ChainSecure - A Scalable and Proactive Solution for protecting Blockchain applications using SDN

Zakaria Abou El Houda [1,2], Lyes Khoukhi [1] and Abdelhakim Hafid [2]

[1] ICD/ERA, University of Technology of Troyes, France
[2] Department of Computer Science and Operational Research, University of Montreal, Canada
{zakaria.abou_el_houda, lyes.khoukhi}@utt.fr
{zakaria.abou.el.houda, ahafid}@iro.umontreal.ca

Abstract*—Nowadays, blockchain is seen as one of the main technological innovations. Many applications can rely on the blockchain to secure their exchanges. However, applications with private interest cannot rely on *public blockchains*. First, in a *public blockchain*, anyone can read the whole data of the blockchain. Second, anyone can participate to the "consensus process"; the process for determining the validity of each transaction. *Consortium* and *fully private blockchains* aim to combine forcefulness of blockchains with controlled consensus process and stricter permissions for deploying a node and joining the blockchain network. In both *consortium* and *fully private blockchains*, the number of peers on the blockchain network is very small in comparisons with *public blockchain*. Nonetheless, by targeting the nodes of blockchains, an attacker can easily manage the whole blockchain and takes control of the consensus process to validate his illegitimate transactions. In this paper, to defend blockchain nodes from DNS amplification attacks, we propose a scalable and proactive solution in the context of software defined networks (SDN), named ChainSecure. ChainSecure consists of 3 schemes: (1) StateMap, a novel stateful mapping scheme (SMS) to perform a mapping one-to-one between DNS request and response; (2) Entropy calculation scheme (ECS) to measure the disorder / randomness of data using sFlow in order to detect illegitimate flows; (3) DNS DDoS Mitigation (DDM) module to effectively mitigate illegitimate DNS requests. The experimental results show that ChainSecure protects blockchain nodes and can detect/mitigate the attack quickly and achieves high accuracy in detecting illegitimate DNS traffic making it a promising solution to protect blockchain nodes from DNS amplification attacks.

Keywords— DDoS; DNS Amplification; OpenVswitch; SFlow; OpenFlow; Blockchain; Entropy.

I. Introduction

A. *Overview*

Blockchain technologies have caught the attention of many industrials and researchers since the beginning of Bitcoin in 2008 [1] and are in the road to become the fifth disruptive innovation after mainframes, PCs, Internet, and mobile networking. Blockchain is a technology that allows to store and transmit information securely without a central trusted tier; it is a transparent register that everyone can consult but without ever being able to modify these entries. The blockchain consists of an ordered set of blocks; each block contains a set of transactions in the case of Bitcoin and the execution of smart contracts in the case of Ethereum [2]. Each block has a hash value of its predecessor block, as shown in Fig.1. The hash value of the predecessor block forms a link between blocks which makes the blockchain immutable; if an attacker tries to falsify a transaction in n^{th} block, he must change $n-1$ previous blocks, which is tricky in terms of computing capabilities. A complete list of blocks (from the first called genesis to the latest) are stored in each blockchain node and shared between peers relying on the peer-to-peer network. To append new blocks to the blockchain, a *consensus algorithm* is used. Bitcoin network relies on proof-of-work to ensure the *consensus process*, while other blockchains use proof-of-stake or a system of votes to ensure the validity of the pending block.

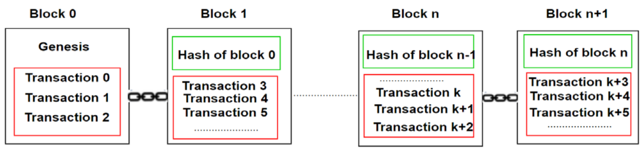


Fig.1. Blockchain.

While the public blockchain network, like Bitcoin makes all blocks and transactions accessible to public and anyone can participate to *consensus process*, *fully private* or *consortium blockchains*, like R3 [3], restrict the access to the blockchain to specific participants. The permission access in a fully private blockchain is managed by a centralized organization, while a consensus process in a consortium blockchain is controlled by a pre-selected set of nodes. Because of the limited number of participants of a fully private and consortium blockchain, it is very important to ensure the availability of all nodes at all time and protect them against DDoS attacks. DNS amplification attack is one of the most devastating types of DDoS attacks aiming to make a targeted blockchain node unable to transmit or receive any information of the blockchain neither participate to the consensus process. In what follows, we describe DNS amplification attacks.

