Towards Test Automation for Certification Tests in the Banking Domain

Anıl Elakaş
Fibabanka R&D Center
Istanbul, Türkiye
anil.elakas@fibabanka.com.tr

Ozan Tarlan
Ozyegin University
Istanbul, Türkiye
ozan.tarlan@ozu.edu.tr

Ilgın Şafak
Fibabanka R&D Center
Istanbul, Türkiye
ilgin.safak@fibabanka.com.tr

Kubra Kalkan
Ozyegin University
Istanbul, Türkiye
kubra.kalkan@ozyegin.edu.tr

Hasan Sözer
Ozyegin University
Istanbul, Türkiye
hasan.sozer@ozyegin.edu.tr

2023 8th International Conference on Computer Science and Engineering (UBMK) | 979-8-3503-4081-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/UBMK59864.2023.10286638

***Abstract*—Software systems in the banking domain are business-critical applications that provide financial services. These systems are subject to rigorous certification tests, which are performed manually, and take weeks to complete. In this paper, we suggest that automation of the certificate tests are possible and it will save a considerable amount of time. A certification testing operation which can take a few weeks can be reduced to a few seconds. Firstly, we review the existing test activities to identify the ones that can be automated and introduce a prototype tool for automating some of the tests used for certification. We focus on rules that are verified by analyzing the banking infrastructure. Our tool takes the network topology of the banking infrastructure as input and verifies a subset of these rules. The tool can be extended with additional rules in order to reduce the effort for certification tests. In addition to this tool, we introduce software-defined network-based tests to automatically verify compliance with the rules by checking the firewall constraints and host connections. In particular, we focus on a security certification standard named Payment Card Industry Data Security Standard. This certification aims to reduce the risk of data breaches in cardholder data by ensuring industry standard practices for payment card transactions. Our tool offers effort reduction in auditing through automation. It supports continuous auditing and network security enhancement processes.**

***Keywords*—Test automation, network topology, certification tests, banking domain, industrial case study, software-defined network**

I. Introduction

In industries such as banking and finance, quality management, security management, and production management, companies of all sizes may face security risks, including data breaches and malicious attacks, as well as insider threats and phishing attacks [1], [22], [29]. In addition, cybercriminals can exploit vulnerabilities in a company's network and systems to gain access to confidential data. Insiders can gain access to company resources for their own purposes. A company may also face distributed denial of service (DDoS) attacks [8], [9], [17] that can disrupt operations, code injection attempts that may allow malicious code to be executed remotely, and cross-site scripting attacks that may spread malware and other harmful content. Therefore, companies should have a comprehensive security plan in place to avoid or mitigate these threats, which includes regular updates and input sanitation [4] for instance.

By obtaining certification, businesses can protect their assets and customers from security threats [2]. The certification process may identify weak points in the security system, such as outdated software and hardware, and provide guidance on how to protect assets like customer data. Certification can help companies understand their security posture and provide a roadmap for improvement. Certification can also assist businesses in ensuring compliance with industry regulations and laws. This will reduce their risk of cyber-attacks and data breaches, and ensure the safety and security of their customers' data. [19], [26]

Certification tests are applied in various domains, particularly for safety-critical software systems since these systems are subject to catastrophic failures. Standards like DO-178B [11] which is adopted for testing avionics software, govern the regulations for these tests.

Similar regulations also take place in other application domains like consumer electronics [6]. This paper focuses on the banking domain where software systems are not safety-critical. However, they are business-critical applications that provide financial services. Hence, they are also subject to rigorous certification tests for conforming to various standards.

