iot network, Wirel. Commun. Mob. Comput. Volume 2019, http://dx.doi.org/10.1155/2019/8469410.
[41] Razan Abdulhammed, Hassan Musafer, Ali Alessa, Miad Faezipour, Abdelshakour Abuzneid, Features dimensionality reduction approaches for machine learning based network intrusion detection, Electronics 8 (2019) 322, http://dx.doi.org/10.3390/electronics8030322.
[42] N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Trans. Emerg. Top. Comput. Intell. 2 (1) (2018) 41–50.
[43] Ku'bra Kalkan, JESS: Joint entropy based DDoS defense scheme in SDN, IEEE J. Sel. Areas Commun. http://dx.doi.org/10.1109/JSAC.2018.2869997.
[44] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, K. Goswami, Machine-learning based threat-aware system in software defined net- works, in: Proc. IEEE ICCCN'17, Vancouver, BC, Canada, 2017, pp. 1–9.
[45] Hurley, J. E. Perdomo, A. Perez-Pons, HMM-based intrusion detection system for software defined networking, in: Proc. IEEE ICMLA'16, Anaheim, CA, USA, 2016, pp. 617–621.
[46] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, B. Yang, Predicting network attack patterns in SDN using machine learning approach, in: Proc. IEEE NFV-SDN'16, Palo Alto, CA, USA, 2016, pp. 167–172.
[47] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, L. Gong, Detection and defense of ddos attack-based on deep learning in openflow- based SDN, Int. J. Commun. Syst. (2018).
[48] L. Barki, A. Shidling, N. Meti, D.G. Narayan, M.M. Mulla, Detection of distributed denial of service attacks in software defined networks, in: Proc. IEEE ICACCI'16, Jaipur, India, 2016, pp. 2576–2581.
[49] A.S. da Silva, J.A. Wickboldt, L.Z. Granville, A. Schaeffer-Filho, ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN, in: Proc. IEEE NOMS'16, Istanbul, Turkey, 2016, pp. 27–35.
[50] Yunhe, et al., SD-anti-DDoS: Fast and efficient DDoS defense in software-defined networks, J. Netw. Comput. Appl. 68 (2016) p65–79.
[51] Q. Niyaz, W. Sun, A.Y. Javaid, 'A deep learning based DDoS detection system in software-defined networking (SDN)', EAI Endorsed Trans. Secur. Safety 4 (12) (2017) 1–12.
[52] T.A. Tang, L. Mhamdi, D. McLernon, S.A.R. Zaidi, M. Ghogho, Deep learning approach for network intrusion detection in software defined networking, in: Proc. IEEE WINCOM'16, Fez, Morocco, 2016, pp. 258–263.
[53] R. Vijayan and, D. Devaraj, B. Kannapiran, Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection, Comput. Secur. 77 (2018) 304–314.
[54] E. Min, J. Long, Q. Liu, J. Cui, Z. Cai, J. Ma, SU-IDS: A semi-supervised and unsupervised framework for network intrusion detection, in: International Conference on Cloud Computing and Security, Springer, Cham, Switzerland, 2018, pp. 322–334.