FISTS: A Field-based Security Testing Tool for Updates in Software-Defined Networks

Jahanzaib Malik
University of Luxembourg
Luxembourg, Luxembourg
jahanzaib.malik@uni.lu

Fabrizio Pastore
University of Luxembourg
Luxembourg
fabrizio.pastore@uni.lu

ABSTRACT

We present FISTS, a tool that enables security testing of configuration updates in software-defined networks (SDNs). In contrast to conventional approaches, which are model-based and coupled to a specific SDN system, FISTS is entirely black-box thus enabling testing of different platforms, which is necessary for most industries since they typically use proprietary SDN frameworks. FISTS works by probing the hosts on a network before and after an SDN reconfiguration, automatically identifying corresponding nodes, and determining their port state change; finally, it leverages anomaly detection algorithms to prioritize the inspection of hosts data. FISTS also enable engineers to minimize the number of inspected nodes while maximizing the vulnerabilities found by avoiding the inspection of consecutive false alarms.

We have evaluated FISTS on 300 new and unique datasets based on distinct configuration scenarios; our results show that FISTS leads to best results (up to 0.99 recall) with COF and HBOS, but KNN leads to close recall and minimal execution time. Further, it enables detecting all vulnerable hosts by inspecting less than 10% of the monitored data.

A demo of FISTS is available online at the following URL: https://youtu.be/UIMwFqAvAXg. The tool is available (open source) at: https://doi.org/10.5281/zenodo.18201382.

CCS CONCEPTS

• **Software and its engineering** → **Software maintenance tools**; **Software verification and validation**; • **Security and privacy** → **Network security**.

ACM Reference Format:
Jahanzaib Malik and Fabrizio Pastore. 2026. FISTS: A Field-based Security Testing Tool for Updates in Software-Defined Networks. In *34th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE Companion '26), July 5–9, 2026, Montreal, QC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3803437.3806399

1 INTRODUCTION

Modern communication sectors, such as satellite communications (SATCOM) and telecommunications (TELCO) are vital to modern society; requiring constant innovation to deliver advanced services for demanding users while managing operational costs. To address these challenges, a key capability is the rapid reconfiguration of network components for multiple purposes. This flexibility is largely enabled by Software-Defined Networks (SDNs) [23], which, by replacing hardware components (e.g., routers, firewall) with pure software ones, allow for dynamic network architecture adjustments.

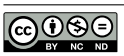
However, the interaction between various applications such as firewalls and routers can lead to inconsistencies or conflicts in configurations. Such situations fall under what we refer to as *SDN misconfigurations*, and we focus on misconfiguration affecting security requirements, which are particularly critical because of the sensible applications of SDNs in communications systems.

To address these issues, we introduce Field-based Security Testing of SDN Configuration Updates (FISTS), a tool designed to perform black-box testing of SDN systems by implementing an homonymous approach presented in a previous publication [14]. It employs network mapping techniques using Nmap [12], an open-source scanning utility, before and after configuration updates. It integrates a dedicated algorithm to match hosts before and after a reconfiguration, and then generates, for each host, a summary of port state changes (e.g., going from closed to open). Such information is then processed through dedicated anomaly detection algorithms which are used to prioritize the inspection of hosts, with the objective to inspect all vulnerable hosts first. Further, it integrates a heuristic to minimize the number of hosts being inspected (we call it *hosts selection*); specifically, it suggests stopping the inspection of hosts when an empirically determined number of consecutive false positives is observed.

Our previous publication includes an extensive evaluation of FISTS with 220 datasets (network scans) collected in industrial contexts; our results suggest that the K-Nearest Neighbour algorithm shall be used to obtain best results for hosts selection (up to 0.95 precision and 1.00 recall), further, it enables detecting all vulnerable hosts by inspecting less than 10% of the monitored data [14]. Further, an additional extensive empirical assessment of several anomaly detection algorithms, performed on 300 datasets and described in a recent PhD thesis [13] suggests that other algorithms (e.g., OCSVM and Isolation Forest) perform similar to KNN.

FISTS is released as open source [15]; compared to the replication packages of our previously published work, which enable the replication of results using dataset CSV files, this paper introduces a tool that can be used by engineers on raw NMAP results.

The remainder of this paper proceeds as follows: Section 2 describes related work, Section 3 explains the FISTS approach, Section 4 provides an overview of the FISTS architecture, Section 5 provides an overview of our empirical assessment, Section 6 concludes the paper.



FSE Companion '26, July 5–9, 2026, Montreal, QC, Canada

ACM ISBN 979-8-4007-2636-1/26/07.
https://doi.org/10.1145/3803437.3806399

2 RELATED WORK

Related approaches include model-based verification techniques for SDN configurations and fuzz testing approaches.