These tests are performed manually and they take weeks to complete. As a result, they are time- and money-consuming. One of the regulations applied in the banking domain is framed by the Payment Card Industry Data Security Standard (PCI DSS) [23] that must be followed by companies interacting with credit card information. Although not a legal requirement, PCI DSS is an industry standard that is enforced by major credit card companies. It provides guidance for securing both the credit card data and the personal information of the cardholder against threats [14], [27]. Securing this data is critical for avoiding financial losses that might be suffered by the customer. The data breach may also result in fines and legal fees that must be paid by the merchant. Compliance with PCI

DSS may also help companies in building trust and reputation. Hence, practicing its guidelines is beneficial for any company that stores or processes cardholder data.

The PCI DSS Self-Assessment Questionnaire (SAQ) [23] is a document for merchants to self-evaluate and prove their PCI DSS compliance. This questionnaire must be submitted by merchants annually in order to achieve PCI DSS compliance. The questionnaire guides merchants in identifying potential vulnerabilities that threaten the security of cardholder data and devising solutions to these issues. The approval of the PCI DSS SAQ form is usually enough to comply with PCI DSS, except for certain merchants with higher levels of risk or complexity. The procedure for complying with the self-assessment questionnaire requires a significant amount of time and effort by the companies [7], [18]. It is a complex, demanding, and time-consuming task to understand the guidelines, conduct relevant testing activities, and reinforce the company policies and procedures according to the SAQ. PCI DSS has 12 main requirements as follows [23].

1) Apply necessary implementations and maintenance to the firewall in order to secure cardholder data.
2) Avoid using vendors' default security settings and passwords for system access and other security parameters.
3) Ensure the security of any stored cardholder information.
4) Employ encryption protocols when transmitting cardholder data across public networks.
5) Employ anti-virus software or programs and use up-to-date versions.
6) Implement and manage secure applications and systems.
7) Restrict access to cardholder data to only those with a legitimate business need.
8) Give unique identifiers to each personnel with access to the computers.
9) Limit physical access to areas where cardholder data is stored.
10) Keep a record of all network assets and cardholder information accesses.
11) Perform tests on security systems and processes on a regular basis.
12) Establish and maintain a comprehensive information security policy that applies to all personnel.

In this paper, we aim to automate network topology examination and configuration testing in PCI DSS requirement (1) stated above. Using automated testing we aim to reduce the time required for manual testing for this requirement. The first requirement of PCI DSS is to install and maintain Network Security Controls (NSCs). In PCI DSS SAQ, NSCs are defined as network security technologies such as firewalls which typically control network traffic between two or more network segments. The compliance of NSCs with industrial configuration standards is a prerequisite, as emphasized in the PCI DSS SAQ form. To ensure adherence, thorough examination of NSC diagrams in alignment with the specified standards and rules is essential. These standards, provided by the National Institute of Standards and Technology (NIST), the Center for Internet Security (CIS), and the International Organization for Standardization (ISO), are considered the reference points for NSCs. These configuration standards include suggestions of best practices for NSCs in order to eliminate possible security vulnerabilities. In summary, this paper proposes a faster solution for testing network topology configurations and enforces industry-standard network security rules for NSCs automatically.

We use Mininet which is a network emulation tool that facilitates the creation of a virtual network consisting of hosts, switches, controllers, and links. The hosts in Mininet operate on standard Linux network software, while the switches offer support for OpenFlow, enabling customizable routing and Software-Defined Networking. This versatile platform serves various purposes such as research, development, learning, prototyping, testing, and debugging, providing a comprehensive experimental network environment that can be conveniently deployed on a laptop or other personal computer.

Our contributions are as follows:

1) Development of a tool with a graphical user interface to construct network topologies for analyzing the constructed models and checking for PCI DSS examination rules automatically.
2) Implementation of automated checks of a predefined list of rules for testing the network security configurations according to industry standards.
3) Implementation of automated tests with Mininet for the mentioned predefined list of rules.

The paper is organized as follows: Section II presents the related work. Sections III and IV presents an overview of the proposed model and the industrial case study. Section V, presents the results and discussion. Finally, Section VI presents the conclusions and future work.

