Detecting Stealthy Scans in SDN using a Hybrid Intrusion Detection System

Abdullah H Alqahtani*
ahalqahtani1@sheffield.ac.uk
The University of Sheffield
Sheffield , United Kingdom

John A. Clark*
john.clark@sheffield.ac.uk
The University of Sheffield
Sheffield , United Kingdom

ABSTRACT

The Software-Defined Network (SDN) paradigm separates the control layer from the infrastructure layer. SDN switches maintain so called 'flow rules' which define how they handle incoming communications, e.g. indicating how specific packets may be forwarded. Reconstructing such flow rules of a network can facilitate attacks on the victim network, i.e. where it is a preliminary to further attacks. Its prevent or detection is a significant challenge. An adversary may adopt a stealth scan to gain this information (seeking to avoid detection). Advanced Persistent Threats (APTs) are sophisticated attacks that implement stealth scans in some of its attack phases. Insiders play essential roles for APTs campaign such as delivering malicious software. In this paper we build a hybrid Network Intrusion Detection System (NIDS) to detect such forms of stealth scan targeting SDN networks to reconstruct flow rules. The proposed model takes advantage of the two main detection techniques in Machine Learning (ML) (signature-based and anomaly-based detection) to reduce the error rates. XGBoost (for supervised-based detection) and One-Class Support Vector Machine (one-class SVM) (for anomaly-based detection), form the basis of our proposed system. Promising results are shown scoring 0.98, 0.98, 0.90 and 0.94 in Accuracy, Recall, Precision and F1-score respectively.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection and malware mitigation**.

KEYWORDS

Software-Defined Network, SDN, stealth scan, APT, IDS, Machine Learning

1 INTRODUCTION

1.1 Software Defined Networks

Software-defined networks (SDNs) separate the control plane, where the routing decisions are made, from the data plane, which deals with the packet forwarding process. Every network switch has one or more data tables containing flow rules. These flow rules are a set of instructions that indicate how to handle arriving packets, e.g. forward the packet to a port, drop it, or send it to the controller [28]. However, an adversary can send probing packets to switches and gather information to reconstruct the flow rules [9, 36, 37]. This information can then be used to facilitate further attacks.

1.2 Advanced Persistent Threats

Advanced Persistent Threats (APTs) are cyber-attack campaigns that are complex and sophisticated. They are considered as one of the major cyber threats for large enterprises. APT creators are usually highly skilled and well-funded. They employ advanced techniques in their attacks including the exploitation of zero-day vulnerabilities. Unlike traditional attacks, people behind APTs act stealthily. To remain undetected they move slowly and with low volume in their communications and activities (often referred to as a "low and slow" strategy) [31].

1.3 The Insider Perspective

Insiders are employees or third parties in an organisation whose privileges are exploited to launch or aid an attack. They may unintentionally contribute to an attack, e.g. by installing malicious software without knowing its harmfulness, or else do so intentionally [17, 21]. Some APT attacks require the co-operation of an insider. For example, analysis of Stuxnet reveals that it is likely that an insider was involved in delivering the malicious malware via a compromised USB stick [13, 15]. Actions taken by insiders may be authorised but malicious. The malice lies in the *purpose* for which privileges are being used. It is the abuse of privilege that matters, not simply the use. Of course, such distinctions may be hard to reliably detect. Furthermore, insiders' presence on the network is authorised and does not in and of itself arouse suspicion. An insider is therefore in a very powerful position. The insider can gather sensitive information or install required malicious software in the targeted network [19]. For an isolated or well protected network, insider involvement may be essential for a successful attack.

Detection of insiders in networks is hard. An insider adversary can also adopt stealth strategies to make detection even harder. For example, as part of reconnaissance activities they may carry our scanning over an extended period of time, with probes issued at a very low rate. This also presents an opportunity for defence. Detection of insiders can be regarded as the 'stretch goal' for intrusion detection in modern systems, such as those employing SDN architectures. If we can detect insider attacks, external attacks should not present much of a problem. Thus, we believe it is prudent to target insider attacks as a priority. Finally, an *insider perspective* for intrusion detection can provide defence-in-depth. Ultimately, a system's defences may fail to prevent compromise of a node by an external agent and that comprised node is now an insider!

SDN provides an extensive network management API that allows timely network information gathering and configuration. However, this API can be easily abused by an attacker. An insider can carry out a reconnaissance mission and escape detection more easily than an outsider. In this paper, we develop ML-based techniques to detect malicious scanning activity that is targeted towards the inferring network device "flow rules".