Model-based verification involves two primary strategies: first, employing model-checking techniques to detect potential conflicts in the system's configuration [17]; and second, generating inputs such as network packets based on the configuration itself. For instance, requesting access to a whitelisted URL can help verify if the SDN allows the expected outcome [8]. However, one significant limitation of these methods is the reliance on system models, which must be either manually created by engineers or derived from configurations provided to the SDN software. Both processes are resource-intensive: manual modeling demands skilled personnel, while automated derivation requires tight integration with SDN tools, such as those using proprietary formats like Versa [22]. This integration can be costly and complex to maintain. Moreover, model-based verification approaches that depend on model checking have additional constraints. They necessitate a detailed system model, meaning any unmodeled aspects are not tested. Additionally, they may fail to detect issues that emerge during system execution due to dynamic factors like transmission delays.

Given these limitations, automated testing approaches that do not depend on SDN modeling could be more suitable for our research problem. In this context, fuzz testing has been explored as a potential solution. Tools such as DELTA [9], BEADS [5], FragScapy [3], and AFLNET [16] are notable examples of fuzzers used in SDN contexts. However, they primarily focus on generating anomalous traffic or identifying failures due to malicious components within the network (e.g., applications or controllers), which is not our objective.

3 OVERVIEW OF THE FISTS APPROACH

FISTS can detect security vulnerabilities caused by erroneous configurations resulting in unexpected port states. An example is illustrated in Figure 1. The original SDN configuration shows a Server Message Block (SMB) server operating on port 445 connected to switch 3, with a high-priority flow rule on switch 2 that forwards all traffic for port 445 to switch 3. This ensures confidentiality by routing the SMB-related traffic only to the subnet containing the SMB server. However, in an update, engineers relocate the SMB server to a dedicated switch (switch 6) but fail to update the corresponding rule on switch 2 to forward SMB traffic to the new destination switch 6 instead of switch 3. As a result, all SMB-related traffic is misdirected to the wrong subnet, posing significant confidentiality risks even if packets may be redirected to reach switch 6 (not depicted in the figure). This highlights the importance of proactive SDN flows, where packet routing is preconfigured, as reactive SDN flows (where rules are created as packets arrive at a switch) cannot prevent such mistakes due to confidentiality constraints preventing uncontrolled packet routing. FISTS works in five steps, depicted in Fig.1 and described in the following.

3.1 FISTS Steps 1 and 2: Traffic Generation and Response Monitoring

The FISTS methodology relies on network reconnaissance through two sequential processes. First, generate network traffic to gather initial data about the network (Step 1). Second, collect and analyze responses from network hosts to identify open ports. For both steps, engineers are expected to leverages Nmap due to its comprehensive scanning capabilities. Notably, Nmap's TCP SYN and UDP scan options are employed to ensure thorough port state identification. Note that both Step 1 and Step 2 are performed twice, before and after an SDN reconfiguration.

3.2 FISTS Step 3: Host Matching and Comparison

In Step 3, the Host Matching and Comparison (HMC) module, integrated into the FISTS tool, processes the data collected before and after reconfiguration. This step aims at two primary objectives: first, identifying which data entries correspond to the same host across both configurations; second, determining any changes in port states for each host. It is crucial to recognize that SDN reconfigurations can alter a host's IP or MAC address without affecting other attributes. Failing to account for such changes could lead to misinterpretation of port state variations. For instance, if the IP or MAC address of a server changes during reconfiguration, simply comparing port states might not be sufficient to detect these modifications accurately. To address this challenge, we have developed a greedy algorithm designed to match hosts correctly in the updated configuration. This algorithm compares data entries from the initial and updated configurations systematically. Specifically, it performs $n \times m$ comparisons between the n data entries from the original configuration and the m data entries from the updated configuration. For each pair of data entries, the algorithm computes the degree of likelihood for their match, based on the following formula:

$$Combined\_Score = ip\_score + mac\_score + ports\_score$$

Based on previous successful assessments [14], the default value of ip_score is set to 0.20 when two IPs match, otherwise it is set to 0. The value of mac_score is set to 0.20 when MAC addresses match, otherwise it is set to 0. The value of $ports\_score$ is calculated by multiplying the proportion of matching port states by 0.60. All pairs are sorted based on matching score until all hosts are considered resulting in a Network State Change Report (NSCR). The NSCR is a tabular report where each row provides information about the state change for one host, it includes the IP and MAC addresses before and after the reconfiguration, two columns indicating whether MAC and IP were modified by the reconfiguration, and columns capturing state change information for each port (e.g., ClosedToOpen). An example NSCR is shown in Figure 1.

3.3 FISTS Step 4: Preprocessing

In Step 4, we preprocess the NSCR data to enable the application of anomaly detection algorithms. Specifically, we leverage one-hot encoding (OHE) [7] to transform each categorical variable into a set of binary variables, where each unique category is represented by its own column in the dataset (e.g., "1" or "0"). For our network port state data, this results in 16 columns for each TCP and UDP port. These columns correspond to all possible combinations of states that can occur during network operations.