II. Related Work

In the following, we summarize related studies that focus on PCI DSS testing in particular.

A testbed called BuggyCart is used for evaluating PCI DSS scanners [24]. In their study, the researchers discovered that all six scanners they examined did not completely adhere to the guidelines. Additionally, they developed a tool called PciCheckerLite to assess 1,203 websites involved in processing cardholder data. The findings revealed that 86% of the examined websites exhibited at least one PCI DSS violation that should have made them non-compliant.

However, we should note that automated topological evaluation testing for the PCI DSS requirement (1) is not present in PciCheckerLite.

Cardpliance [20] employs static code analysis on Android applications to extract the relevant user interface elements and construct a data flow graph for analyzing the use of sensitive cardholder data. The tool checks for PCI DSS compliance by employing graph traversal algorithms on this data flow graph. For instance, it checks whether the application stores a credit card number without obfuscating it. It is reported that Cardpliance detected 20 violations by analyzing 358 applications. This tool focuses on mobile applications rather than the banking infrastructure.

Another tool [12] aims at keeping track of the flow of credit card data across multiple machines in real-time. It analyzes virtualized systems that run on private cloud servers and it employs virtual machine introspection technology. It also checks for PCI DSS compliance but it does not consider the topology configuration.

To the best of our knowledge, this paper presents the first development and implementation of a graphical user interface tool that automates the testing for PCI DSS network topology configurations, enabling the construction, analysis, evaluation, and automated testing to ensure compliance with examination rules and industry security standards.

III. A Tool for Defining and Analyzing Banking Network Topologies

We introduce a prototype tool to formally define and check a banking network topology. Components of this topology include Automated Teller Machine (ATM), Internet Service Provider (ISP), Router, Switch, Firewall, Database, Backup Database, Server, Demilitarized Zone (DMZ), Cardholder Database Environment (CDE), Cardholder Database, and e-Banking. Our tool provides a graphical user interface to users for adding any number of these components to topology and defining interconnections between them. The tool keeps a graph model of the constructed network in the background. It can also take a text-based formal specification as input and visualize the topology as depicted at the bottom part of Figure 1. The tool explores possible paths in the graph model to check the topology according to the set of predefined rules [13], [25] that are listed in the following.

- Rule 1: The path should start with entrance nodes and end with endpoint nodes.
- Rule 2: There must be a firewall between entrance nodes and endpoint nodes.
- Rule 3: There must be a firewall in the path between the switch and the internet
- Rule 4: The cardholder database can only have a path between the CDE.
- Rule 5: There must be a firewall in the path between the CDE and the internet.
- Rule 6: There must be a firewall in the path between the CDE and the DMZ
- Rule 7: There must be a firewall in the path between the DMZ and the internet.

The tool checks possible paths in the constructed topology to detect and report violations of these rules. We discuss an application of our tool in Section V. In the following section, we describe our approach to checking a software-defined network (SDN) [21] topology.

