OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL
UNIVERSITY DUBLIN

BLOCKCHAIN INTEGRATION FOR ENHANCED SECURITY IN SDN : MITIGATING DDOS ATTACKS

Andrew Brady

C20769825

CMPU4019

TU821/4

22/12/2023

Executive Summary

This report into the security challenges within Software-Defined Networking (SDN), specifically focusing on Distributed Denial of Service (DDoS) attacks. SDN, while offering enhanced flexibility, faces vulnerabilities due to its architecture's separation of control and data planes. Various DDoS attack categories within SDN are identified—application layer, control layer, and infrastructure layer—each targeting different parts of the network.

The report's objectives are :

1) Propose algorithms for real-time detection of potential DDoS attacks.
2) Design mitigation strategies against DDoS threats.
3) Strengthen the SDN infrastructure to proactively defend against and recover from DDoS attacks.

Several detection methods are reviewed, including entropy-based approaches, machine learning-based analysis, traffic pattern analysis, and connection rate analysis. Additionally, the report proposes a decentralised reputation-based system using Blockchain technology to mitigate DDoS attacks.

Blockchain's architecture, incorporating database, block, hash, miner, transaction, and consensus mechanism, is outlined. Proof of Authority (PoA) consensus algorithm is chosen due to its identity-based validator selection. The proposed integration involves three key components: SDN controller, data plane, and PoA Blockchain layer. This integration ensures authentication, validation, and secure execution of control plane instructions, enhancing security against DDoS attacks.

An implementation example is simulated using Mininet Python API, illustrating PoA consensus among validators and message exchange within the network. However, an attempted DDoS simulation code is unsuccessful due to implementation issues.

The report underscores the criticality of blockchain integration in SDN to strengthen control plane security, highlighting its transparent and secure environment. This integration not only strengthens resilience against DDoS but also establishes a robust framework for authenticated communications, fortifying the network against security threats.

Contents

Introduction

The emergence of Software-Defined Networking (SDN) has brought with it, improvements in networking, including but not limited to advancements in flexibility and scalability solutions. While SDN-based clouds have many positive aspects, there are several issues to be aware of, such as security, availability, and performance. This report will focus in on the issue of security in SDN. The intrinsic design of the SDN architecture can lead to security challenges. Compared to traditional networks, SDN offers a larger attack surface because it separates the control plane from the data plane. [1] Security analysis has showed that the SDN framework suffers many security threats, including [2] :

1) Unauthorized access, e.g. unauthorized controller access or unauthenticated application access
2) Data leakage, e.g. flow rule discovery (side channel attack on input buffer) forwarding policy discovery (packet processing timing analysis)
3) Data modification, e.g. flow rule modification to modify packets
4) Malicious applications, e.g. fraudulent rule insertion controller hijacking
5) Configuration issues, e.g. lack of TLS (or other authentication techniques) adoption policy enforcement
6) Denial of service, e.g. controller-switch communication flood switch flow table flooding

For this report, Distributed Denial of Service (DDoS) attacks will be discussed.

Distributed Denial of Service (DDoS) Attacks

The DDoS is an attack in which multiple compromised computers are used to flood the victim servers, with a large number of packets and block them, rendering the resources unavailable to authorized users. [3]

DDoS attacks can be divided into two groups according on the protocol level they aim to target [4] :

1) Network/transport-level DDoS flooding attacks: These attacks, which primarily involve TCP, UDP, ICMP, and DNS protocol packets, aim to interfere with the connectivity of authorized users by taking up all available bandwidth on the victim network.
2) Application-level DDoS flooding attacks: The main goal of these attacks is to degrade the services provided to authorized users by depleting server resources. [5]

In SDN, there are three possible DDoS attack categories;

1) Application layer DDoS attacks: DDoS attacks against applications can be initiated in two ways. Attacking some applications is one; attacking northbound API is another.
2) Control layer DDoS attacks: DDoS attacks against the control plane can be initiated by hitting the controller, northbound, southbound, westbound, or eastbound APIs.

When a new flow is given in the flow table, the complete packet or portion of the packet header is sent to the controller to resolve the query. Sending a lot of packets to the controller during a DDoS assault would use up a lot of bandwidth.

3) Infrastructure layer DDoS attacks: Attacks against switches and attacks against southbound APIs are the two ways that data plane DDoS assaults are launched. The data plane may not have enough space to hold flow rules for regular network flows as a result of creating fictitious flow requests, which might result in many pointless flow rules that must be maintained. [6]

Objective

The objective of this report is to;

1) Propose algorithms and protocols for real-time detection of anomalous traffic patterns indicative of potential DDoS attacks within a SDN environment.
2) Design and implement mitigation strategies to neutralise DDoS threats.
3) Strengthening the SDN infrastructure to proactively defend against and recover from DDoS attacks.