Table 1: Prioritized NCSR report example

IP'	IP"	tcp_Port 22	udp_Port 22	tcp_Port 445	udp_Port 445	tcp_Port 8080	...	udp_Port 5674	Anomaly Score
10.0.1.12	10.0.2.10	ClosedToClosed	ClosedToClosed	**OpenToFiltered**	ClosedToClosed	ClosedToClosed	...	ClosedToClosed	1.37
10.0.1.11	10.0.1.111	ClosedToClosed	ClosedToClosed	ClosedToClosed	ClosedToClosed	**OpenToOpen**	...	ClosedToClosed	1.33
10.0.1.18	10.0.1.18	ClosedToClosed	ClosedToClosed	ClosedToClosed	ClosedToClosed	ClosedToClosed	...	**OpenToOpen**	1.04
10.0.1.56	10.0.1.56	ClosedToClosed	ClosedToClosed	ClosedToClosed	ClosedToClosed	ClosedToClosed	...	**OpenToOpen**	1.00
10.0.3.10	10.0.3.10	ClosedToClosed	ClosedToClosed	ClosedToClosed	ClosedToClosed	ClosedToClosed	...	ClosedToClosed	0.00
...	...	...	...	...	...	...	...	...	...

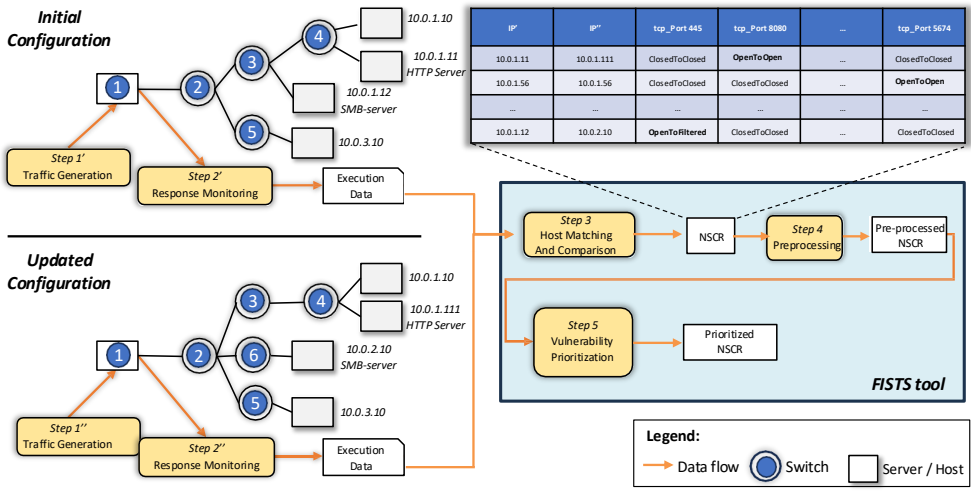


Figure 1: Overview of the FISTS methodology and tool. It shows the FISTS steps along with example outputs. The left side shows a network being tested by FISTS before and after reconfiguration. The NSCR output is exemplified by the provided table.

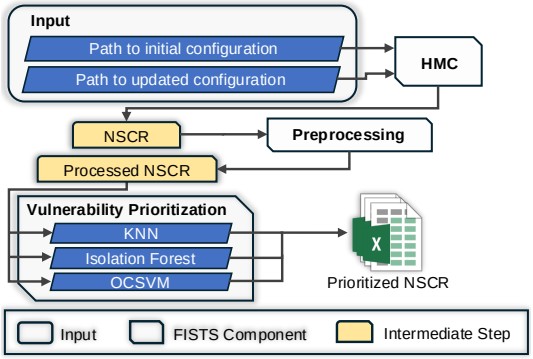


Figure 2: Overview of the FISTS Tool Architecture.

Additionally, Step 4 may include an optional sub-step (*pruning*) aimed at optimizing the dataset by removing redundant entries. Specifically, we prune records where no change occurs between consecutive updates (e.g., hosts whose configuration remains unchanged). Pruning can help reducing unnecessary data and improves computational efficiency [13].

3.4 FISTS Step 5: Vulnerability Prioritization

In Step 5, FISTS sorts all entries in the NSCR according to their likelihood of being vulnerable (i.e., they correspond to a misconfigured host), using the anomaly score generated by an anomaly detection algorithm applied to the NSCR. An example of sorted NSCR is illustrated in Table 1

Our key intuition is prioritizing hosts based on their anomaly scores, allowing engineers to inspect the most likely vulnerable ones first after a configuration update. Further, if engineers need to minimize the number of inspected hosts, FISTS can be used to perform *hosts selection*, specifically, they can stop inspecting hosts when the number of observed consecutive false positives (i.e., hosts that are not vulnerable) reaches an empirically determined stopping condition (SC). In practice, engineers determine false positives based on their expectations for the configuration update analyzed: for each host, they look at the port state and the port state change, if they are consistent with expectations (e.g., no unexpected open port) it's a false positive because the host shall not have been reported among the most suspicious ones.

4 TOOL ARCHITECTURE

Step 1 and Step 2, which involve information gathering using Nmap are not incorporated into the FISTS tool because users may leverage different Nmap setups, based on their network and constraints (e.g., scanning the whole network from one node, or multiple nodes). Therefore, the FISTS tool has three main components, which we call *HMC*, *preprocessing*, and *vulnerability prioritization*; they are illustrated in Figure 2.