B. *Description of DNS amplification attacks*

DNS amplification attack is one of the most devastating types of Distributed Denial of Service (DDoS) attacks that relies on the use of Open Resolver (publically accessible DNS servers) to flood a victim system with DNS response traffic [4]. In this type of attacks, the attacker spoofs the IP address of DNS requests by replacing the source-address field with the victim's IP. The spoofed queries sent by the attacker are of the type "ANY"; they include all known information about a DNS zone in a single request; then, he manages a botnet of machines (called zombies) to start the attack. Each zombie sends requests to Open Resolver (public DNS server) that makes a recursive resolution and responds to these requests with responses that are sent to the victim. In this case, the victim is flooded with DNS responses that do not correspond to any request he sent (see Fig. 2). According to a recent study, there are about 7.5 million external DNS servers in the Internet; more than

75% of these servers allow recursive name service to the public [5]. This means that if attackers use many recursive servers to generate the attack, this can cause significant collateral damage on the victim.

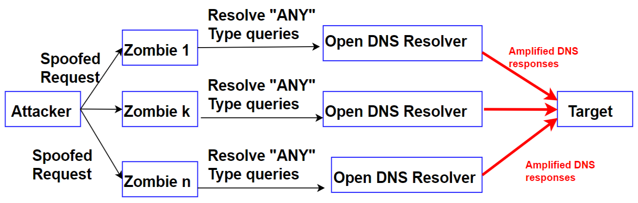


Fig.2. Concept of DNS amplification attacks.

In this paper, to protect the blockchain nodes from DNS amplification attacks, we propose a scalable and proactive solution in the context of software defined networks (SDN), named ChainSecure. To the best of our knowledge, our paper is one of the first works to deal with protecting nodes of private blockchain against DNS amplifications attacks. In the SDN environment, SDN controller has a global view of the network and can be programmed directly by network administrators [6]. While SDN can protect the network from DDoS attacks [7], it can be a victim of these attacks [8]. To address this problem, ChainSecure makes use of a novel stateful mapping scheme implemented in OpenFlow switch (e.g. OpenvSwitch) to defend against DNS amplification attacks and protect SDN controller. Each OpenFlow switch filters DNS packets according to the header fields. By comparing the IP address, MAC address and UDP Port of each request and its corresponding response, SMS checks the legitimacy of the responses and automatically drops illegitimate responses. This allows OpenFlow switches to be smart enough to react very quickly to detect and mitigate attacks, and not wait for a reactive action from the controller. Thus, it can effectively protect the controller resource. However, if a switch processes all DNS traffic that it receives, it will be overwhelmed. To alleviate this issue and protect Ternary Content Addressable Memory (TCAM) of switches which is limited in size, ChainSecure makes use of a robust detection scheme based on traffic flow features using Entropy calculation scheme (ECS). ChainSecure protects the nodes of blockchain without modifying the software of nodes/ blockchain.

Our main contributions can be summarized as follows:

- We propose a novel stateful mapping scheme (SMS) based on in-OpenFlow switch processing capabilities; it allows OpenFlow switches to be smart enough to secure blockchain nodes from DNS amplification attacks. SMS greatly reduces exchanges between OpenFlow switches and OpenFlow controller.
- We propose a real-time detection scheme, called Entropy calculation scheme (ECS), to measure disorder/ randomness of data in order to detect illegitimate DNS requests using sFlow.
- We propose a DNS DDoS Mitigation (DDM) module to effectively mitigate illegitimate DNS requests.
- We evaluate the performance of ChainSecure in terms of scalability, effectiveness and efficiency. The experiments results show that our scheme can effectively mitigate the attack with high accuracy and low overhead.

The rest of the paper is organized as follows. Section II presents related work. Section III introduces design overview and system architecture. Section IV presents StateMap stateful mapping scheme (SMS). Section V presents Entropy calculation scheme (ECS). Section VI describes DNS DDOS mitigation module. Section VII evaluates ChainSecure. Finally, Section VIII concludes the paper and discusses future work.