1.4 Intrusion Detection Systems and Machine Learning

An Intrusion Detection System (IDS) is a system that monitors a network or a host to detect suspicious activities. A Network Intrusion Detection System (NIDS) is a type of IDS that monitors network traffic to detect suspicious attacks. Signature-based and anomaly-based detection are the two main detection techniques adopted by IDSs. Signature-based approaches, also called *misuse-based*, detect abnormalities based on predefined patterns or *signatures* of attacks. Signature-based approaches can be used with considerable confidence to detect known attacks and are often exhibit low false alarm rates, i.e., they are not prone to classifying benign traffic or activities as malicious. However, such approaches may often miss unforeseen malicious activities that have different properties to known malware.

In anomaly-based detection the system is trained over normal traffic and builds a profile of it. When the behaviour of a user differs significantly from its normal profile a potential attack is suspected. Therefore, when adversaries create new attacks and adapt their behaviour (as APTs do) employing anomaly-based IDS has better prospects for detecting such previously unseen malicious activity. However, this comes at a price. Anomaly-based approaches raise more false alarms, often referred to as False Positives (FPs). A further problem is that some malicious activity, even known malicious activity, may 'look like' normal activity, effectively exhibiting sets of measured properties consistent with experienced benign activity and so not raise an alarm. This gives rise to so called False Negatives (FNs).

Interest has grown in the use of Machine Learning (ML) techniques to provide the basis for IDSs. ML classifiers may be trained using supervised-learning (where the classification model is trained over labelled data (for example malicious or normal), and unsupervised-learning, where the model is trained over normal traffic and attacks are detected as deviations from the learned pattern of normal behaviour.

1.5 Motivations and Contributions

Stealth scans can be used to gather information that underpins further attacks, such as discovering the running applications in the victim network (e.g. defence tools or network balancing) and DDoS attacks [2]. Their adoption of some APT characteristics makes stealth scans hard to detect. With the help of an insider, the attackers can bypass the front-line defences and access the targeted network.

Implementing network IDS (NIDS) in SDNs is easier and more flexible than in traditional networks. In a traditional network, intrusion detection requires the provision of extra tools or methods to collect information from network devices. In an SDN retrieving such information is easier: the SDN controller has a global view of the whole network and its devices and can retrieve network status and traffic flow information [35]. Network information gathering primitives are provided as part of general SDN operation.

We propose a hybrid Machine Learning based NIDS to detect stealth attacks, such as APTs, in an SDN. The incorporation of the two detection techniques, signature-based and anomaly-based, can address the weaknesses of the individual approaches. An anomaly detection approach can enhance the detection of unseen attacks (a weakness of the signature-based approach) while the false positives (false alarms) of anomaly detection will be reduced by use of a signature-based technique. Two significant Machine Learning (ML) approaches are adopted to implement these approaches: XGBoost [8] for signature-based detection and a One-class Support Vector Machine (one-class SVM or OC-SVM for short) [25] for anomaly-based detection. We evaluate the proposed model over an SDN-based dataset that includes stealth scans targeting the reconstruction of flow rules. The dataset (called APT-SDN dataset) has been made available [3]. It contains network data from three different sized networks. We carry out dimensionality reduction on the dataset and hyper-parameter tuning on both ML techniques to improve performance. All scanning takes place from within the relevant network, attacks are carried out by insider.

The contributions of this paper are:

- Identification and use of the insider perspective as a conservative and challenging default threat model for SDN intrusion detection. The insider perspective considers the attacker at their most advantageous position. As observed above, this is the de facto position that would arise from network compromise.
- Demonstration of a hybrid IDS that integrates two detection techniques to reduce the incorrect decisions (FPs and FNs) is novel for this domain.To the best of our knowledge, it is the first IDS system using anomaly-based techniques to detect APTs in SDN networks.
- The evaluation of the proposed model over datasets generated by SDN-based networks of various sizes.
- A comparison between the proposed system and the other standard ML detection techniques and most related works is given.

1.6 Paper Organisation

The remainder of this paper is organised as follows. Section 2 reviews the most relevant proposed IDS systems in the literature. Section 3 explains what we mean by stealth flow rules reconstruction in SDN. Section 4 describes the proposed approach. Section 5 presents the implementation and evaluation of the proposed model. Section 6 presents our conclusions and future work.

2 RELATED WORKS

Network Flow Guard [10] is a modular application proposed to leverage SDN controller to detect ARP spoofing generated by insider.Some research proposals have sought to detect unknown attacks in Software-Defined Networks using anomaly-based detection systems [11, 32, 33]. They have employed Deep Learning techniques and have evaluated their approaches over NSL-KDD[20] which does not include SDN traffic. However, the developed detection systems have exhibited high error rates. Integration of a One-class SVM and a long short-term memory (LSTM)-Autoencoder was proposed and tested over an SDN based dataset (InSDN [12]) [24], however, the attacks form more than 80% of total traffic in the dataset, which makes it far from representative of a stealth attack approach. In a similar way, a hybrid system [18] using an LSTM together with a Conventional Neural Network (CNN) are proposed for anomaly-detection, achieving a high detection rate. However, it is evaluated over the CICIDS 2017 dataset [27] which contains only traditional attacks. Nevertheless, none of aforementioned works consider stealth attacks such as APTs or have been evaluated over datasets that include APTs. Table 1 presents a comparison between our work and the most relevant works.