IV. Testing the network topologies over SDN

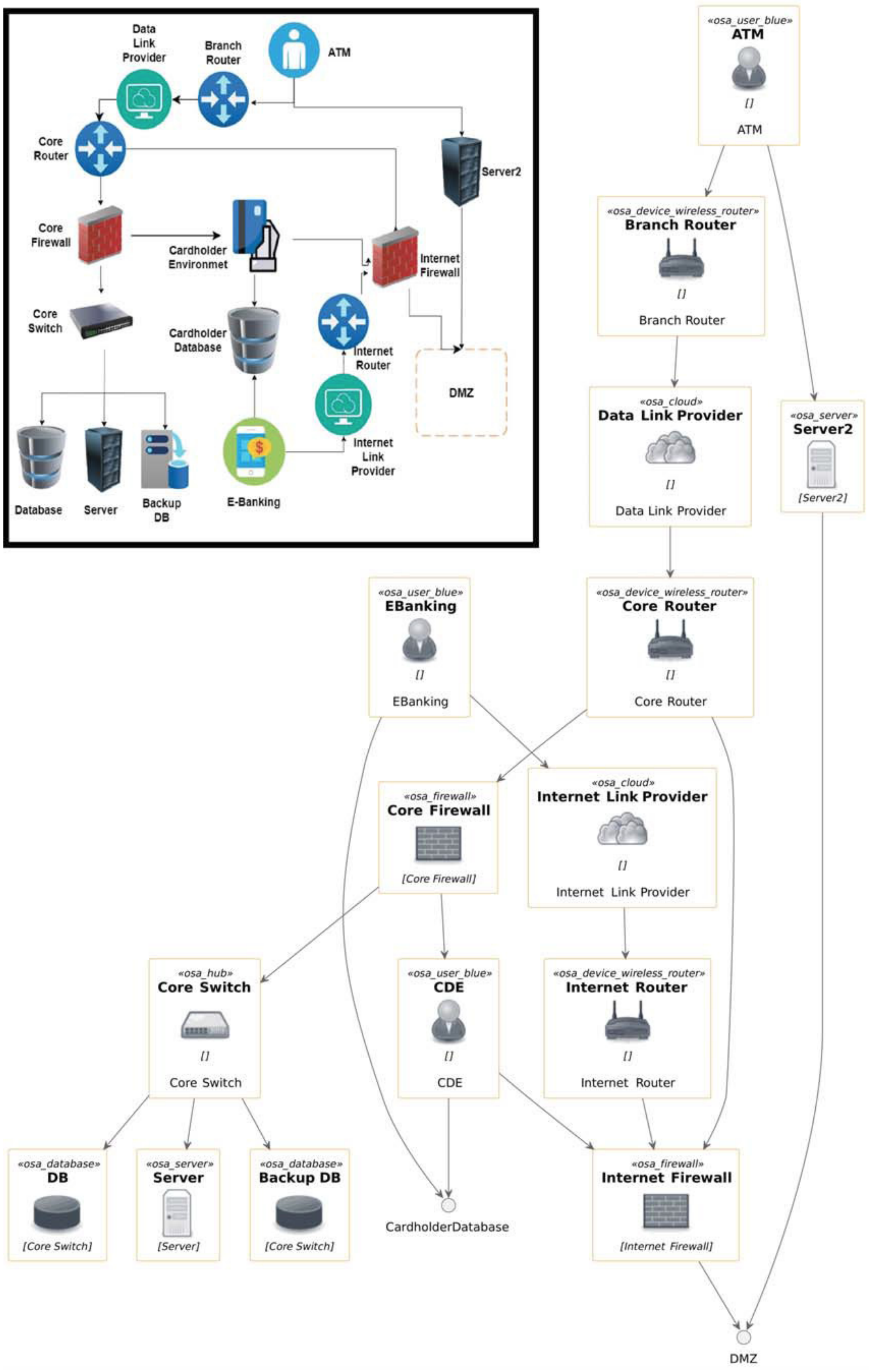
SDN is a recently emerged networking framework that has the potential to modify the limits of present network infrastructures. By isolating the network's control logic from the its routers and switches that handle traffic forwarding, it departs from the traditional vertical integration. Additionally, the distinct data and control planes reduces the complexity of policy enforcement, network configuration, and evolution since the control mechanism is implemented in a centralized controller (or network operating system), and network switches are reduced to simple forwarding devices [16]. While traditional networking architecture is hardware-based, the SDN is software based [3]. This approach provides many benefits, including ease of segmentation, cheaper hardware, a smaller physical footprint, and scalability.

For utilizing SDN in testing the network topology requirements for PCI DSS, we used Mininet[1], which is used for quick network prototyping on a single computer. It uses simple virtualization techniques, like processes and network namespaces, to build scalable SDNs. These capabilities enable the Mininet to swiftly develop, interact with, alter, and share prototypes [5]. Users can build virtual networks, operate network applications and/or protocols, and produce fast software using Mininet. Mininet's main use case is to enable the development of SDN.