Literature Review

Numerous methods have been proposed to detect different types of DDoS attacks in SDN. These include;

Entropy-based approach: Entropy involves measuring the randomness of packets entering the network. Entropy rises when unpredictability rises and falls when unpredictability falls. There are two entropy techniques used to achieve the detection if DDoS attacks in a network, information entropy and log entropy. When an attack happens, it has been found that the information entropy's value will be greatly reduced, but the attack cannot be rapidly identified. In contrast, the log energy entropy can identify attacks quickly, but its entropy value is less obvious than the information entropy's. [7]

Machine learning-based approach: Utilises machine learning techniques to identify DDoS attacks by analysing network data. This method uses the rate of the traffic and its asymmetry to estimate the probability that a given packet is a part of a DDoS attack. [8]

Traffic pattern analysis approach: Involves looking for DDoS attacks by analysing the pattern of network traffic. To identify whether an attack is occurring, this technique searches for irregularities in the traffic pattern, such as an abrupt spike in traffic volume or strange traffic flows. [8]

Connection rate analysis approach: Involves analysing the pace at which connections are being established to the network. This technique scans the connection rate for spikes, which may be signs of a DDoS attack. The network is said to be under assault if the connection rate exceeds the standard level. [8]

Proposal

The proposed implementation for DDoS mitigation within SDN is a decentralised reputation-based system for accessing the network. This can be implemented using Blockchain technology.

Blockchain

Blockchain is a decentralized, distributed ledger of transactions used to maintain an ever-growing collection of information. The majority of users in the blockchain network must approve and register their agreement for a transaction to be stored in the ledger. A bundle of transactions allots a block in the chain of blocks that makes up the ledger. Each block includes a timestamp and a hash function to the one before it in order to connect the others. The data inside the block is verified for integrity and nonrepudiation using the hash function. Every member of the blockchain network maintains a copy of the original ledger, and all users are synced and updated with any new developments to ensure that everyone is informed. [9]

The blockchain is based on the following building blocks: database, block, hash, miner, transaction, and consensus mechanism.

1) Database : The database is one of the key components of the blockchain. The database serves as a record of every transaction made by every user in the blockchain network. The characteristics of this kind of database include immutable data storage, low latency, decentralized control, high throughput, and integrated security. [10]

2) Block: The blockchain's primary storage component is the block. It preserves and includes information about several transactions. By include the prior block's hash in the current block, the blocks are linked together. [10]

3) Hash: The hash function is a complex mathematical problem which the miners must solve in order to find a block. A database may be searched using the concept of a hash function. Since hash functions do not collide, it is quite unlikely to discover two hashes that are exactly the same for two distinct messages. As a result, the blocks are uniquely identifiable by their hash, which fulfils the functions of integrity verification and identification [11]. To connect blocks, each block contains the parent's hash inside its header, starting a chain that extends back to the first block and producing a series of hashes. The hash values are stored in a hash table, an efficient indexing method that improves the efficiency of the search operations. [12]

4) Miner: A miner is a CPU that attempts to solve a computationally intense mathematical problem to discover a new block. Broadcasting new transactions to every user on the blockchain network begins the process of discovering a new block.

Every user adds new transactions to a block and searches for the proof-of-work for the block. The block will be published to every user to confirm it if someone discovers it. Only when all of the transactions therein are legitimate will the block be validated. When the users of the blockchain network begin to generate the next block in the chain using the hash of the accepted block as the previous hash, the block may be regarded as accepted by all of the participating users in the network. [11]

5) Transaction: A blockchain transaction is any minor operation that is recorded in a public record. Only after being confirmed by the majority of users within the blockchain network are these records put into effect, carried out, and kept on the blockchain. While past transactions cannot be modified, they may all be examined at any moment. For miners, the size of the transaction matters since bigger transactions need more block space and energy consumption, whereas smaller transactions are simpler to verify and use less energy. [12] In terms of SDN, the transaction can be thought of as a packet being sent from one node to another.

6) Consensus mechanism: A consensus mechanism is a protocol that brings all nodes of a distributed blockchain network into mutual agreement on a single data set. They serve as the standards for verification by which every blockchain transaction is accepted. This method is particularly helpful in SDN since it makes it simple and affordable to verify the dependability of massive volumes of data. [13]

The workflow of the blockchain can be summarised as follows: Every time a node completes a transaction, a new block is asked for. The miners will work through algorithms to validate blocks. The hash is calculated upon the addition of a new block, and it is subject to change in response to any modifications made inside the Blockchain. Therefore, it is simple to identify and eliminate any transactions that include inaccurate data.

Consensus Algorithm

The use of consensus algorithms in SDN can help to ensure the reliability of the data without having to use a third-party validator. There are multiple consensus algorithms currently in use in Blockchain technologies including; Proof of Work, Proof of Stake, Delegated Proof of Stake…

However, the consensus algorithm that will be chosen for this proposed solution is Proof of Authority (PoA). Because of their reputation or identity stakes in the network, PoA chooses a limited number of authorities to serve as transaction validators. Users must go through a rigorous notarization procedure to compete for validators. During this process, they must present paperwork proving their true identities and connect them to their on-chain identities to build their online reputation. [14] Permissioned networks established between different institutions can benefit from PoA as it offers a fast and safe transaction process.

The group of validators should stay relatively small to ensure manageable security and efficiency of the network.