3 STEALTH FLOW RULES RECONSTRUCTION IN SDN

Stealthy attacks, such as APTs, exhibit low rates of traffic [23] as they carry out their malicious activities, either when moving around the network or when communicating. They operate under the detection thresholds of most defence tools. The most popular APTs such as Duqu [5], Kelihos [29], Flame [34], Shamoon [22], Hydraq [14] and Pegasus [1] implement waiting times during their attacks to avoid a high volume of communications within a certain time, making it stealthier. Attackers can send probing packets to infer the required information such as source IP, destination IP, MAC address, port numbers and used protocols to reconstruct flow rules in SDN switches [2]. SDNMap [26] is a network scanner used to reconstruct flow rules in SDN. The open-source APT-SDNmap [4] is an extension to SDNMap employing different waiting times during their scans. For example, they implement two types of sleep functions between their scans. Waiting for a random time between 120-150 seconds while the scan is going from one category to another (for example, scanning all open ports over one protocol; when going to scan again over another protocol it will wait around 120-150 seconds). The other waiting function, implements shorter waiting times, e.g. 10-15 seconds, between scans inside one type of scan (for example, while doing port scan over one protocol, it will wait 10-15 seconds from scanning one port to another). These waiting time strategies are used to implement various degrees of stealth. In this paper, we assume our adversary is an insider and can install a scanner inside a victim's network. We assume that the insider has installed APT-SDNmap in at least one host and launches flow rule reconstruction attacks using it.

4 PROPOSED SCHEME

The proposed hybrid system is composed of two main components. The signature-based detection module uses the XGBoost classifier and the anomaly-based detection module uses a One-class SVM. Streaming data is mirrored to the NIDS for investigation. When a new packet arrives at the IDS, the XGBoost classifier determines whether the pattern of the traffic is malicious or normal. If the pattern of the attack is already defined, then the signature-based detection is going to detect it and for that reason we make XGBoost carry out the first check. If it is classified as malicious, then this transaction is deemed to be an attack and the network administrator will be notified for further action. Otherwise, the transaction is inspected by the anomaly-based component (using a One-class SVM), as a further check for unknown attacks. If it is deemed malicious, then the administrator would again be notified to take action, such as updating a flow rule table. If it is normal in both cases, then no further action is taken.

The proposed system architecture is illustrated in Fig 2. The NIDS has the advantage of the global view of the SDN controller to monitor all network devices. The controller can request the network

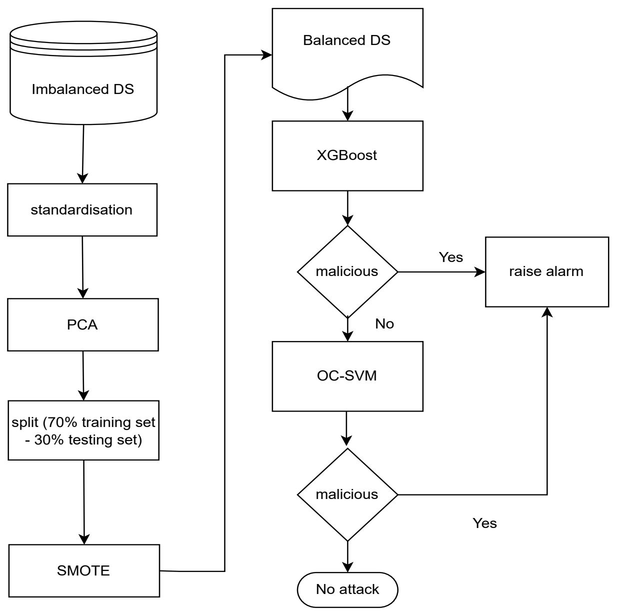

Figure 1: Hybrid NIDS

statistics from OpenFlow switches using $ofp\_flow\_stats\_request$ messages. The feedback from the OpenFlow switches is sent over the $ofp\_flow\_stats\_reply$ messages. All these statistics are correlated and sent to the IDS to analyse and detect any anomalies. The administrator is then notified to make an appropriate decision, e.g. to update flow tables to drop communications originating at the compromised machine.