Mininet makes simulating the physical network topology rules and architecture possible. With the emerging cloud technologies, SDN became a widely adopted technology [28]. For simulating and testing both the physical and the software-defined networks, using Mininet is a much cheaper, faster, alternative with a low physical footprint [15], [10]. That is why we can easily test the network topology requirements for PCI DSS over Mininet. Using Mininet, we can easily and automatically test large network topologies for the rules that we defined in the previous section from the PCI DSS requirements. Out of the 7 rules that we have defined we can automatically test 6 rules (rules 2 to 7). The first rule that we defined is purely topological and requires no further testing over the SDN.

For the firewall-based rules, first, we implement the given network topology using switches and hosts and then group and label these objects as DMZ, CDE, cardholder database, and internet according to the given network topology. Subsequently, we add the firewall constraints for this given network and apply our firewall test automatically for each labeled group of network hosts and switches. The function takes two hosts from the same group if available, and another host from the other group. For example, rule 6 states "There must be a firewall in the path between the CDE and the DMZ", so we take 2 hosts from the CDE labeled group and 1 from the DMZ labeled group. Then we conduct ping tests to see if the firewall denies the pings across the groups and also verify that the pinging does not result in packet drops inside the CDE. If both conditions are met, the firewall exists between the given two groups. In 1. the pseudocode for the firewall rules is summarized. We are testing the communication between hosts because there is no object that is defined as firewall in Mininet

[1]http://mininet.org/



Data Link Provider
Branch Router
ATM
Core Router
Server2
Core Firewall
Cardholder Environmet
Internet Firewall
Core Switch
Cardholder Database
Internet Router
DMZ
Internet Link Provider
Database
Server
Backup DB
E-Banking
«osa_user_blue» ATM [] ATM
«osa_device_wireless_router» Branch Router [] Branch Router
«osa_cloud» Data Link Provider [] Data Link Provider
«osa_server» Server2 [Server2]
«osa_user_blue» EBanking [] EBanking
«osa_device_wireless_router» Core Router [] Core Router
«osa_firewall» Core Firewall [Core Firewall]
«osa_cloud» Internet Link Provider [] Internet Link Provider
«osa_hub» Core Switch [] Core Switch
«osa_user_blue» CDE [] CDE
«osa_device_wireless_router» Internet Router [] Internet Router
«osa_database» DB [Core Switch]
«osa_server» Server [Server]
«osa_database» Backup DB [Core Switch]
CardholderDatabase
«osa_firewall» Internet Firewall [Internet Firewall]
DMZ

Fig. 1. A sample output of the tool for the given network topology model.

and in other SDN software. The firewall rules are managed by the SDN controller.

Algorithm 1 Checking Firewall Between Groups

```
1: procedure TEST FIREWALL(
           $host, allowedConn, deniedConn$)
2:     $outputAllowed = host.ping(allowedConn)$
3:     $outputDenied = host.ping(deniedConn)$
4:     if $outputAllowed$ & $outputDenied$ then
5:         return True
6:     else
7:         return False
```

For the fourth rule "The cardholder database can only have a path between the CDE." we check the paths of the cardholder database to the CDE. For this test, we iterate over all hosts and ping them and let all other hosts ping the cardholder database to check if there is any path between the database and the entities other than the CDE. If we obtain anything other than 100% packet drop rate between other hosts than the CDE entities, the test fails. Otherwise, the test is passed. In Algorithm 2 the pseudocode for the fourth rule is summarized.