II. RELATED WORK

Blockchains is considered as a new technology for secure, store and transmit information in a decentralized manner without a trusted tier. Blockchain provides a robust solution to protect all the exchanges made between users against any types of alterations. To the best of our knowledge, there is no work that has been proposed to protect nodes of *private blockchain* against DNS amplifications attacks. In this paper, we do not consider *public blockchain* due to the large number of nodes (e.g., bitcoin blockchain has over 7000 nodes); in case of *consortium and fully private blockchains*, which are becoming more widespread, the security and availability of blockchain nodes need to be considered. In [9], *Mathis et al.* proposed an OpenFlow-based firewall to provide security to the blockchain nodes. The solution proposed in [9] is implemented as a module in a SDN controller and uses SDN functionalities for filtering network traffics; it provides access control functionality and protect blockchain nodes from DoS attacks. However, a very high packet rate from switches to SDN controller may overload the control plane. In [10], *Wang et al.* proposed an entropy-based flow statistics scheme in the OF switch; it focuses on detection but it cannot find the victim or the illegitimate hosts and block them. Moreover, the calculation of the packets IP addresses destination entropy value may delay the response time. *Sun* [11] proposed a low-cost hardware solution to defend against amplification attacks. The solution works well; however, it is hardware-based making it hard to update and extend. *Guo et al. [12]* proposed a mechanism that deploys filters at the border of the network to block incoming source IP addresses not sent from the network. The effectiveness of this method [12] depends on the global deployment across the Internet. It is "neighborhood policy" that requires all Internet service providers (ISP) to participate in order to provide the list of IP addresses that do not belong to their networks. *Kambourakis et al.* [13] proposed a solution for the DNS amplification attack by storing all incoming DNS requests and responses. Each time an illegitimate response is received, a counter is incremented until it reaches a threshold. When the threshold is reached, an attack alert is generated and a DNS amplification attack is assumed to have happened. The problem with this approach is that it does not scale for large networks because it needs to store all DNS traffic queries and responses. *Zaalouk et al.* [14] proposed a solution based on SDN to counter DNS amplification attacks. The solution uses sFlow[15] to monitor DNS traffic. If the detection module detects an attack, it informs the orchestrator. The orchestrator commands the controller to return back traffic to the orchestrator for inspection. This solution does not distinguish between legitimate and illegitimate responses since all DNS responses will be sent to SDN controller and may overload the orchestrator. To address the shortcomings of existing solutions [9-14], we propose an efficient and scalable solution, called ChainSecure, to detect and prevent DNS amplification attacks. In our work, we combine entropy calculation using sFlow with SDN functionalities to block illegitimate traffic. We also use in our process of detection/mitigation, the REST [16] API to manage controller and block illegitimate hosts.

III.SYSTEM DESIGN

A. *Design Overview*

ChainSecure should give a full protection to blockchain nodes from any illegitimate traffic. Unlike existing solutions [9-14] which try to analyze the state of the network and detect the attacks, we aim to act proactively in order to avoid sending illegitimate traffic to blockchain nodes; this is ensured via the proposed StateMap scheme. In addition, to protect Ternary Content Addressable Memory (TCAM) of switches which is limited in size, ChainSecure makes use of a robust detection scheme, based on traffic flow features using Entropy calculation scheme (ECS), to detect illegitimate traffic. Finally, the attacks should be effectively mitigated and the whole system has to be as scalable as possible.

B. *System Architecture*

The architecture of ChainSecure consists of three main schemes (see Fig.3): (1) StateMap, a novel stateful mapping scheme (SMS); (2) Entropy calculation scheme (ECS); and (3) DNS DDoS mitigation module.

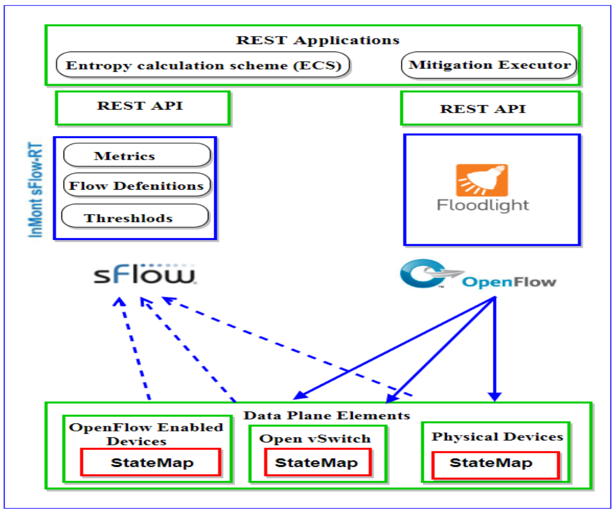


Fig.3. System Architecture

StateMap, a novel stateful mapping scheme (SMS), is based on in-OpenFlow switch processing capabilities. SMS performs one-to-one mapping between DNS request and response; it operates proactively by sending only legitimate responses, excluding the amplified illegitimate traffic, to SDN controller; this allows protecting SDN controller from DNS amplification attacks. Entropy calculation scheme (ECS) aims to measure the disorder/randomness of data using flow statistics; ECS has the objective to detect, in real-time, illegitimate flows based on current network features. ECS is running as an application on the top of the controller and using sFlow protocol; it collects traffic information and detects automatically illegitimate flows. DNS DDoS mitigation (DDM) module aims to effectively mitigate illegitimate DNS requests. OpenFlow was not designed to support QoS features; however, OpenFlow 1.3 introduces *meters* to the OpenFlow protocol (see Section VI). A flow entry can specify a *meter*; *meter* entries with different *Meter_id* are deployed to monitor the speed of DNS requests of the classified illegitimate flows by ECS; if the packet rate exceeds the *band*, DDM drops suspected packets (rate limiter).