The *HMC component* takes as input two xml files, which are the result of the network scans performed by Nmap on the initial and updated SDN. It then saves the NSCR in CSV format.

The *Preprocessing component* further processes NCSR and performs OHE and pruning (if select by the end-user) thus resulting into a *preprocessed NCSR*.

The preprocessed NSCR is taken as input by the vulnerability prioritization component, which executes the anomaly detection algorithm selected by the end-user and generates the *prioritized NSCR*. Specifically, it generates one prioritized NCSR stored as a CSV file, for each algorithm selected by the end-user. The FISTS tool currently supports the prioritization of hosts based on the anomaly detection algorithms that performed best in our empirical assessments [13, 14], they are: K-Nearest Neighbor (KNN) [4], Isolation Forest (IF) [10], Histogram-based Outlier Score (HBOS) [2], One-Class Support Vector Machines (OCSVMs) [18], Connectivity-Based Outlier Factor (COF) [21] .

5 EMPIRICAL EVALUATION

In a first study, we assessed FISTS with six different anomaly detection algorithms to compute the anomaly score [14], which are KNN, IF, Hierarchical Agglomerative Clustering (HAC) [19], K-means clustering (KMC) [6], and Local Outlier Factor (LOF) [1]. Following the first study, we assessed FISTS with seven additional algorithms, including COF, HBOS, and OCSVM [13]. For KNN we considered four configurations for its hyper-parameter K, which captures the number of neighbours to consider when computing the anomaly score. With each algorithm, we assessed different thresholds for the stopping condition (from 1 to 20) and their use with and without pruning. In total, we assessed 260 different FISTS pipelines.

Our assessment has been based on 300 datasets derived as follows. 100 *synthetic* datasets obtained by monitoring eight different network setups (four valid reconfiguration and four invalid ones) and duplicating their data in random combinations till reaching 405 hosts per dataset. 100 *real* datasets were obtained by considering data including valid network configuration collected by Shodan and integrating it with invalid network reconfigurations. 100 *realistic* datasets were obtained by combining real and synthetic data.

Below, we report on three research questions investigated in our work [14]. At a high level, we aim to compare the different pipelines resulting from the configuration parameters of FISTS, which are pruning (enabled/disabled), the stopping condition (SC) for hosts selection, and the anomaly detection algorithm.

RQ1. Does pruning improve FISTS results? This research question investigates whether pruning (i.e., ignoring hosts not affected by any state change during the generation of NSCR) helps to obtain better results. To evaluate the performance of each pipeline for *hosts selection* we use precision and recall. For *hosts prioritization* we compute the Debugging Cost (DC) as the number of hosts to be inspected in order to achieve 100% recall. We assess the usefulness of pruning by discussing the best results observed with and without pruning, for each dataset.

For *hosts selection*, we observe that results vary, with clustering-based approaches (KMC and HAC) performing better with pruning; however, for recall and F1 score, there is no difference for the overall best result observed with and without pruning.

For *hosts prioritization* pruning leads to less hosts to be inspected (lower DC), especially in the presence of a large set of changes (synthetic dataset) but the best overall results is achieved by HBOS without pruning.

RQ2. What FISTS pipeline leads to the best results? This research question aims to identify which pipeline leads to best results. We used the same results from previous research question.

By looking at the averages across datasets, COF with SC=20 and pruning leads to the best result for hosts prioritization (DC=29.58) and second-best for hosts selection (recall=0.995, F1 score=0.73), HBOS with SC=20 and no pruning leads to the best result for hosts selection (recall=0.999, F1 score=0.73) and second-best for hosts prioritization (DC=30.29), but differences are minimal. For hosts selection, OCSVM with SC=20 perform similar to sCOF and HBOS (recall=0.992, F1 score=0.73). KNN and IF present a recall that is still high (i.e., 0.988, without pruning), with IF performing slightly worse for F1 score but better for DC score.

RQ3. How does FISTS scale? We assessed the scalability of each step in the pipeline. For Step 1 and 2 we report that the time taken by Nmap to scan 1000 ports in 405 IPs ranges between 30.26 and 30.35 minutes, which is acceptable since it fits into a typical network maintenance window. Further, the scan can be split across subnets thus reducing the total time required. Steps 3 and 4 take only 1 to 2 seconds, which are minimal and acceptable. For Step 5, we observe that KNN-20 with SC-20 takes approximately 0.005 to 0.025 seconds on average, which is minimal; OCSVM, HBOS, and IF, instead, take between 0.05 and 1.0 seconds thus being 40x more expensive than KNN-20 and potentially less scalable for larger datasets. COF took up to 14 seconds, thus being inappropriate for large networks.

Concluding, for medium sized networks (i.e., 400 nodes, like ours), HBOS, COF, OCSVM, KNN, and IF are appropriate choices leading to accurate results; for larger networks, we suggest leveraging either OCSVM or KNN-20 with SC=20 since they performed sufficiently well for both hosts selection and prioritization.