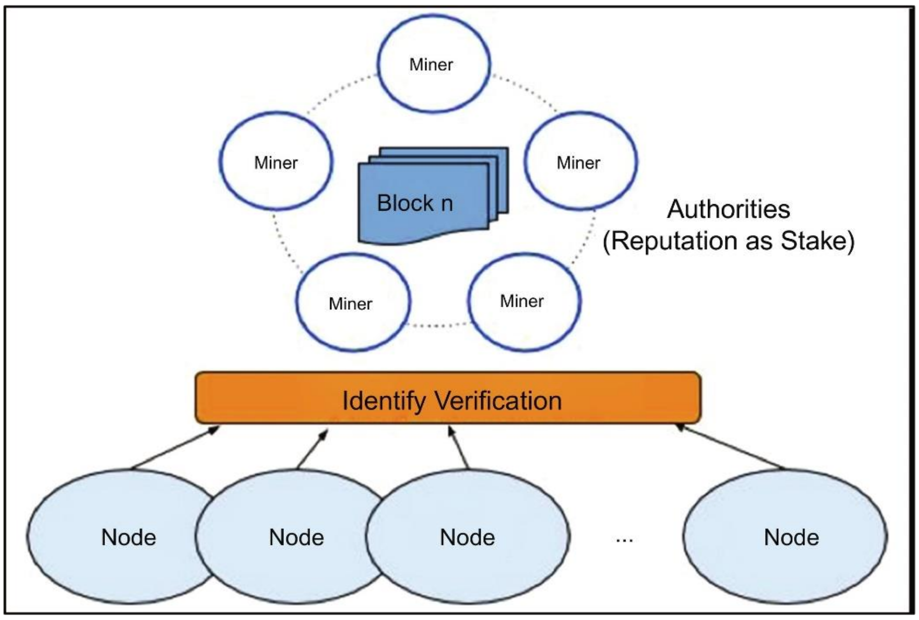


Figure 1 - Proof of Authority Consensus Mechanism Architecture [14]

SDN Network Design

The architecture comprises of three core components:

1) SDN Controller: This centralised entity manages network traffic and policies.
2) Data Plane: Comprising switches, routers, and other devices that forward traffic based on instructions from the SDN controller.
3) PoA Blockchain: Serving as a validation layer, ensuring authentication and integrity of control plane commands.

Integrating a blockchain layer within an SDN architecture serves as a crucial enhancement for security, particularly in mitigating DDoS attacks. This integration involves a meticulous process focused on validation, authentication, and the secure execution of control plane instructions.

The primary function of the blockchain layer lies in its validation and verification capabilities. It acts as a gatekeeper, ensuring that each control plane instruction is legitimate and authenticated before it is executed within the network.

Technically, this integration involves the creation of smart contracts or chaincode specifically designed for the SDN environment. These contracts outline the rules and validation mechanisms governing control plane commands. A PoA consensus mechanism is chosen for its suitability in validating these commands, ensuring that only authenticated entities can propose and verify control instructions.

The heart of the blockchain layer is its ledger. This ledger records validated control plane commands, offering a transparent and tamper-resistant history of all executed instructions.

Any attempt to tamper with or alter these records is immediately detectable due to the blockchain's inherent security features.

The integration reinforces identity management between the SDN controller and network devices. This prevents unauthorized entities from gaining access and executing commands within the network. Additionally, secure communication channels, employing encryption and robust protocols, ensure the confidentiality and integrity of data transmission among blockchain nodes, the SDN controller, and network devices.

Leveraging analytics on the blockchain data provides valuable security insights. Patterns or deviations from normal behaviour can be analysed to further refine and enhance DDoS mitigation strategies.

The integration of the blockchain layer within the SDN architecture creates a tamper-resistant, transparent, and highly secure environment for control plane instructions. This reinforcement not only strengthens the network's resilience against DDoS attacks but also establishes a framework for authenticated communications, fortifying the network infrastructure against various security threats.

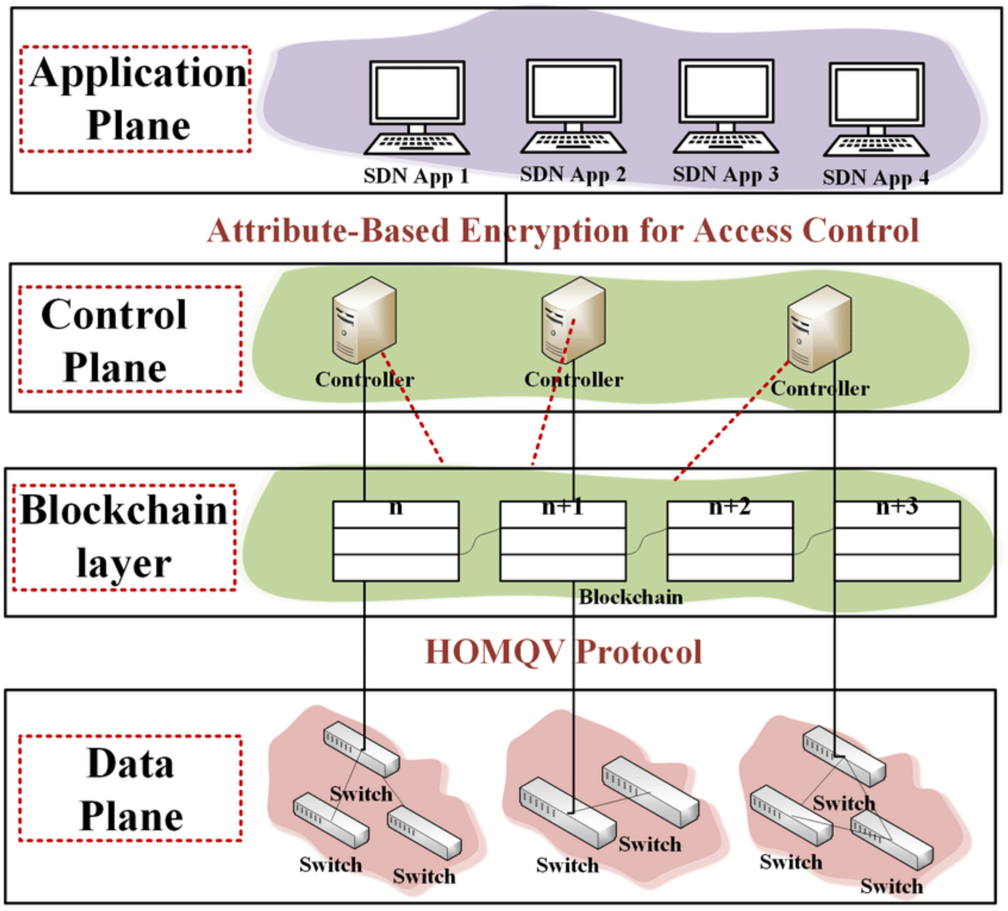


Figure 2 - Blockchain architecture for SDN [15]