Algorithm 2 Check Links With Database

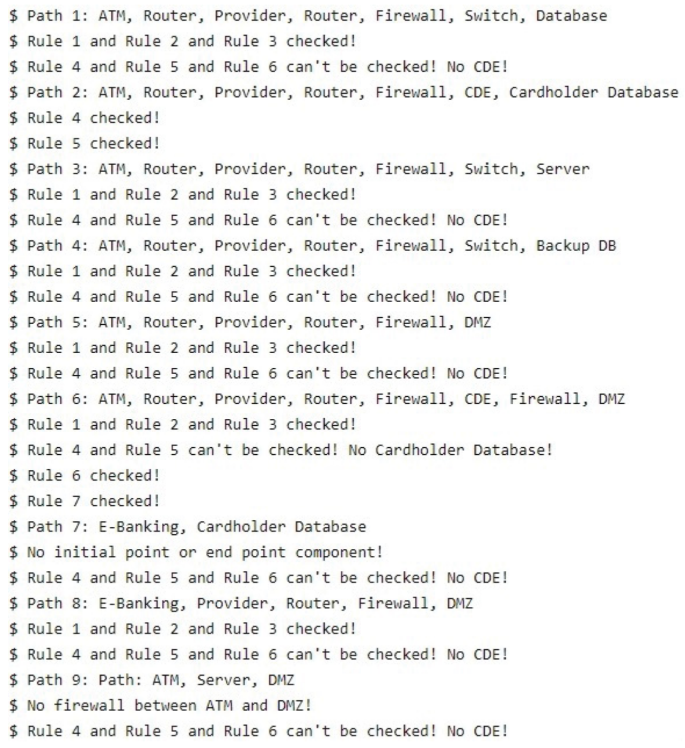
```
1: procedure CHECK LINKS($db, CDE, net$)
2:     for host in net.hosts do
3:         if host != db and host != CDE then
4:             if net.ping(host,db) != 100 or
                 net.ping(db,host) != 100 then
5:                 return False
6:     return True
```

For both of the tests we described, manual testing is exponentially harder to conduct due to the increasing number of network entities and the paths between these entities. With these tests and network simulation implementation capabilities of Mininet, this process is much easier, faster, and requires much less hardware.

V. Results and Discussion

We applied our tool to a case study, where we used a representative network topology model as depicted on top of Figure 1. We defined this topology via the user interface of the tool. The tool visualized the topology as depicted at the bottom part of Figure 1. It took a few seconds for the tool to analyze this topology. A snippet from the output of the tool can be seen in Figure 2.

The tool checks the defined rules provided in Section III, for each path. We see that nine paths are explored in this example. Some of these rules cannot be checked for some paths since a component that is relevant to the rule is not visited along the path such as the absence of CDE in paths 1 and 3. There are also two rule violations reported from the tool. For instance, there exists a path (Path 9) from an ATM to a DMZ without any firewall which violates Rule 2. The tool is open source and it is available in a public repository[2].



```
$ Path 1: ATM, Router, Provider, Router, Firewall, Switch, Database
$ Rule 1 and Rule 2 and Rule 3 checked!
$ Rule 4 and Rule 5 and Rule 6 can't be checked! No CDE!
$ Path 2: ATM, Router, Provider, Router, Firewall, CDE, Cardholder Database
$ Rule 4 checked!
$ Rule 5 checked!
$ Path 3: ATM, Router, Provider, Router, Firewall, Switch, Server
$ Rule 1 and Rule 2 and Rule 3 checked!
$ Rule 4 and Rule 5 and Rule 6 can't be checked! No CDE!
$ Path 4: ATM, Router, Provider, Router, Firewall, Switch, Backup DB
$ Rule 1 and Rule 2 and Rule 3 checked!
$ Rule 4 and Rule 5 and Rule 6 can't be checked! No CDE!
$ Path 5: ATM, Router, Provider, Router, Firewall, DMZ
$ Rule 1 and Rule 2 and Rule 3 checked!
$ Rule 4 and Rule 5 and Rule 6 can't be checked! No CDE!
$ Path 6: ATM, Router, Provider, Router, Firewall, CDE, Firewall, DMZ
$ Rule 1 and Rule 2 and Rule 3 checked!
$ Rule 4 and Rule 5 can't be checked! No Cardholder Database!
$ Rule 6 checked!
$ Rule 7 checked!
$ Path 7: E-Banking, Cardholder Database
$ No initial point or end point component!
$ Rule 4 and Rule 5 and Rule 6 can't be checked! No CDE!
$ Path 8: E-Banking, Provider, Router, Firewall, DMZ
$ Rule 1 and Rule 2 and Rule 3 checked!
$ Rule 4 and Rule 5 and Rule 6 can't be checked! No CDE!
$ Path 9: Path: ATM, Server, DMZ
$ No firewall between ATM and DMZ!
$ Rule 4 and Rule 5 and Rule 6 can't be checked! No CDE!
```