6 CONCLUSION

We presented FISTS, a novel tool for security testing of network reconfigurations. It leverages a field-based testing approach to enable black-box assessment without the need for network configuration information. The tool leverages Nmap to collect information from hosts before and after the configuration. It integrates a dedicated algorithm to match hosts across the two configurations based on IP, MAC and ports. Finally, it leverages anomaly detection algorithms to prioritize the inspection of hosts.

In our extensive empirical evaluation, we determined that Histogram-based Outlier Score (HBOS) and Connectivity-Based Outlier Factor (COF) lead to best vulnerability detection results but the K-Nearest Neighbor algorithm with 20 Neighbors leads to minimal execution time and vulnerability detection results similar to HBOS and COF. To minimize the number of hosts to be inspected without affecting the recall (i.e., fault detection rate) of the approach, engineers shall stop after observing 20 false positives.

ACKNOWLEDGMENTS

This work has been supported by SES [20] and the Luxembourg National Research Fund (FNR) under the project INSTRUCT (IPBG19/14016225/INSTRUCT [11]).

REFERENCES

[1] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. *SIGMOD Rec.* 29, 2 (may 2000), 93–104. https://doi.org/10.1145/335191.335388

[2] Markus Goldstein and Andreas Dengel. 2012. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track* 1 (2012), 59–63.

[3] Frédéric Guihéry. [n. d.]. *Fragscapy*. https://github.com/AMOSSYS/Fragscapy

[4] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. 2003. KNN Model-Based Approach in Classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, Robert Meersman, Zahir Tari, and Douglas C. Schmidt (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 986–996.

[5] S. Jero, X. Bu, C. Nita-Rotaru, H. Okhravi, R. Skowyra, and S. Fahmy. 2017. BEADS: Automated Attack Discovery in OpenFlow-Based SDN Systems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10453 LNCS (2017), 311–333. https://doi.org/10.1007/978-3-319-66332-6_14

[6] Xin Jin and Jiawei Han. 2010. *K-Means Clustering*. Springer US, Boston, MA, 563–564. https://doi.org/10.1007/978-0-387-30164-8_425

[7] Scikit learn contributors. 2026. *One Hot Encoding*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

[8] D. Lebrun, S. Vissicchio, and O. Bonaventure. 2014. Towards test-driven software defined networking. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. 1–9. https://doi.org/10.1109/NOMS.2014.6838225

[9] Seungsoo Lee, Jinwoo Kim, Seungwon Woo, Changhoon Yoon, Sandra Scott-Hayward, Vinod Yegneswaran, Phillip Porras, and Seungwon Shin. 2020. A comprehensive security assessment framework for software-defined networks. *Computers & Security* 91 (2020), 101720. https://doi.org/10.1016/j.cose.2020.101720

[10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. (2008), 413–422. https://doi.org/10.1109/ICDM.2008.17

[11] Luxembourg National Research Fund. [n. d.]. INSTRUCT - INtegrated Satellite – TeRrestrial Systems for Ubiquitous Beyond 5G CommunicaTions. https://instruct-ipbg.uni.lu/. Last Accessed: 2022.

[12] Gordon Fyodor Lyon. 2009. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, Sunnyvale, CA, USA. Available at http://nmap.org.

[13] Jahanzaib Malik. 2024. *Field-based Security Testing of SDN Configuration Updates*. Ph.D. thesis. University of Luxembourg, Luxembourg. https://orbilu.uni.lu/handle/10993/63497

[14] Jahanzaib Malik and Fabrizio Pastore. 2025. Field-Based Security Testing of SDN Configuration Updates. *IEEE Transactions on Reliability* (2025), 1–15. https://doi.org/10.1109/TR.2025.3531654

[15] Jahanzaib Malik and Fabrizio Pastore. 2026. FISTS source code. https://doi.org/10.5281/zenodo.18201382

[16] V. T. Pham, M. Bohme, and A. Roychoudhury. 2020. AFLNET: A Greybox Fuzzer for Network Protocols. (2020), 460–465. https://doi.org/10.1109/ICST46399.2020.00062

[17] Elisa Rojas, Roberto Doriguzzi-Corin, Sergio Tamurejo, Andres Beato, Arne Schwabe, Kevin Phemius, and Carmen Guerrero. 2018. Are we ready to drive software-defined networks? A comprehensive survey on management tools and techniques. *Comput. Surveys* 51, 2 (2018). https://doi.org/10.1145/3165290

[18] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.

[19] Scikit-learn.org. 2026. https://scikit-learn.org/stable/modules/generated/sk\protect\discretionary{\char\hyphenchar\font}{}{}learn.cluster.AgglomerativeClustering.html.

[20] SES Luxembourg. [n. d.]. Leading Satellite Operator. https://ses.com/. Last Accessed: 2026.

[21] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. 2002. Enhancing effectiveness of outlier detections for low density patterns. In *Advances in Knowledge Discovery and Data Mining: 6th Pacific-Asia Conference, PAKDD 2002 Taipei, Taiwan, May 6–8, 2002 Proceedings 6*. Springer, 535–548.