Implementation

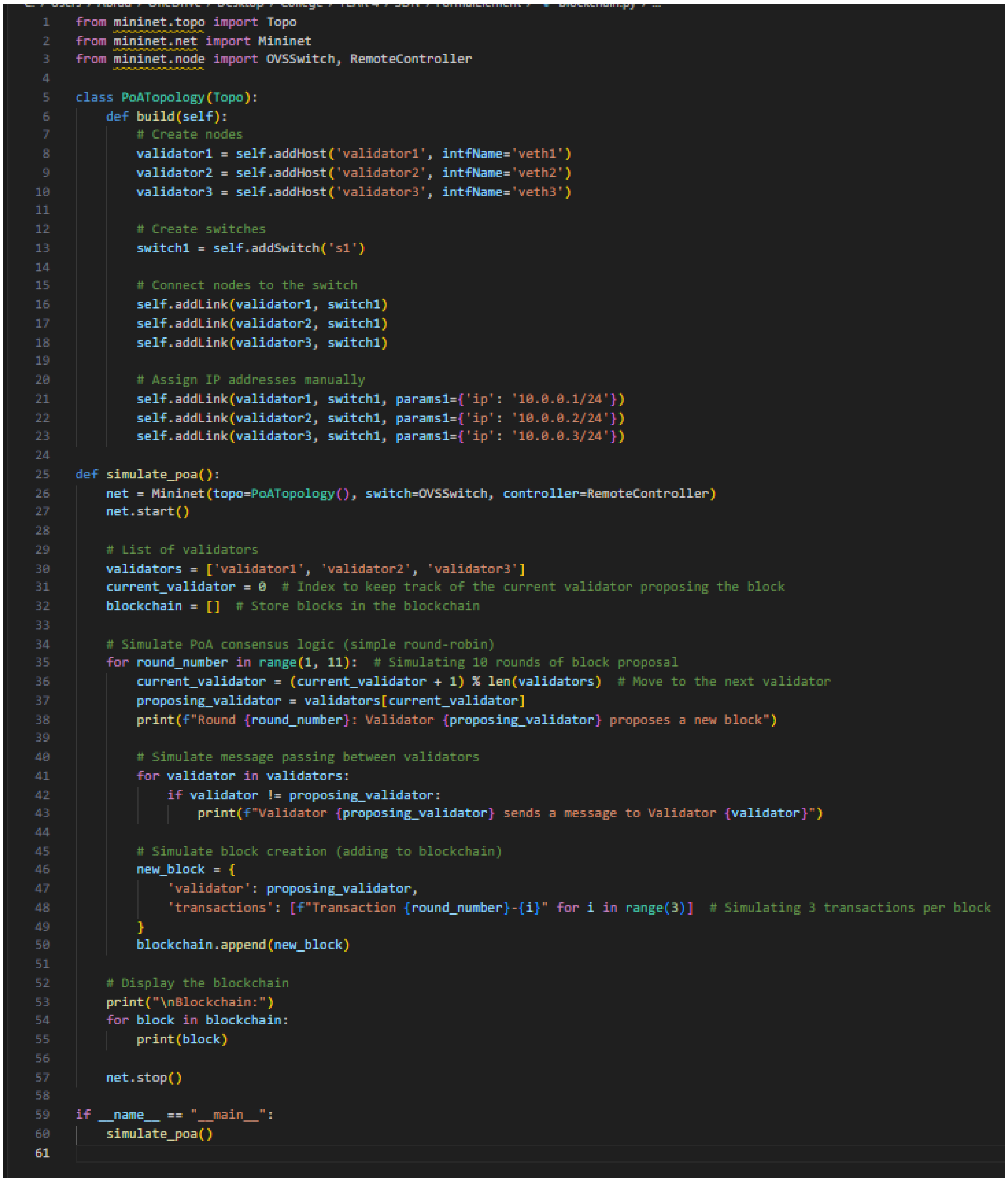
The implementation utilises the Mininet Python API to simulate a network topology for a PoA consensus algorithm. The network topology is established using the ‘PoATopology’ class, which inherits from Mininet’s ‘Topo’ class. Within this topology, three validator nodes (‘validator1’, ‘validator2’, and ‘validator3’) are created and connected to a single switch (‘s1’).

The ‘simulate_poa()’ function orchestrates the PoA consensus simulation. It initializes the Mininet network, defines a list of validator nodes, and implements a round-robin approach for block proposal among these validators.

During each round of the simulation (10 rounds in this case), validators take turns proposing blocks. The current validator proposing the block is tracked using a round-robin mechanism. Validators simulate the creation and proposal of blocks containing a simulated set of transactions.

To illustrate communication between validators, each validator node sends a message to every other validator node in each round of the simulation. These simulated messages signify the exchange of information or transactions between validators within the PoA network.