IV.STATEFUL MAPPING SCHEME (STATEMAP)

StateMap is a new stateful mapping scheme that performs one-to-one mapping between DNS request and response. The SDN controller pushes the control functionality to OpenFlow switches in order to process all DNS packets. StateMap considers a DNS response as legitimate if there is a pre-sent DNS request matching that response. More specifically, the DNS response must have the same reversed values for the fields MAC, IP, and UDP port of a pre-sent request; otherwise, DNS response will be classified as illegitimate and systematically eliminated. This voids any attempt of external attacks that aims to flood the blockchain nodes with amplified DNS responses. When an attacker is within the network, he can spoof a source IP address of a blockchain node to direct the DNS response to that node/victim. To remedy this problem, the DNS response packet received, by each OpenFlow switch, is transferred to the original port from which the corresponding request came (see Fig.4). If the IP address has been spoofed, the attacker will receive the returned traffic. Otherwise, it will be the legitimate source that receives the legitimate response. Thus, the blockchain is totally protected; thanks to our proposed one-to-one mapping, between request and response based on ingress port of incoming request, we ensure that the blockchain node does not receive any illegitimate traffic.

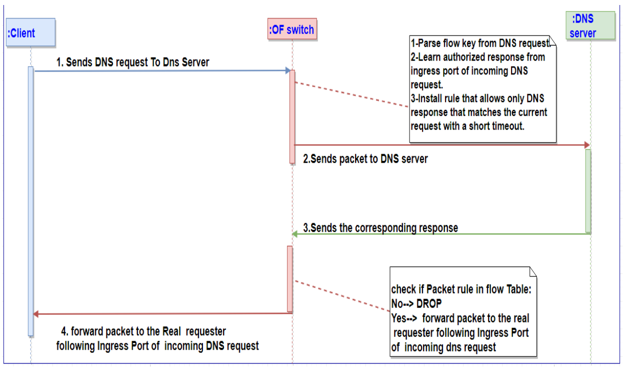

:Client
:OF switch
:DNS server
1. Sends DNS request To Dns Server
1-Parse flow key from DNS request.
2-Learn authorized response from ingress port of incoming DNS request.
3-Install rule that allows only DNS response that matches the current request with a short timeout.
2.Sends packet to DNS server
3.Sends the corresponding response
check if Packet rule in flow Table:
No--> DROP
Yes--> forward packet to the real requester following Ingress Port of incoming dns request
4. forward packet to the Real requester following Ingress Port of incoming DNS request

Fig.4. StateMap functionalities.

V.ENTROPY CALCULATION SCHEME (ECS)

The huge amount of DNS requests may exhaust Ternary Content Addressable Memory (TCAM) of the OF switch. To alleviate this issue, we propose ECS to detect the illegitimate flows and contain them in the real requester's space. First, we describe our information collection method based on flow packet sampling using sFlow; then, we describe ESC, an entropy based anomaly detection scheme to measure the disorder/ randomness of data.

1. *Flow statistics collection*

There are two commonly methods for collecting information: the first is based on OF protocol and the second is based on flow monitoring. In this paper, we choose to implement ECS using flow monitoring methods with sFlow. In the following, we describe each of these methods and we justify our choice.

To detect DNS amplification attacks in SDN, most existing solutions propose to collect and send, periodically, the features of the flows (e.g. number of received packets, duration of matched flows) to SDN controller using OpenFlow (OF) protocol. Features collection with OF protocol can be initiated when the controller sends a feature request (*ofp_flow_stats_request*) to the OF switches which respond by sending the flow table content (*ofp_flow_stats_reply*). This method can collect the overall traffic of flow information passing through the data plane. However, this method can overload the control plane and exhausts the bandwidth between the OF controller and OF switches; furthermore, it may exhaust TCAM in OF switches. Therefore, OF based method is not adequate for detecting high rate DNS amplification attacks.