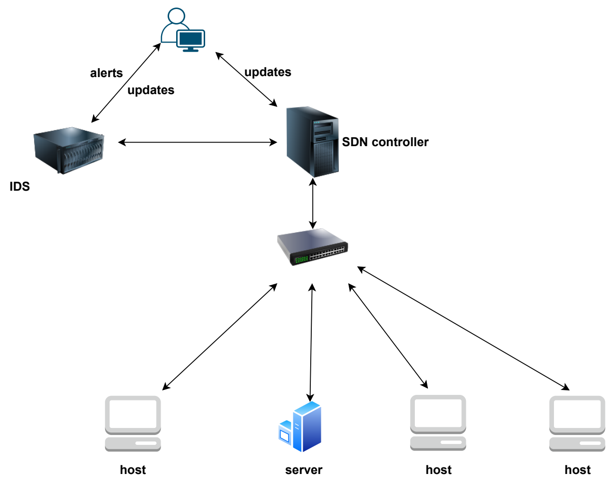

Figure 2: System Architecture

4.1 eXtreme Gradient Boosting (XGBoost)

XGBoost is a boosting ensemble classifier combining decisions from different models. It comprises a sequence of models where every

Table 1: Related works Comparison

Scheme	SDN	Stealth	Accuracy	Dataset Evaluation	Technique
[32]	×	×	75.75%	NSL-KDD	DL
[33]	×	×	89%	NSL-KDD	GRU-RNN
[11]	×	×	87%	NSL-KDD	GRU-LSTM
[24]	✓	×	90.5%	InSDN	OC-SVM-LSTM-Autoencoder
[18]	✓	×	98.60%	CICIDS2017	LSTM-CNN
Our proposed model	✓	✓	98%	APT-SDN dataset	XGboost-OC-SVM

* **DL**: Deep Learning; **GRU**: Gated Recurrent Unit; **RNN**:Recurrent Neural Network; **LSTM**: Long Short Term Memory; **OC-SVM**: One-Class SVM

(non-initial) model makes a decision based on the errors of the previous models, aiming to correct incorrect predictions. (Models are trained sequentially.) It is well known for its execution speed and high performance. In our experiments, the model is trained over labelled training datasets, which include attacks and normal traffic, and is evaluated over the test sets.

4.2 One-Class SVM (OC-SVM)

An anomaly-detection algorithm profiles normal behaviour to form a model of that behaviour. Its model defines an envelope of acceptable behaviour and data outside this envelope are considered suspicious. An OC-SVM is a significant option when non-malicious data dominates a dataset (as is the case with stealth scanning attacks). We applied OC-SVM to the normal portion of the training set but evaluated it over the whole testing set (containing normal and malicious behaviours).

4.3 Model Parameters

Hyper-parameter tuning is the process of finding the parameters for which the model gives its best performance. We seek to avoid over-fitting, where the model is tuned too tightly to the training data and performs well on it, but does not perform well on new data. (We say that the model does not generalise). We also seek to avoid under-fitting, where the model performs poorly both on the training data and new data [6], because it fails to fully exploit information in both sets. In a One-Class SVM, the parameter $\nu \in (0, 1]$ plays a major role in the trade-off between generalisation and over-fitting. Fig 3 shows the effects of tuning the ν value. Another parameter in OC-SVM is $\gamma \in (0, 1]$. In XGBoost the maximum number of features $max\_features$, the number of trees $en\_estimators$, and the maximum depth of every tree max_depth are the most important parameters. Employing a wide range of experiments for different parameters for every classifier, allows a high preforming model to be identified. Table 2 presents the selected parameters to be used in the proposed model.

5 IMPLEMENTATION AND EVALUATION

In our experiments we use APT-SDN dataset [3] which is the most recent SDN dataset. The dataset contains packets of stealth scans aiming to reconstruct flow rules. As mentioned in section 3, the stealth scanner tool APT-SDNmap was used to launch attacks. (An insider has installed the tool and starts doing the scan.) There are three different network sizes available. The first consists of four hosts (h = 4), one SDN switch and one SDN controller. The second consists of eight hosts (h = 8), one SDN switch and one SDN controller. The third has a similar architecture but with sixteen hosts (h = 16). The system architecture is very similar to Figure2 but excludes the IDS.

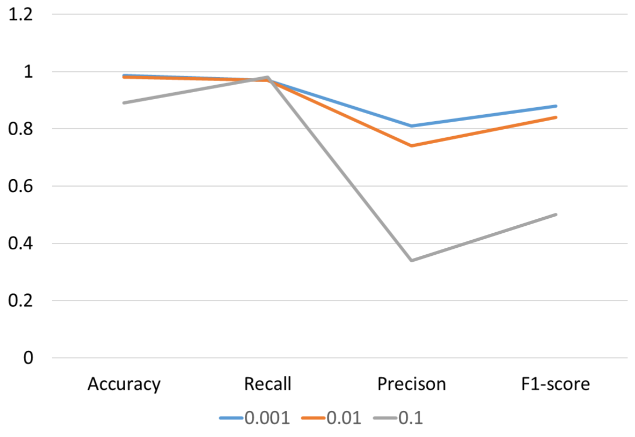

Figure 3: ***nu*** **value tuning**

Table 2: Hyper-parameter tuning

parameter	description	optimal value
eXtreme Gradient Boosting(XGBoost)		
max_features	Maximum number of features in every single run	sqrt
en_estimators	Number of trees	200
max_depth	Maximum depth for any tree	7
One-Class SVM (OC-SVM)		
Kernel	Transforming data function	rbf
ν	This controls the fraction of outliers in the system	0.001
Gamma	Kernel coefficient	0.9