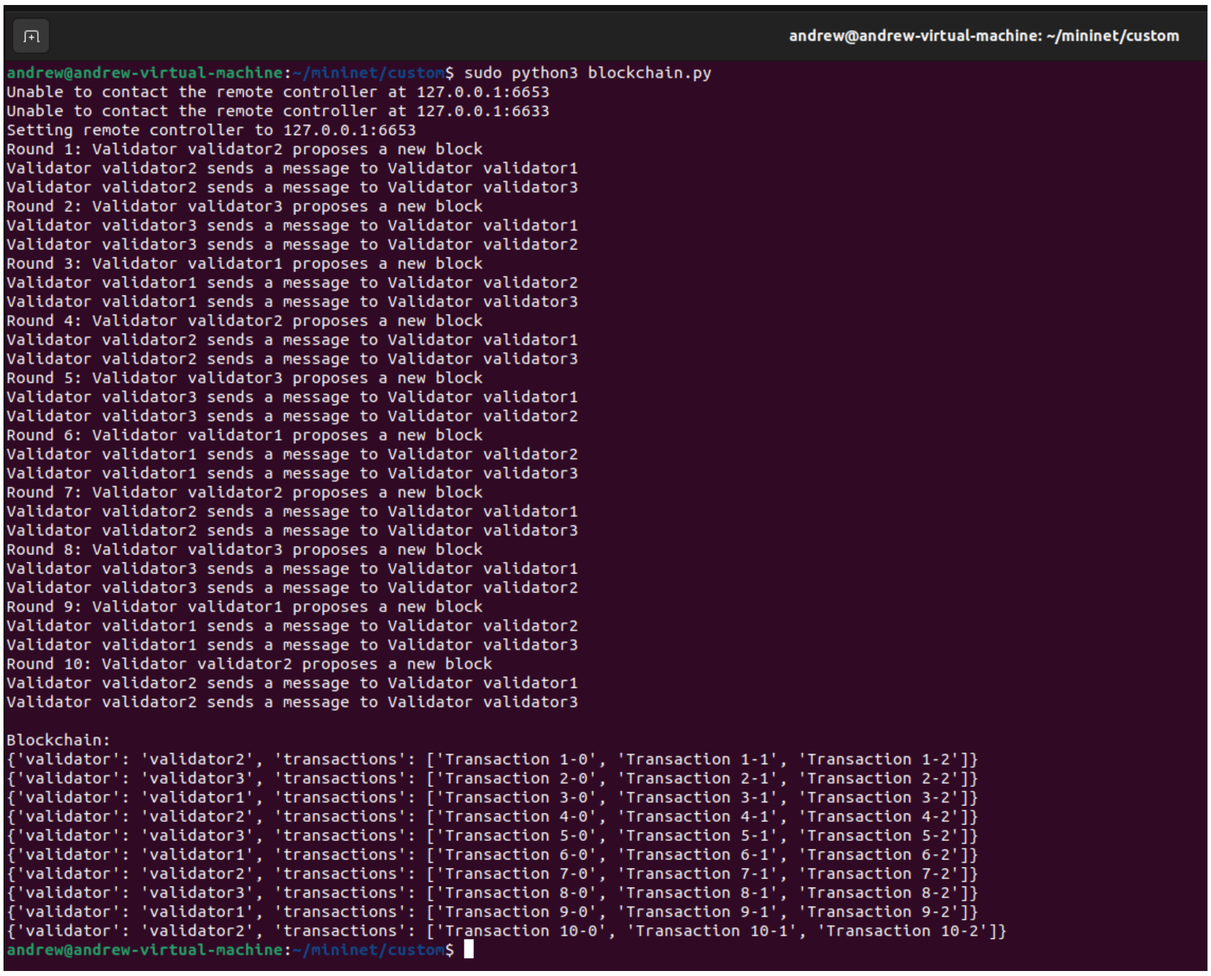
The implementation encountered interface creation conflicts in Mininet, which were resolved by specifying custom interface names for the validator nodes ('eth0'). Additionally, manual assignment of IP addresses to the interfaces of the validator nodes was implemented to mitigate interface creation issues.



```python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import OVSSwitch, RemoteController

class PoATopology(Topo):
    def build(self):
        # Create nodes
        validator1 = self.addHost('validator1', intfName='veth1')
        validator2 = self.addHost('validator2', intfName='veth2')
        validator3 = self.addHost('validator3', intfName='veth3')

        # Create switches
        switch1 = self.addSwitch('s1')

        # Connect nodes to the switch
        self.addLink(validator1, switch1)
        self.addLink(validator2, switch1)
        self.addLink(validator3, switch1)

        # Assign IP addresses manually
        self.addLink(validator1, switch1, params1={'ip': '10.0.0.1/24'})
        self.addLink(validator2, switch1, params1={'ip': '10.0.0.2/24'})
        self.addLink(validator3, switch1, params1={'ip': '10.0.0.3/24'})

def simulate_poa():
    net = Mininet(topo=PoATopology(), switch=OVSSwitch, controller=RemoteController)
    net.start()

    # List of validators
    validators = ['validator1', 'validator2', 'validator3']
    current_validator = 0  # Index to keep track of the current validator proposing the block
    blockchain = []  # Store blocks in the blockchain

    # Simulate PoA consensus logic (simple round-robin)
    for round_number in range(1, 11):  # Simulating 10 rounds of block proposal
        current_validator = (current_validator + 1) % len(validators)  # Move to the next validator
        proposing_validator = validators[current_validator]
        print(f"Round {round_number}: Validator {proposing_validator} proposes a new block")

        # Simulate message passing between validators
        for validator in validators:
            if validator != proposing_validator:
                print(f"Validator {proposing_validator} sends a message to Validator {validator}")

        # Simulate block creation (adding to blockchain)
        new_block = {
            'validator': proposing_validator,
            'transactions': [f"Transaction {round_number}-{i}" for i in range(3)]  # Simulating 3 transactions per block
        }
        blockchain.append(new_block)

    # Display the blockchain
    print("\nBlockchain:")
    for block in blockchain:
        print(block)

    net.stop()

if __name__ == "__main__":
    simulate_poa()