To address the shortcomings of the method described above, we decided to monitor flows using sFlow. This is more efficient, scalable and does not consume bandwidth between controller and OF switches. sFlow performs flow aggregation that is required during DDoS attacks when the number of flow entries is very high. The sFlow collector (sFlow-RT[17]) receives periodically packet samples from each sFlow agent embedded in data plane (network devices) and updates the counters of each flow during the monitoring interval. Afterwards, periodically, ECS calculates entropy of the current OF switch. In what follows, we describe the details of ECS.

2. *ECS*

The entropy calculation is a concept of information theory [18] which measures the disorder/randomness of incoming data (i.e., the incoming flow for a given time period). ECS runs as an application on the top of the controller and uses sFlow protocol; it collects traffic information and computes the entropy of each OF Edge switch. When the network of blockchain nodes is under attack, the number of packets that have the same $ipsrc$ towards a specific blockchain node will increase causing a concentrated distribution of $ipsrc$. High entropy values mean more dispersed probability distribution of $ipsrc$, while low entropy values mean the concentration of distribution of $ipsrc$. Therefore, we use ECS to measure the changes of traffic information during a monitoring interval ΔT. A flow is characterized by a sequence of packets which have similar properties reaching the same OF Edge switch for a given period of time.

In our work, we define a flow as a seven-tuple: $\{macsrc, macdst, ipsrc, ipdst, srcport, dstport = 53, proto = UDP\}$. We denote an input flow on a local OF Edge switch by $<ipsrc_i, ES_j, t>$, we use I as the set of positive integers, $i, j \in I$ and $t \in R$. $ipsrc$ is the source address of the i^{th} input flow of the j^{th} OF Edge switch ES_j, and t is the current timestamp. We denote by $ES = \{ES_j, j \in I\}$ the set of the OF Edge switches. Thus, an input flow at OF Edge switch can be described as follows:

$$F_{i,j}(ipsrc_i, ES_j) = \{<ipsrc_i, ES_j, t> | ES_j \in ES, i, j \in I, t \in R\}. \quad (1)$$

We set $|F_{i,j}(ipsrc_i, ES_j, t)|$ as the count number of packets of the input flow $F_{i,j}$ at time t. The variation of the number of packets of each flow during the interval ΔT can be expressed as follows:

$$N_{F_{i,j}}(ipsrc_i, ES_j, t + \Delta T) = |F_{i,j}(ipsrc_i, ES_j, t + \Delta T)| - |F_{i,j}(ipsrc_i, ES_j, t)| \quad (2)$$

ECS counts periodically the entropy values of each OF Edge switch separately.

We set the vector $X = \{X_1, X_2, X_3 \dots\dots X_n\}$ as the count number of flows per $ipsrc$ during monitoring interval ΔT reaching each OF Edge switch. X_k represents the number of incoming flow for the k^{th} $ipsrc$. The variation of the number of packets for k^{th} flow during ΔT can be expressed as follows:

$$X_k = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} N_{F_{i,j}}[IP_k = ipsrc] \quad (3)$$

We use the density of each IP address to estimate its probability as follows:

$$P_k = \frac{X_k}{\sum_{k=1}^{N} X_k} \quad (4)$$

P_k gives the probability distribution of $ipsrc$ address IP_k during ΔT. Then, we get the probability distribution of each source $ipsrc$ among the different IP addresses, $P = \{p_1, p_2, p_3, p_4 \dots\dots p_N\}$, and $\sum_{k=1}^{N} P_k = 1$.

We calculate the entropy of j^{th} OF Edge switch ES_j, during ΔT, as follows::

$$H(ES_j) = -\mathbb{E}[\log_2 P_k(ES_j)] = -\sum_{k=1}^{N} P_k \log_2 P_k \quad (5)$$

$$0 \leq H(ES_j) \leq log_2 N \quad (6)$$

In order to normalize the entropy values, we divide the entropy values by the maximum value which is $log_2 N$. Therefore, the normalized entropy values will be in [0, 1] and will be as follows:

$$H'(ES_j) = \frac{H(ES_j)}{log_2 N} \quad (7)$$

We divide the state of the network into normal state and abnormal state. Let $H_{nor}(ES_j)$ and $H_{abn}(ES_j)$ denote the entropy value of OF Edge switch ES_j in the normal state and abnormal state, respectively. $H_{abn}(ES_j)$ decreases when the blockchain nodes network is under attack. On the other hand, $H_{nor}(ES_j)$ is stable during the monitoring interval. Let $\mathbb{E}[ES_j]$ be the mean entropy of $H_{nor}(ES_j)$ for the OF Edge switch ES_j and δ be an adaptive threshold; if the current flow satisfies inequality (8) at least β times in the last μ monitoring intervals (of ΔT), ECS triggers the occurrence of the attack. Afterwards, the flow where distribution of its $ipsrc$ is more concentrated will be classified as illegitimate. Otherwise, it is considered as a normal flow.