5.1 Preparation and Pre-processing

We selected fourteen features from the dataset after eliminating those that could expose the identity of devices. table 3 shows the selected features to be used in our experiments. We then standardise the dataset features using the scikit-learn object *StandardScaler*.

Principal component analysis (PCA) is used to synthesise new features. These new features are linear combinations of the current raw features and are information rich, i.e. they can inform classification more strongly than raw features. Such features do not exhibit the redundancy of the raw features and fewer of them are needed to effect high-performing classification. Using such information-rich features also decreases the amount of computation involved in classification [16]. Our experiments show that four such features (also referred to as principal components) are sufficient to represent the whole dataset. Figure 5 indicates how much the dataset can be explained by increasing numbers of components. Experiments with more than four components give broadly similar results to those obtained using four components. Thus, four components are used in all experiments presented here. Fig .4 illustrates how these components have a significant impact on the classification model.

The dataset is split into 70% for training and 30% for testing. Because the dataset is imbalanced (malicious scanning transactions are rare), we apply SMOTE (synthetic minority oversampling technique) [30] on the training set to make the number of samples in each of the two classes (denoted by 0 and 1) relatively equal. Figure 1 illustrates the preparation and pre-processing phases and how the two ML models (XGBoost and One-class SVM) work together to investigating attacks.

Table 3: APT-SDNmap dataset features

Feature	T	Description
Protocol	N	The type of protocol
TCP	B	TCP or not
UDP	B	UDP or not
ICMP	B	ICMP or not
IGMP	B	IGMP or not
ARP	B	ARP not
src_host_count	F	The average number of packets initiated by the source
src_ARP_count	F	The average number of ARP packets initiated by the source
src_ICMP_count	F	The average number of ICMP packets initiated by the source
src_IGMP_count	F	The average number of IGMP packets initiated by the source
src_TCP_count	F	The average number of TCP packets initiated by the source
src_UDP_count	F	The average number of UDP packets initiated by the source
diff_proto_rage	F	The average number of different protocols used by the source
diff_dst_port	F	The average number of different ports used by the source
Label	B	Malicious or benign

T: feature type; **N**: Nominal; **B**: Binary; **F**: Float;

5.2 Evaluation Metrics

Different standard metrics are determined to give a comprehensive evaluation and benchmarking of the model with other works. These metrics are *Accuracy*, *Recall*, *Precision*, *F-measure (F1)*, *True Positive Rate (TPR)*, *False Positive Rate (FPR)*, *False Negative Rate (FNR)*, and *True Negative Rate (TNR)*. Four main measures are used in the calculation of these metrics:

- **True Positive (TP)**: the number of instances of malicious transactions classified correctly (as malicious).
- **True Negative (TN)**: the number of instances of benign transactions classified correctly (as benign)
- **False Positive (FP)**: the number of instances of benign transactions classified incorrectly (as malicious)
- **False Negative (FN)**: the number of instances of malicious transactions being classified incorrectly (as benign)

These evaluation metrics are calculated as following:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

$$FNR = \frac{FN}{TP + FN} \tag{7}$$

$$TNR = \frac{TN}{TN + FP} \tag{8}$$

5.3 Results and Discussion

In this work we have conducted two separate experiments on the two main ML detection techniques to find the optimal technique from every type to be employed in the proposed model, followed by experiments on the proposed hybrid scheme on different network sizes included in the dataset.

- **Signature-based detection technique:** We have evaluated the most popular supervised-learning algorithms in ML. Table 4, presents the results for Logistic Regression, K-nearest Neighbour, Decision Tree, Random Forest, Naive Bayes, Support Vector Machine and XGBoost (eXtreme Gradient Boosting). All experiments used the $h = 4$ dataset. XGBoost outperforms all other techniques. Consequently, it is proposed as the classifier for the signature-based detection.
- **Anomaly-based detection techniques:** The most popular One-class classifications (ML-based anomaly detection approaches when there are just two classes *normal* and *abnormal*) are the One-class SVM, Isolation Forest (IF) and Local Outlier Factor (LOF) techniques [7]. We evaluated these