Fig. 2. A snippet from the output of the tool.

VI. Conclusions and Future Work

This paper presents a proposed tool that automates network topology examination and configuration testing for PCI DSS and allows the definition of network topology through a graphical user interface. The proposed tool offers substantial time savings in the verification process, which can often span several hours or even days, depending on the complexity of the network topology. Automating the examination and configuration testing of a predefined set of rules significantly reduces the effort traditionally required, which typically involves personnel manually checking the rules on-site. With its swift rule-checking capabilities, the tool promptly generates results within seconds, streamlining the overall verification process and enhancing operational efficiency. Additionally, network topology configuration tests using Mininet are introduced, tailored to the predefined rule set. The tool aims to facilitate construction, analysis, evaluation, and automated testing, ensuring compliance with examination rules and industry security standards.

In future work, a controlled experiment will be conducted to assess how much time and cost savings the presented tool provides. In addition, we aim to continuously monitor existing network infrastructure to ensure its PCI DSS compliance for audit purposes. We also foresee extending this tool for checking new rules. Initially, our objective is to enhance the

[2]https://github.com/anilelakas/PCIDSSTester

scope of our tool to encompass 12 requirements outlined in the PCI DSS, thereby expanding its capability to address a comprehensive range of security aspects. As a second step, our aim is to establish a mechanism where the set of rules can be defined independently and readily utilized as input for the tool, enabling the flexibility to introduce new rules or modify existing ones without necessitating any modifications to the tool itself.

ACKNOWLEDGMENT

This research is funded by The Scientific and Technological Research Council of Turkey (TUBITAK) under grant number 119C111. The authors express their gratitude to Korhan Kocabıyık and Mustafa Ulus for their valuable managerial assistance. Additionally, the authors extend their thanks to the Fibabanka employees for their valuable contribution in facilitating the case study.

REFERENCES

[1] H.M. Alzoubi, T.M. Ghazal, M.K. Hasan, A. Alketbi, R. Kamran, N.A. Al-Dmour, and S. Islam. Cyber security threats on digital banking. In *Proceedings of the 1st International IEEE Conference on AI in Cybersecurity (ICAIC)*, pages 1–4, 2022.

[2] R. Anderson and S. Fuloria. Certification and evaluation: A security economics perspective. In *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*, pages 1–7, 2009.

[3] B.N. Astuto, M. Mendonca, X.N. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634, 2014.

[4] G. Manogaran B.S. Rawal and A. Peter. *The Basics of Hacking and Penetration Testing*, pages 21–46. 2023.

[5] R.L.S de Oliveira, C.M. Schweitzer, A.A. Shinoda, and L.R. Prete. Using mininet for emulation and prototyping software-defined networks. In *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pages 1–6, 2014.

[6] S. Dirim, O.O. Ozener, and H. Sozer. Prioritization and parallel execution of test cases for certification testing of embedded systems. *Software Quality Journal*, 2022.

[7] L. Elluri, A. Nagar, and K.P. Joshi. An integrated knowledge graph to automate gdpr and pci dss compliance. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1266–1271, 2018.