$$|H_{abn}(ES_j) - \mathbb{E}[ES_j]| < \delta \quad (8)$$

To make ECS adaptive, we let the mean $\mathbb{E}[ES_j]$ and threshold δ adaptive to change of network traffic; Then, $\mathbb{E}[ES_j]$ and δ will be adapted according to the current normal state of the network and not to a prefixed threshold. We calculate the weighted mean value of $H_{nor}(ES_j)$ as follows:

$$\mathbb{E}[ES_j] = \sum_{i=1}^{n} \alpha_i H_{nor}(ES_j)[i], \ \sum_{i=1}^{n} \alpha_i = 1 \quad (9)$$

Where α_i, $i = 1,2, \dots. n$ represent the weights for the n past flows. The standard deviation of $H_{nor}(S_j)$ can be calculated as follows

$$\sigma=\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(H_{nor}\left(ES_{\mathrm{j}}\right)^{i}-\mathbb{E}\left[ES_{j}\right]\right)^{2}} \qquad (10)$$

Afterwards, we compute the new threshold δ as follows:

$$\delta=\theta.\sigma \qquad (11)$$

Where θ is a multiplicative factor. Each of these parameters ($\mathbb{E}\left[ES_j\right], \sigma$) will be initiated according to the initial normal state of the network. Fig.5 shows the workflow of ECS. First, the proposed ECS application defines address groups, and flows, and initializes the threshold parameters: $\mathbb{E}\left[ES_{\mathrm{j}}\right]$, $\delta, \mathrm{n}, \mu, \beta, \theta$ and the monitoring interval ΔT. Afterwards, during eachΔT, ECS calculates the entropy value and checks whether the inequality (8) is satisfied at least β times in the last μ monitoring intervals (of ΔT), then ECS triggers the occurrence of the attack and DDM (see VI) blocks illegitimate DNS requests. In the following, we describe our mitigation module.

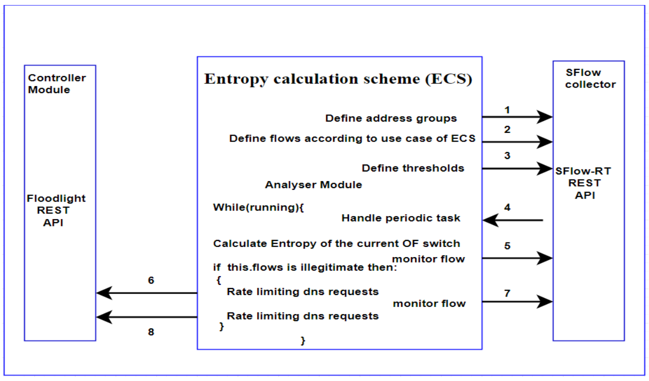

Controller Module
Floodlight REST API
Entropy calculation scheme (ECS)
Define address groups
Define flows according to use case of ECS
Define thresholds
Analyser Module
While(running){
Handle periodic task
Calculate Entropy of the current OF switch
if this.flows is illegitimate then:
{
Rate limiting dns requests
Rate limiting dns requests
}
}
monitor flow
monitor flow
SFlow collector
SFlow-RT REST API
1
2
3
4
5
6
7
8

Fig.5. Workflow of ECS

VI.DNS DDoS MITIGATION(DDM) MODULE

Finally, mitigation action is elaborated to protect blockchain nodes. Once ECS triggers the occurrence of the attack, new flow rules are installed using API of the controller through OpenFlow functionalities into the OF switch under attacks with a high priority to match suspicious packets and monitor their speed. DDM has the purpose to effectively mitigate illegitimate DNS requests. A flow entry can specify a *meter*; *meter entries* with different *Meter_id* are deployed to monitor the speed of DNS requests of the classified illegitimate flows by ECS; if the packet rate surpasses the band, then we drop suspects packets (rate limiter). The table model in OF switches in our study is illustrated in the following figure (Fig.6).

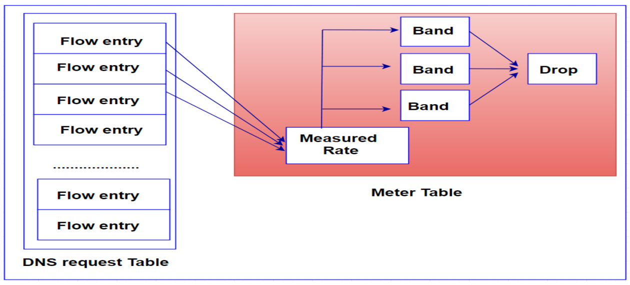


Fig. 6. Table Model in OF switches

VII.EXPERIMENTAL VALIDATION

In this section, we present the experimentation validation of the ChainSecure. First, we introduce the experimental environment. Then, we evaluate the performance of ECS.