[22] Versa Networks. 2026. Versa Secure SD-WAN. https://versa-networks.com/products/sd-wan/

[23] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. 2014. A survey on software-defined networking. *IEEE Communications Surveys & Tutorials* 17, 1 (2014), 27–51.

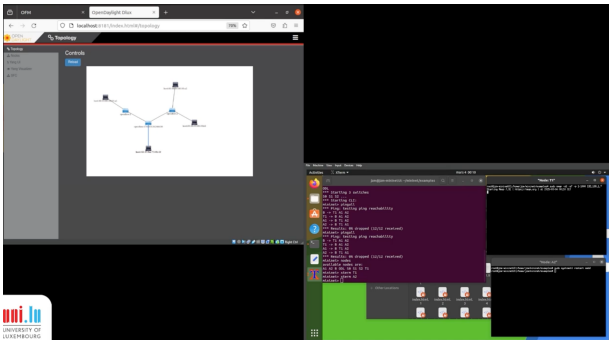


Figure 3: SDN reconfiguration and NMAP execution.

APPENDIX: DEMO SCRIPT

The demonstration will start with an overview of the proposed approach.

We will then simulate two SDN configurations and monitor them with NMAP (Figure 3). The running example will resemble what shown in the tool demo paper:

(1) An SMB server becomes unreachable (filtered port 445) after moving from IP `10.0.1.12` to IP `10.0.2.10` because packets are wrongly forwarded to the old subnet (i.e., `10.0.1.*`). This is the vulnerable host.
(2) The HTTP Server with IP `10.0.1.11` has been assigned with IP `10.0.1.111`.
(3) The FTP Server (port 574) with IP `10.0.1.56` does not show any change (i.e., FTP port remains open).
(4) The HTTP Server (port 8080) with IP `10.0.1.18` does not present any change (i.e., HTTP port remains open).

We will then run FISTS on the xml files collected with NMAP; by default, FISTS runs all its algorithms. We will first visualize (e.g., in the *Visual Studio Code IDE*) the results generated with FISTS by leveraging the KNN algorithm (i.e., file *PNSCR_KNN*, shown in Figure 4).

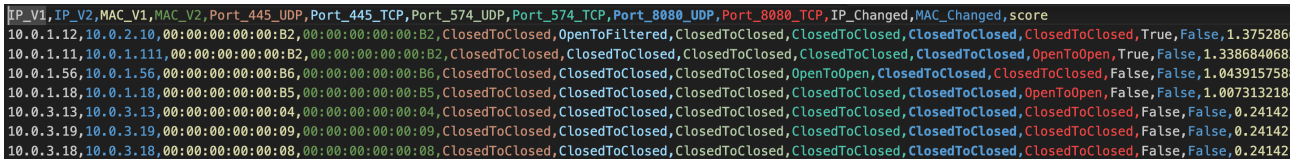
The prioritized NSCR shows the following:

- Host `10.0.1.12` → `10.0.2.10` is correctly the most anomalous (i.e., most vulnerable) because it presents a set of changes not observed in other nodes.
- Host `10.0.1.11` → `10.0.1.111` is second place because it presents a set of characteristics not observed in others: port 8080 remains open and the IP address changes.
- Host `10.0.1.56` → `10.0.1.56`, with port 574 remaining open, is third place because although presenting a characteristic common to others (i.e., port remains open), such characteristic is unique (i.e., no other node has port 574 open.
- Host `10.0.1.18` has a characteristic not common with the majority of hosts (i.e., port remaining open). Since port 8080 remains open also for host `10.0.2.10`, it is less anomalous than host `10.0.1.56`.
- The other hosts (not all shown in Figure 4) present the same anomaly score (i.e., 0.0) because they share the same characteristics (no port open, no other changes). They are not vulnerable.

For host selection, engineers could stop at the first false positive and not inspect what follows host `10.0.1.11`.

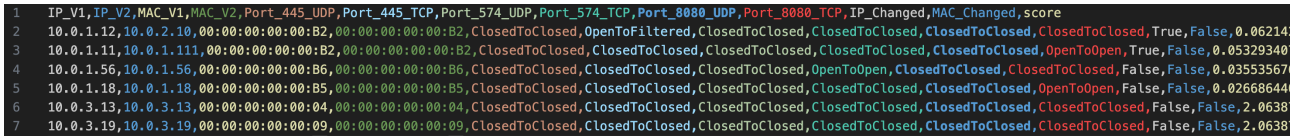
In this simple example, the prioritized results obtained with KNN above are similar to the ones observed with IF, HBOS and OCSVM (i.e., same ranking, as shown in Figures 6 and 8), the only difference is the anomaly score. IF, instead, has a different ranking for the non anomalous nodes, while COF ranks `10.0.1.111` first, thus introducing a false positive (engineers would like to investigate vulnerable nodes first).

The example shown for the demo is delivered with our tool and can be reproduced by the interested reader. The tool also stores a csv file containing only changes in host states (after host matching), which could be used by other researchers to assess new anomaly detection approaches (see Figure 9).