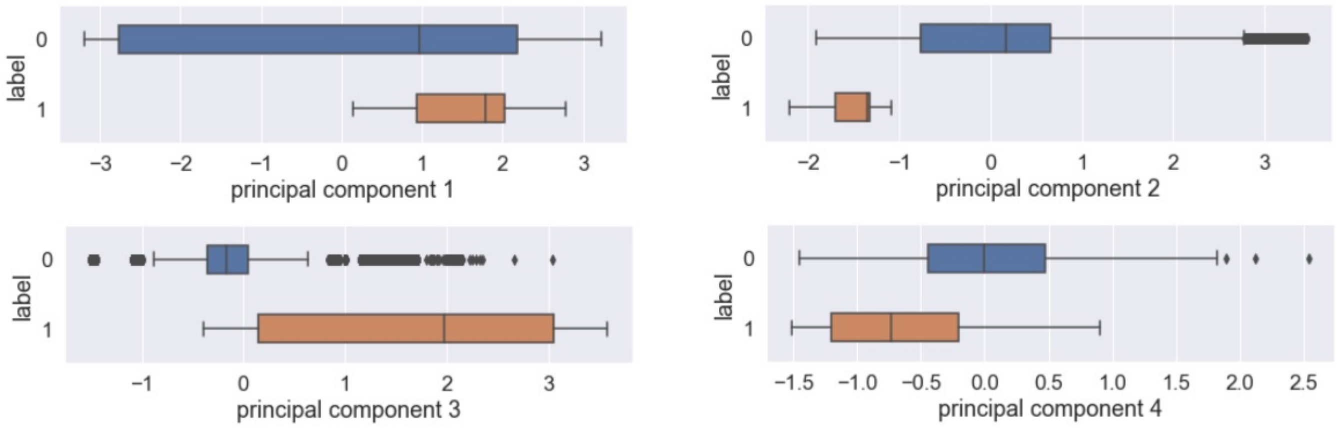

Figure 4: Analytical overview of the selected four components

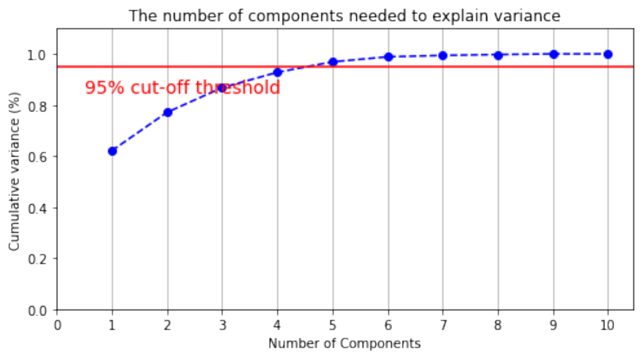

Figure 5: PCA

techniques over the dataset when the network size $h = 4$. The One-Class SVM give far better results than the others. Although, the Isolation Forest shows better results than Local Outlier Factor, both suffer from high numbers of false positives and false negatives compared to One-Class SVM, as shown in the confusion matrices in Figure 6. The promising results for the OC-SVM (One-Class SVM) justify its use as the anomaly detector in the proposed scheme. Table 5 shows the comparison between these techniques.

- **Hybrid Intrusion Detection System:** The proposed hybrid IDS, employing OC-SVM and XGBoost, is evaluated over the three network sizes ($h = 4$, $h = 8$, and $h = 16$) and results are shown in Table 6. The proposed model shows very promising results and is scalable when the network size is increased. The proposed model outperforms all the standard anomaly detection techniques, as shown in Table 5 .

These results are highly creditable. As mentioned in section 1.3, targeting insider malfeasance is the most challenging goal for the defender. Furthermore, if defences against external attackers fail and nodes are compromised, malware may be installed and run as a de facto insider anyway. Our proposed hybrid model exhibits a significant and practical detection capability. We would recommend the insider perspective to any researchers in intrusion detection. Furthermore, applying optimal hyper-parameter tuning and dimensionality reduction brings significant benefits.

Table 4: Supervised-learning results

Technique	Accuracy	Recall	Precision	F1-score
Logistic Regression	0.96	0.36	0.86	0.51
K-nearest neighbour	0.99	0.94	0.99	0.96
Decision Tree	0.99	0.85	0.98	0.91
Random Forest	0.99	0.90	0.99	0.94
Support Vector Machine	0.98	0.80	0.86	0.83
XGBoost	0.99	0.96	0.99	0.97

Table 5: Anomaly-based Detection Techniques Comparison

Tech	Acc	Rec	Pre	F1	TPR	FPR	TNR	FNR
OC-SVM	0.98	0.97	0.81	0.88	0.97	0.01	0.98	0.02
Isolation Forest	0.89	0.75	0.30	0.43	0.75	0.10	0.89	0.24
Local Outlier Factor	0.87	0.28	0.16	0.20	0.28	0.08	0.91	0.71