A. *Experimental Environment*

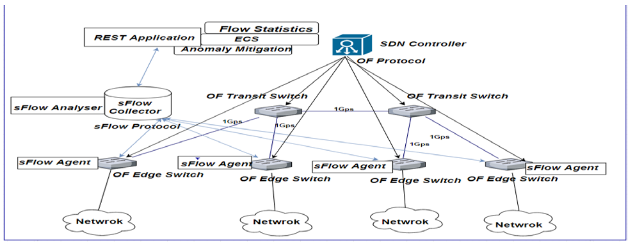
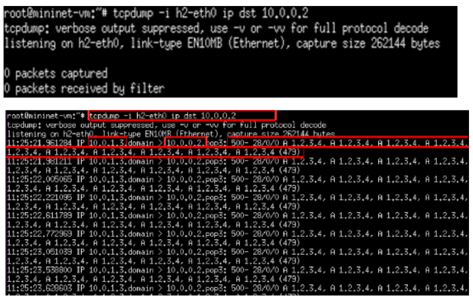


Fig.7. Experimental Environment

Virtual network topology is implemented with Mininet [19] in order to simulate a real network environment. Mininet uses Linux containers and OpenvSwitch to allow realistic virtual networks of hosts and switches to be constructed using a virtual machine. StateMap is implemented in OpenvSwitch. While, ECS is running as an application on the top of the controller and using sFlow protocol ECS collects traffic information and detects automatically illegitimate flows. Then, mitigation action is elaborated to block illegitimate flows. The experimental environment, consists of (1) an OpenFlow controller (i.e., Floodlight[20]) which offers elementary connectivity which can be canceled using the Static Flow Pusher API; (2) an sFlow network monitor (i.e., sFlow-RT), which is a the sFlow collector that can perform monitoring of 7500 switch ports in a data center network; (3) a REST application to perform the anomaly detection scheme(ECS); (4) 6 OF switches, the bandwidth of each link is set to 1 Gbps. Each OpenFlow network contains more than 20 hosts; multiple hosts are simulated to launch the attack and other hosts are legitimate blockchain nodes executing blockchain application and (5) multiple hosts in our topology are simulated to act as Open Resolver and send amplified DNS responses. The rates of the attack are changed from 50 to 500 Mbps in the objective to test the scalability of our proposed solution in large scale, the sampling rate in sFlow is 1/64. For the attack script, Scapy's Python library [21] module of Python is used to forge DNS queries in large amounts. NodeJs [22] is used to create the DNS test server using dnsd package. The server is implemented to send large DNS records as responses to the victim. The experiment results are presented as follows: Without ChainSecure, the blockchain node is flooded by illegitimate DNS responses as illustrated in the bottom of Fig.8. After deploying of ChainSecure, we re-launched the attack. The capturing with Tcpdump in the top of Fig. 8 shows that the blockchain node (victim) does not receives any illegitimate traffic. As shown in Fig. 9, thanks to StateMap, we can also maintain the bandwidth resources of node of the blockchain. Even if the attack rate reaches 2000 packets per second (pps) the bandwidth still keeps almost 12 Mbps; however, without ChainSecure, the bandwidth decreases sharply, which means that the blockchain node is flooded with illegitimate traffic. Fig. 10 illustrates that when the controller is disabled, the traffic attack sustains over 2000 DNS requests per second. However, when the controller is enabled, the traffic of DNS requests is stopped when ECS classifies the flow as illegitimate. ECS instructs the controller which communicates with the switches to mitigate the DDoS traffic, the time taken by the detection and mitigation operations is less than 13 seconds as shown in Fig. 11.To examine our proposed ECS, we simulate our attack within an interval of 250 s. We launch the attack during the interval of 150-200s.We can see in Fig. 12 that the normalized entropy values decrease rapidly.



```
root@mininet-vm:~# tcpdump -i h2-eth0 ip dst 10.0.0.2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes

0 packets captured
0 packets received by filter
```

```
root@mininet-vm:~# tcpdump -i h2-eth0 ip dst 10.0.0.2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
11:25:21.961284 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4 (479)
11:25:21.981211 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4 (479)
11:25:22.005065 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4 (479)
11:25:22.221095 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4 (479)
11:25:22.611789 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4 (479)
11:25:22.772969 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4 (479)
11:25:23.061039 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4 (479)
11:25:23.538800 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4 (479)
11:25:23.628603 IP 10.0.1.3.domain > 10.0.0.2.pop3: 500- 28/0/0 A 1.2.3.4, A 1.2.3.4, A 1.2.3.4, A 1.2.3.4,
```

Fig.8. Capture on node's network with and Witouh ChainSecure.