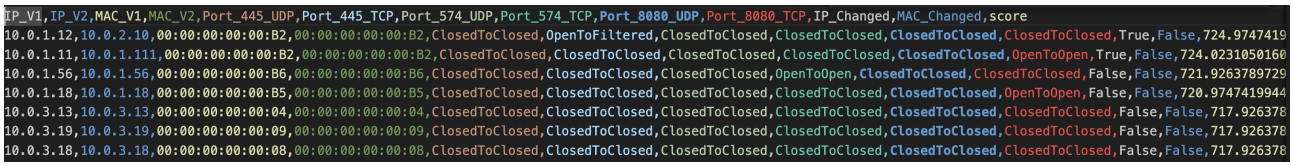
```
IP_V1,IP_V2,MAC_V1,MAC_V2,Port_445_UDP,Port_445_TCP,Port_574_UDP,Port_574_TCP,Port_8080_UDP,Port_8080_TCP,IP_Changed,MAC_Changed,score
10.0.1.12,10.0.2.10,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,OpenToFiltered,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,True,False,1.375286
10.0.1.11,10.0.1.111,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,True,False,1.338684068
10.0.1.56,10.0.1.56,00:00:00:00:00:B6,00:00:00:00:00:B6,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,ClosedToClosed,ClosedToClosed,False,False,1.043915758
10.0.1.18,10.0.1.18,00:00:00:00:00:B5,00:00:00:00:00:B5,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,False,False,1.007313218
10.0.3.13,10.0.3.13,00:00:00:00:00:04,00:00:00:00:00:04,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,0.24142
10.0.3.19,10.0.3.19,00:00:00:00:00:09,00:00:00:00:00:09,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,0.24142
10.0.3.18,10.0.3.18,00:00:00:00:00:08,00:00:00:00:00:08,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,0.24142
```

Figure 4: Prioritized NSCR obtained with KNN.



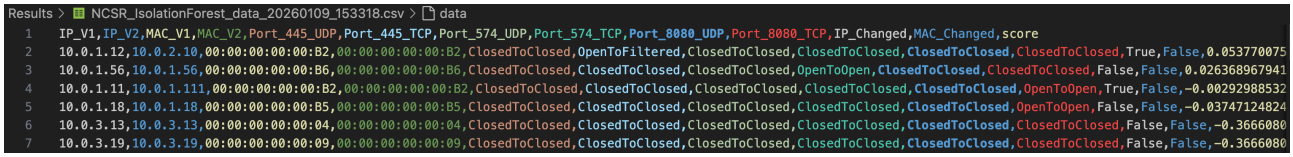
```
IP_V1,IP_V2,MAC_V1,MAC_V2,Port_445_UDP,Port_445_TCP,Port_574_UDP,Port_574_TCP,Port_8080_UDP,Port_8080_TCP,IP_Changed,MAC_Changed,score
10.0.1.12,10.0.2.10,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,OpenToFiltered,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,True,False,0.06214
10.0.1.11,10.0.1.111,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,True,False,0.05329340
10.0.1.56,10.0.1.56,00:00:00:00:00:B6,00:00:00:00:00:B6,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,ClosedToClosed,ClosedToClosed,False,False,0.03553567
10.0.1.18,10.0.1.18,00:00:00:00:00:B5,00:00:00:00:00:B5,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,False,False,0.02668644
10.0.3.13,10.0.3.13,00:00:00:00:00:04,00:00:00:00:00:04,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,2.0638
10.0.3.19,10.0.3.19,00:00:00:00:00:09,00:00:00:00:00:09,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,2.0638
```

Figure 5: Prioritized NSCR obtained with OCSVM.



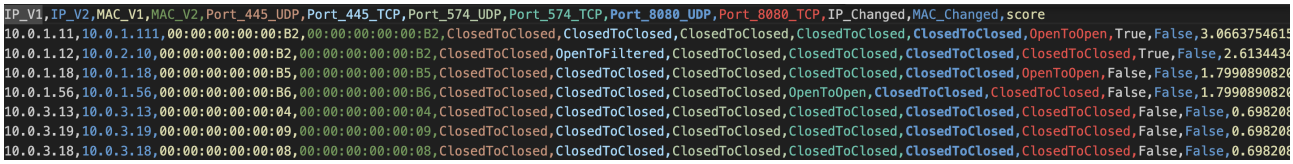
```
IP_V1,IP_V2,MAC_V1,MAC_V2,Port_445_UDP,Port_445_TCP,Port_574_UDP,Port_574_TCP,Port_8080_UDP,Port_8080_TCP,IP_Changed,MAC_Changed,score
10.0.1.12,10.0.2.10,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,OpenToFiltered,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,True,False,724.9747419
10.0.1.11,10.0.1.111,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,True,False,724.0231050160
10.0.1.56,10.0.1.56,00:00:00:00:00:B6,00:00:00:00:00:B6,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,ClosedToClosed,ClosedToClosed,False,False,721.9263789729
10.0.1.18,10.0.1.18,00:00:00:00:00:B5,00:00:00:00:00:B5,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,False,False,720.9747419944
10.0.3.13,10.0.3.13,00:00:00:00:00:04,00:00:00:00:00:04,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,717.926378
10.0.3.19,10.0.3.19,00:00:00:00:00:09,00:00:00:00:00:09,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,717.926378
10.0.3.18,10.0.3.18,00:00:00:00:00:08,00:00:00:00:00:08,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,717.926378
```