* **OC-SVM**: One-Class SVM; **Acc**: Accuracy; **Rec**: Recall; **Pre**: Precision;

Table 6: The Hybrid NIDS Experimental results

No. of hosts	Acc	Rec	Pre	F1	TPR	FPR	TNR	FNR
4	0.98	0.98	0.90	0.94	0.99	0.01	0.98	0.006
8	0.98	0.98	0.91	0.94	0.99	0.01	0.98	0.003
16	0.98	0.98	0.88	0.92	0.99	0.01	0.98	0.006

* **Acc**: Accuracy; **Rec**: Recall; **Pre**: Precision;

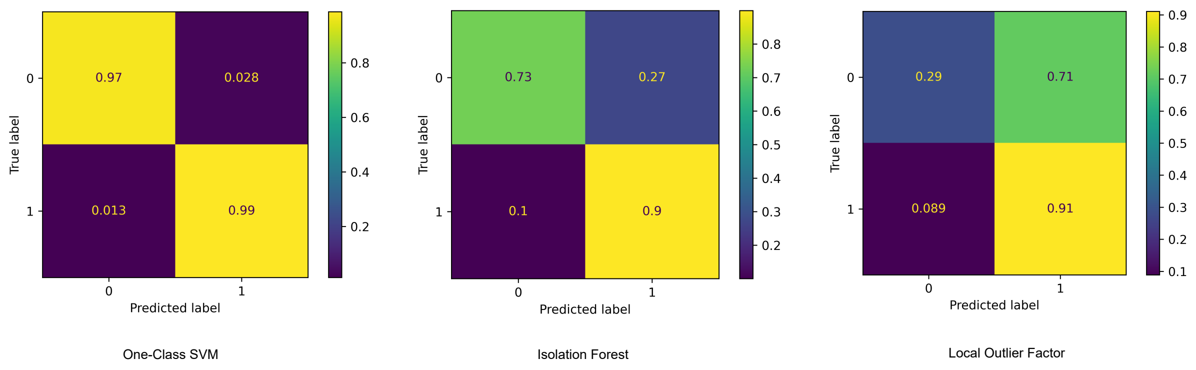

Figure 6: Confusion Matrix for OC-SVM, IF, and LOF

6 CONCLUSIONS AND FUTURE WORK

Detecting stealth attacks in Software-Defined Networks is a critical security endeavour for the community. Insiders have a natural degree of stealth but can additionally adopt strategies that make their detection even harder. In this paper, scanning attacks were launched from compromised SDN nodes (i.e. nodes within the network) that limited the rate at which scanning messages were sent. Such strategies have been shown to evade detection by traditional anti-malware systems. The specific scanning attacks were geared to the reconstruction of SDN flow rules. We proposed a highly effective hybrid model, overcoming the limitations of the individual techniques, improving the detection methodology and reducing erroneous decisions. XGBoost and OC-SVM were implemented to detect known and unknown types of attacks, respectively. Hyperparameter tuning and dimensionality reduction were beneficially adopted. We are currently seeking to enhance the set of features we work with. In particular, we are developing more history-based features evaluated over either the full operational lifetime to date or within a recent time window.