```

Figure 3 - PoA Implementation (Python Code)

The output of the code running shows the validators proposing & validating blocks and validators sending messages to other validators successfully. It also shows the blockchain records for each transaction.



```
andrew@andrew-virtual-machine:~/mininet/custom$ sudo python3 blockchain.py
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
Round 1: Validator validator2 proposes a new block
Validator validator2 sends a message to Validator validator1
Validator validator2 sends a message to Validator validator3
Round 2: Validator validator3 proposes a new block
Validator validator3 sends a message to Validator validator1
Validator validator3 sends a message to Validator validator2
Round 3: Validator validator1 proposes a new block
Validator validator1 sends a message to Validator validator2
Validator validator1 sends a message to Validator validator3
Round 4: Validator validator2 proposes a new block
Validator validator2 sends a message to Validator validator1
Validator validator2 sends a message to Validator validator3
Round 5: Validator validator3 proposes a new block
Validator validator3 sends a message to Validator validator1
Validator validator3 sends a message to Validator validator2
Round 6: Validator validator1 proposes a new block
Validator validator1 sends a message to Validator validator2
Validator validator1 sends a message to Validator validator3
Round 7: Validator validator2 proposes a new block
Validator validator2 sends a message to Validator validator1
Validator validator2 sends a message to Validator validator3
Round 8: Validator validator3 proposes a new block
Validator validator3 sends a message to Validator validator1
Validator validator3 sends a message to Validator validator2
Round 9: Validator validator1 proposes a new block
Validator validator1 sends a message to Validator validator2
Validator validator1 sends a message to Validator validator3
Round 10: Validator validator2 proposes a new block
Validator validator2 sends a message to Validator validator1
Validator validator2 sends a message to Validator validator3

Blockchain:
{'validator': 'validator2', 'transactions': ['Transaction 1-0', 'Transaction 1-1', 'Transaction 1-2']}
{'validator': 'validator3', 'transactions': ['Transaction 2-0', 'Transaction 2-1', 'Transaction 2-2']}
{'validator': 'validator1', 'transactions': ['Transaction 3-0', 'Transaction 3-1', 'Transaction 3-2']}
{'validator': 'validator2', 'transactions': ['Transaction 4-0', 'Transaction 4-1', 'Transaction 4-2']}
{'validator': 'validator3', 'transactions': ['Transaction 5-0', 'Transaction 5-1', 'Transaction 5-2']}
{'validator': 'validator1', 'transactions': ['Transaction 6-0', 'Transaction 6-1', 'Transaction 6-2']}
{'validator': 'validator2', 'transactions': ['Transaction 7-0', 'Transaction 7-1', 'Transaction 7-2']}
{'validator': 'validator3', 'transactions': ['Transaction 8-0', 'Transaction 8-1', 'Transaction 8-2']}
{'validator': 'validator1', 'transactions': ['Transaction 9-0', 'Transaction 9-1', 'Transaction 9-2']}
{'validator': 'validator2', 'transactions': ['Transaction 10-0', 'Transaction 10-1', 'Transaction 10-2']}
andrew@andrew-virtual-machine:~/mininet/custom$
```