[8] K.D. Gaylah and R.S. Vaghela. Mitigation and prevention methods for distributed denial-of-service attacks on network servers. In *International Conference on Advancements in Smart Computing and Information Security*, pages 70–82. Springer, 2022.

[9] Q. Gu and P. Liu. Denial of service attacks. *Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications*, 3:454–468, 2007.

[10] M. Hasan, H. Dahshan, E. Abdelwanees, and A. Elmoghazy. Sdn mininet emulator benchmarking and result analysis. In *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 355–360, 2020.

[11] V. Hilderman and L. Buckwalter. *Avionics Certification: A Complete Guide to DO-178 (Software), DO-254 (Hardware)*. Avionics Communications Inc., USA, 1st edition, 2007.

[12] J. Hizver and T. Chiueh. Automated discovery of credit card data flow for PCI DSS compliance. In *Proceedings of the 30th IEEE International Symposium on Reliable Distributed Systems*, pages 51–58, 2011.

[13] ISO. Securing communications between networks using security gateways. ISO ISO/IEC 27033-4:2014, International Organization for Standardization, Geneva, Switzerland, 2014.

[14] N.M. Karie, N.M. Sahri, W. Yang, C. Valli, and V.R. Kebande. A review of security standards and frameworks for iot-based smart environments. *IEEE Access*, 9:121975–121995, 2021.

[15] F. Keti and S. Askar. Emulation of software defined networks using mininet in different simulation environments. In *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*, pages 205–210, 2015.

[16] D. Kreutz, F.M.V Ramos, P. Veríssimo, C.E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[17] N. Kshetri and J. Voas. Banking on availability. *Computer*, 50(1):76–80, 2017.

[18] J. Liu, Y. Xiao, H. Chen, S. Ozdemir, S. Dodle, and V. Singh. A survey of payment card industry data security standard. *IEEE Communications Surveys & Tutorials*, 12(3):287–303, 2010.

[19] W. Liu, X. Wang, and W. Peng. State of the art: Secure mobile payment. *IEEE Access*, 8:13898–13914, 2020.

[20] S.Y. Mahmud, A. Acharya, B. Andow, W. Enck, and B. Reaves. Cardpliance: PCI DSS compliance of android applications. In *Proceedings of the USENIX Security Symposium*, page 1517–1533, 2020.

[21] T.D. Nadeau and K. Gray. *SDN: Software Defined Networks: An authoritative review of network programmability technologies*. "O'Reilly Media, Inc.", Sebastopol, CA, 2013.

[22] B. Panja, D. Fattaleh, M. Mercado, A. Robinson, and P. Meharia. Cybersecurity in banking and financial sector: Security analysis of a mobile banking application. In *Proceedings of the International IEEE Conference on Collaboration Technologies and Systems*, pages 397–403, 2013.

[23] PCI Security Standards Council. PCI DSS document library, 2022.

[24] S. Rahaman, G. Wang, and D. Yao. Security certification in payment card industry: Testbeds, measurements, and recommendations. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, page 481–498, 2019.

[25] K. Scarfone and P. Hoffman. Guidelines on firewalls and firewall policy. Technical Report Special Publication 800-41, U.S. Department of Commerce, Washington, D.C., 2009.

[26] S.U. Syed. Evaluating the effectiveness of cyber security regulations. 2023.

[27] B. Williams and J. Adamson. *PCI Compliance: Understand and Implement Effective PCI Data Security Standard Compliance*. CRC Press, Boca Raton, FL, 5 edition, 2022.

[28] W. Xia, Y. Wen, C.H. Foh, D. Niyato, and H. Xie. A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, 17(1):27–51, 2015.

[29] N. Yildirim and A. Varol. A research on security vulnerabilities in online and mobile banking systems. In *Proceedings of the 7th International Symposium on Digital Forensics and Security*, pages 1–5, 2019.