REFERENCES

[1] 2021. Forensic methodology report: How to catch nso group's pegasus. https://www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-nso-groups-pegasus/
[2] Stefan Achleitner, Thomas La Porta, Trent Jaeger, and Patrick McDaniel. 2017. Adversarial network forensics in software defined networking. In *Proceedings of the Symposium on SDN Research*. 8–20.
[3] APT-SDN dataset 2022. APT-SDN dataset. https://github.com/APT-SDNdataset.
[4] APT-SDNmap 2022. APT-SDNmap. https://github.com/APT-SDNmap.
[5] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Mark Felegyhazi. 2012. The cousins of stuxnet: Duqu, flame, and gauss. *Future Internet* 4, 4 (2012), 971–1003.
[6] Jason Brownlee. 2016. *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch* (v1.1 ed.). Machine Learning Mastery.
[7] Jason Brownlee. 2020. *Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning.* Machine Learning Mastery.
[8] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.
[9] Mauro Conti, Fabio De Gaspari, and Luigi V Mancini. 2018. A novel stealthy attack to gather SDN configuration-information. *IEEE Transactions on Emerging Topics in Computing* 8, 2 (2018), 328–340.
[10] Jacob H Cox, Russell J Clark, and Henry L Owen. 2016. Leveraging SDN for ARP security. In *SoutheastCon 2016*. IEEE, 1–8.
[11] Samrat Kumar Dey and Md Mahbubur Rahman. 2018. Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method. In *2018 4th international conference on electrical engineering and information & communication technology (iCEEiCT)*. IEEE, 630–635.
[12] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D Jurcut. 2020. InSDN: A novel SDN intrusion dataset. *IEEE Access* 8 (2020), 165263–165284.
[13] Nicolas Falliere, Liam O Murchu, and Eric Chien. 2011. W32. stuxnet dossier. *White paper, symantec corp., security response* 5, 6 (2011), 29.
[14] Zarestel Ferrer and Methusela Cebrian Ferrer. 2010. In-depth analysis of hydraq. *The face of cyberwar enemies unfolds. ca isbu-isi white paper* 37 (2010).
[15] Stamatis Karnouskos. 2011. Stuxnet worm impact on industrial cyber-physical system security. In *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 4490–4494.
[16] Max Kuhn and Kjell Johnson. 2020. *Feature Engineering and Selection: A practical Approach for Predictive Models.* Taylor Francis Group, 6000 Broken Sound Parkway NW, Suite 300.
[17] Liu Liu, Olivier De Vel, Qing-Long Han, Jun Zhang, and Yang Xiang. 2018. Detecting and preventing cyber insider threats: A survey. *IEEE Communications Surveys & Tutorials* 20, 2 (2018), 1397–1417.
[18] Jahanzaib Malik, Adnan Akhunzada, Iram Bibi, Muhammad Imran, Arslan Musaddiq, and Sung Won Kim. 2020. Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in SDN. *IEEE Access* 8 (2020), 134695–134706.
[19] Mohammad Mamun and Kevin Shi. 2021. DeepTaskAPT: Insider APT detection using Task-tree based Deep Learning. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 693–700.
[20] NSL-KDD dataset 2022. NSL-KDD dataset. http://www.unb.ca/cic/datasets/nsl.html.
[21] Jason RC Nurse, Oliver Buckley, Philip A Legg, Michael Goldsmith, Sadie Creese, Gordon RT Wright, and Monica Whitty. 2014. Understanding insider threat: A framework for characterising attacks. In *2014 IEEE security and privacy workshops*. IEEE, 214–228.
[22] Costin Raiu, Mohamad Amin Hasbini, Sergey Belov, and Sergey Mineev. 2017. From Shamoon to Stonedrill-Wipers attacking Saudi organizations and beyond. *Kaspersky Lab, March* (2017).
[23] Ethan M Rudd, Andras Rozsa, Manuel Günther, and Terrance E Boult. 2016. A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions. *IEEE Communications Surveys & Tutorials* 19, 2 (2016), 1145–1172.
[24] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. 2020. Network anomaly detection using LSTM based autoencoder. In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. 37–45.
[25] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 1999. Support vector method for novelty detection. *Advances in neural information processing systems* 12 (1999).
[26] SDNMap 2022. SDNMap. https://github.com/sdnmap.
[27] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1 (2018), 108–116.
[28] Zhaogang Shu, Jiafu Wan, Di Li, Jiaxiang Lin, Athanasios V Vasilakos, and Muhammad Imran. 2016. Security in software-defined networking: Threats and countermeasures. *Mobile Networks and Applications* 21, 5 (2016), 764–776.

[29] Abhishek Singh and Zheng Bu. 2013. HOT KNIVES THROUGH BUTTER: Evading File-based Sandboxes. *FireEye* (2013).

[30] SMOTE 2022. SMOTE. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html.

[31] W Symantec. 2011. Advanced persistent threats: A symantec perspective. *Symantec World Headquarters* (2011).

[32] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. 2016. Deep learning approach for network intrusion detection in software defined networking. In *2016 international conference on wireless networks and mobile communications (WINCOM)*. IEEE, 258–263.

[33] Tuan A Tang, Lotfi Mhamdi, Des McLeron, Syed Ali Raza Zaidi, and Mounir Ghogho. 2018. Deep recurrent neural network for intrusion detection in sdn-based networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 202–206.

[34] SKYWIPER ANALYSIS TEAM. 2012. *sKyWIper: A complex malware for targeted attacks*. CrySyS Lab Technical Report. Laboratory of Cryptography and System Security (CrySyS Lab), Budapest.

[35] Changhoon Yoon, Taejune Park, Seungsoo Lee, Heedo Kang, Seungwon Shin, and Zonghua Zhang. 2015. Enabling security functions with SDN: A feasibility study. *Computer Networks* 85 (2015), 19–35.

[36] Mingli Yu, Ting He, Patrick McDaniel, and Quinn K Burke. 2020. Flow table security in SDN: Adversarial reconnaissance and intelligent attacks. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1519–1528.

[37] Yadong Zhou, Kaiyue Chen, Junjie Zhang, Junyuan Leng, and Yazhe Tang. 2018. Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense. *Security and Communication Networkss* 2018 (2018), 1–15.