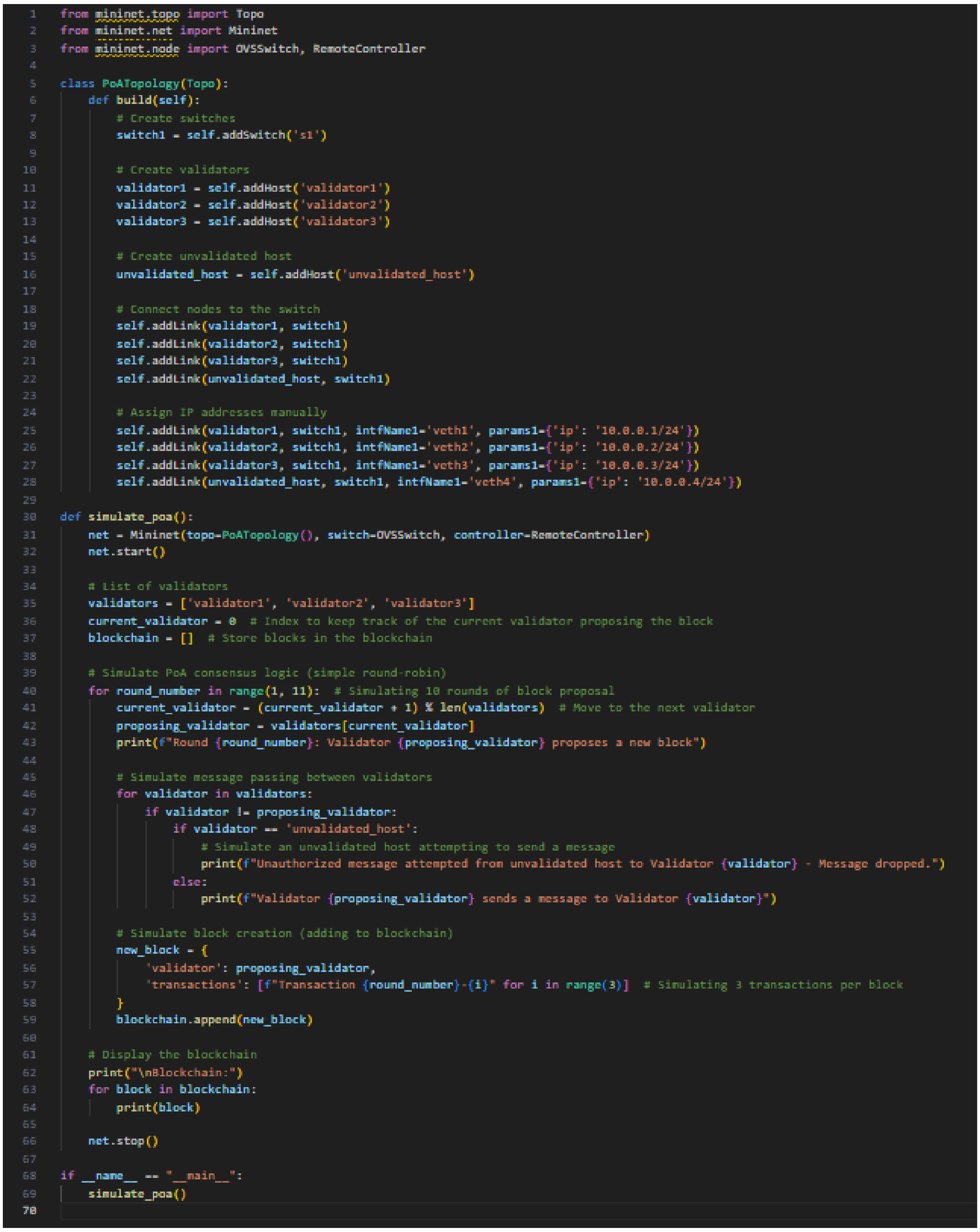
Figure 4 - PoA Terminal Output

Performance Evaluation

To evaluate the performance of the PoA blockchain implementation, a DDoS style attack must be simulated and the protection against this must be validated through the validation system.

The code begins by defining a network topology using Mininet, comprising three validators and an additional unvalidated host. These nodes are interconnected through a switch. The 'simulate_poa' function initializes the network, sets up a list of validators, and starts a loop simulating ten rounds of a Proof of Authority (PoA) consensus mechanism. Within each round, validators take turns proposing blocks, and simulated messages are exchanged between them. If a message attempts to be sent from the unvalidated host, the system logs an unauthorized attempt. For each round, a block is created with the proposing validator and three simulated transactions, adding these blocks to a blockchain representation. Finally, the code displays the resulting blockchain containing all the simulated blocks before terminating the Mininet network simulation. This simulates a basic PoA consensus mechanism among validators while considering potential unauthorized message attempts from an unvalidated source within the network.

This idea is attempted to be implemented in the following code:



```python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import OVSSwitch, RemoteController

class PoATopology(Topo):
    def build(self):
        # Create switches
        switch1 = self.addSwitch('s1')

        # Create validators
        validator1 = self.addHost('validator1')
        validator2 = self.addHost('validator2')
        validator3 = self.addHost('validator3')

        # Create unvalidated host
        unvalidated_host = self.addHost('unvalidated_host')

        # Connect nodes to the switch
        self.addLink(validator1, switch1)
        self.addLink(validator2, switch1)
        self.addLink(validator3, switch1)
        self.addLink(unvalidated_host, switch1)

        # Assign IP addresses manually
        self.addLink(validator1, switch1, intfName1='veth1', params1={'ip': '10.0.0.1/24'})
        self.addLink(validator2, switch1, intfName1='veth2', params1={'ip': '10.0.0.2/24'})
        self.addLink(validator3, switch1, intfName1='veth3', params1={'ip': '10.0.0.3/24'})
        self.addLink(unvalidated_host, switch1, intfName1='veth4', params1={'ip': '10.0.0.4/24'})

def simulate_poa():
    net = Mininet(topo=PoATopology(), switch=OVSSwitch, controller=RemoteController)
    net.start()

    # List of validators
    validators = ['validator1', 'validator2', 'validator3']
    current_validator = 0  # Index to keep track of the current validator proposing the block
    blockchain = []  # Store blocks in the blockchain

    # Simulate PoA consensus logic (simple round-robin)
    for round_number in range(1, 11):  # Simulating 10 rounds of block proposal
        current_validator = (current_validator + 1) % len(validators)  # Move to the next validator
        proposing_validator = validators[current_validator]
        print(f"Round {round_number}: Validator {proposing_validator} proposes a new block")

        # Simulate message passing between validators
        for validator in validators:
            if validator != proposing_validator:
                if validator == 'unvalidated_host':
                    # Simulate an unvalidated host attempting to send a message
                    print(f"Unauthorized message attempted from unvalidated host to Validator {validator} - Message dropped.")
                else:
                    print(f"Validator {proposing_validator} sends a message to Validator {validator}")

        # Simulate block creation (adding to blockchain)
        new_block = {
            'validator': proposing_validator,
            'transactions': [f"Transaction {round_number}-{i}" for i in range(3)]  # Simulating 3 transactions per block
        }
        blockchain.append(new_block)

    # Display the blockchain
    print("\nBlockchain:")
    for block in blockchain:
        print(block)

    net.stop()

if __name__ == "__main__":
    simulate_poa()
```

Figure 5 - DDoS Simulation Code

Unfortunately, the code is implemented incorrectly. Hence, there is no output for the attempted DDoS simulation.