Figure 6: Prioritized NSCR obtained with HBOS.



```
Results > NCSR_IsolationForest_data_20260109_153318.csv > data
IP_V1,IP_V2,MAC_V1,MAC_V2,Port_445_UDP,Port_445_TCP,Port_574_UDP,Port_574_TCP,Port_8080_UDP,Port_8080_TCP,IP_Changed,MAC_Changed,score
10.0.1.12,10.0.2.10,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,OpenToFiltered,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,True,False,0.053770075
10.0.1.56,10.0.1.56,00:00:00:00:00:B6,00:00:00:00:00:B6,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,ClosedToClosed,ClosedToClosed,False,False,0.026368967941
10.0.1.11,10.0.1.111,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,True,False,-0.00292988532
10.0.1.18,10.0.1.18,00:00:00:00:00:B5,00:00:00:00:00:B5,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,False,False,-0.03747124824
10.0.3.13,10.0.3.13,00:00:00:00:00:04,00:00:00:00:00:04,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,-0.3666080
10.0.3.19,10.0.3.19,00:00:00:00:00:09,00:00:00:00:00:09,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,-0.3666080
```

Figure 7: Prioritized NSCR obtained with IF.



```
IP_V1,IP_V2,MAC_V1,MAC_V2,Port_445_UDP,Port_445_TCP,Port_574_UDP,Port_574_TCP,Port_8080_UDP,Port_8080_TCP,IP_Changed,MAC_Changed,score
10.0.1.11,10.0.1.111,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,True,False,3.0663754615
10.0.1.12,10.0.2.10,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,OpenToFiltered,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,True,False,2.6134434
10.0.1.18,10.0.1.18,00:00:00:00:00:B5,00:00:00:00:00:B5,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,False,False,1.7990890820
10.0.1.56,10.0.1.56,00:00:00:00:00:B6,00:00:00:00:00:B6,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,ClosedToClosed,ClosedToClosed,False,False,1.7990890820
10.0.3.13,10.0.3.13,00:00:00:00:00:04,00:00:00:00:00:04,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,0.698208
10.0.3.19,10.0.3.19,00:00:00:00:00:09,00:00:00:00:00:09,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,0.698208
10.0.3.18,10.0.3.18,00:00:00:00:00:08,00:00:00:00:00:08,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False,0.698208
```

Figure 8: Prioritized NSCR obtained with COF.

```
IP_V1,IP_V2,MAC_V1,MAC_V2,Port_445_UDP,Port_445_TCP,Port_574_UDP,Port_574_TCP,Port_8080_UDP,Port_8080_TCP,IP_Changed,MAC_Changed
10.0.1.10,10.0.1.10,00:00:00:00:00:B1,00:00:00:00:00:B1,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.10,10.0.3.10,00:00:00:00:00:A1,00:00:00:00:00:A1,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.1.13,10.0.1.13,00:00:00:00:00:B3,00:00:00:00:00:B3,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.1.14,10.0.1.14,00:00:00:00:00:B4,00:00:00:00:00:B4,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.1.15,10.0.1.15,00:00:00:00:00:B5,00:00:00:00:00:B5,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.1.16,10.0.1.16,00:00:00:00:00:B3,00:00:00:00:00:B3,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.1.17,10.0.1.17,00:00:00:00:00:B4,00:00:00:00:00:B4,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.1.18,10.0.1.18,00:00:00:00:00:B5,00:00:00:00:00:B5,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,False,False
10.0.1.56,10.0.1.56,00:00:00:00:00:B6,00:00:00:00:00:B6,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,ClosedToClosed,ClosedToClosed,False,False
10.0.3.11,10.0.3.11,00:00:00:00:00:01,00:00:00:00:00:01,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.12,10.0.3.12,00:00:00:00:00:02,00:00:00:00:00:02,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.13,10.0.3.13,00:00:00:00:00:04,00:00:00:00:00:04,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.14,10.0.3.14,00:00:00:00:00:04,00:00:00:00:00:04,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.15,10.0.3.15,00:00:00:00:00:05,00:00:00:00:00:05,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.16,10.0.3.16,00:00:00:00:00:06,00:00:00:00:00:06,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.17,10.0.3.17,00:00:00:00:00:07,00:00:00:00:00:07,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.18,10.0.3.18,00:00:00:00:00:08,00:00:00:00:00:08,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.3.19,10.0.3.19,00:00:00:00:00:09,00:00:00:00:00:09,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,False,False
10.0.1.11,10.0.1.111,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,OpenToOpen,True,False
10.0.1.12,10.0.2.10,00:00:00:00:00:B2,00:00:00:00:00:B2,ClosedToClosed,OpenToFiltered,ClosedToClosed,ClosedToClosed,ClosedToClosed,ClosedToClosed,True,False
```

Figure 9: NSCR with only the changes in host states, based on NMAP data.