Discussion

The proposed solution integrating Blockchain into Software-Defined Networking (SDN) architecture for mitigating Distributed Denial of Service (DDoS) attacks presents a promising approach but encounters certain limitations in execution and practicality.

The integration of Blockchain in SDN aims to enhance security by establishing a decentralized reputation-based access control system. This concept aligns with the goal of strengthening SDN against DDoS threats, leveraging Blockchain's ledger and validation mechanisms. The solution theoretically ensures authentication and integrity of control commands, thereby reducing the risk of unauthorised access and manipulation within the network.

The practical implementation encountered challenges in simulating a DDoS attack scenario using the Mininet Python API. The code faced issues, leading to unsuccessful execution of the attack simulation. As a result, the effectiveness of the proposed solution in a real-world scenario remains untested due to the hurdles faced during the simulation phase.

While Blockchain integration offers potential security benefits, it introduces complexities in implementation and maintenance. The computational overhead, scalability concerns, and integration challenges between Blockchain and SDN components require careful consideration.

Despite the issues, the proposed solution holds promise in reinforcing SDN against DDoS attacks. Moving forward, more work should be done on refining the integration process, resolving technical challenges in implementation, and validating the solution in realistic DDoS attack scenarios. This includes refining the code for successful simulation, streamlining Blockchain-SDN integration, and addressing scalability concerns for large-scale network deployments.

Conclusion

Integrating Blockchain into SDN holds promise for combating DDoS attacks. However, practical challenges during simulation hindered a comprehensive real-world assessment. The proposed Blockchain-based system aligns with SDN security goals but faces hurdles in implementation, particularly in simulating attacks and integration complexities.

Despite obstacles, this approach offers potential for enhancing SDN security. Future research should concentrate on refining implementation, overcoming technical barriers, and validating the solution in realistic DDoS scenarios to realize its full potential.

References

[1] Y. Jarraya, T. Madi and M. Debbabi, “A Survey and a Layered Taxonomy of Software-Defined Networking,” *IEEE Communications Surveys & Tutorials,* vol. 16, no. 4, pp. 1955-1980, 2014.

[2] S. Scott-Hayward, G. O'Callaghan and S. Sezer, “Sdn Security: A Survey,” in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1-7.

[3] B. Nagpal, P. Sharma, N. Chauhan and A. Panesar, “DDoS tools: Classification, analysis and comparison,” in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2015, pp. 342-346.

[4] S. T. Zargar, J. Joshi and D. Tipper, “A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks,” *IEEE Communications Surveys & Tutorials,* vol. 15, no. 4, pp. 2046-2069, 2013.

[5] S. Ranjan, R. Swaminathan, M. Uysal and E. W. Knightly, “DDoS-Resilient Scheduling to Counter Application Layer Attacks Under Imperfect Detection,” in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, Barcelona, 2006.

[6] S. Sezer and e. al., “Are We Ready for SDN? Implementation Challenges for Software-Defined Networks,” *IEEE Communications Magazine,* vol. 51, no. 7, pp. 36-43, 2013.

[7] C. Fan, N. M. Kaliyamurthy, S. Chen, H. Jiang, Y. Zhou and C. Campbell, “Detection of DDoS Attacks in Software Defined Networking Using Entropy,” *Applied Sciences,* vol. 12, no. 1, 2022.

[8] S. G. Rawat, M. S. Obaidat, S. Pundir, M. Wazid, A. Kumar Das, D. P. Singh and K.-F. Hsaio, “A Survey of DDoS Attacks Detection Schemes in SDN Environment,” in *International Conference on Computer, Information and Telecommunications Systems (CITS)*, Genoa, 2023.

[9] H. F. Atlam, A. Alenezi, M. O. Alassafi and G. Wills, “Blockchain with Internet of Things: Benefits, Challenges and Future Directions,” *International Journal of Intelligent Systems and Applications,* vol. 41, no. 10, pp. 40-48, 2018.

[10] H. F. Atlam and G. B. Wills, “Chapter One - Technical aspects of blockchain and IoT,” in *Advances in Computers : Vol.115*, Academic Press Inc., 2019, pp. 1-39.

[11] A. Narayanan, J. Bonneau, E. Felten, A. Miller and S. Goldfeder, Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton: Princeton University Press, 2016.

[12] G. Peters and E. Panayi, “Understanding Modern Banking Ledgers Through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money,” in *Banking Beyond Banks and Money*, Springer, 2016, pp. 239-278.

[13] H. N. Nguyen, H. A. Tran, S. Fowler and S. Souihi, “A survey of Blockchain technologies applied to software-defined networking: Research challenges and solutions,” *IET Wireless Sensor Systems,* vol. 11, no. 1, pp. 233-247, 2021.

[14] P. Zhang, D. C. Schmidt, J. White and A. Dubey, “Chapter Seven - Consensus mechanisms and information security technologies,” in *Advances in Computers : Vol.115*, Academic Press Inc., 2019, pp. 181-209.

[15] W. Jiasi, W. Jian, L. Jia-Nan and Y. Zhang, “Secure Software-Defined Networking Based on Blockchain,” 2019.