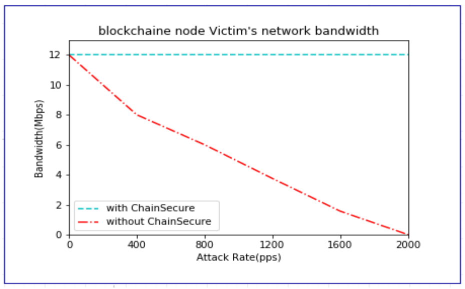


Fig.9.Victim node's netwrok Bandwidth

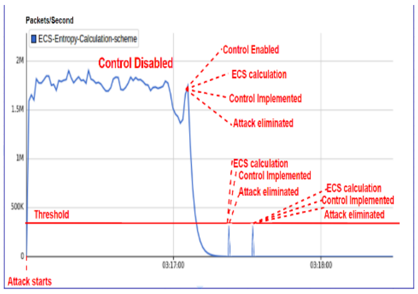


Fig.10. DNS request's flow traffic before and after enabling control

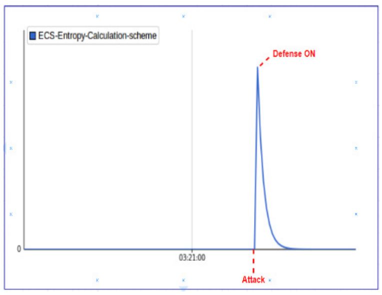


Fig.11. attack mitigation

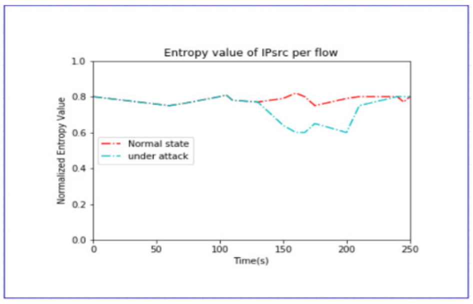


Fig.12.Normalized entropy value of Ipsrc flow

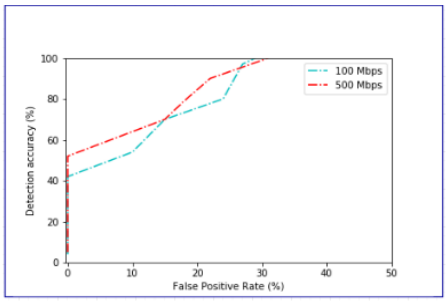


Fig.13. ROC curves for the 100/ 500 Mbps cases .

B. *Performance Evaluaion*

To measure the performance of ECS, we define the Detection Rate (DR) and false positive Rate (FPR) as follows:

$$DR = \frac{TP}{TP + FN}, FPR = \frac{FP}{TN + FP}$$

Where, TP (True Positives) represents the illegitimate flow that are correctly identified as illegitimate, while FN (False Negatives) represents the illegitimate flow that are classified as legitimate. Therefore, DR represents the attack detection rate, FP (False Positives) represents the legitimate flow that are identified as illegitimate, while TN (True Negatives) represents the legitimate flows that are classified as legitimate. The *Receiver Operating Characteristic (ROC)* curves represent the trade-off between DR and FPR. In the experiment we set ΔT as 5s, β is set 2 and μ is 3. As shown in Fig.13, we can see that ECS achieves around 100% detection rate while it has approximately 30% of false positive ratio for both 100 and 500 Mbps. ECS works well in 500 Mbps as we observe a high rate of traffic which leads to a more randomized traffic.

VIII.CONSLUSION

This paper discussed security threats of *Consortium and fully private blockchains*. Because of the small number of peers (nodes) on the blockchain, specific nodes can be targets of DDoS attacks. In order to protect the blockchain nodes from DNS amplification attacks, we proposed a scalable and proactive solution in the context of SDN, named ChainSecure. First, we described a novel stateful mapping scheme that allows OpenFlow switches to be smart enough to secure blockchain nodes from DNS amplification attacks. Then, we proposed a real-time detection scheme called ECS. Finally, a mitigation action is elaborated to block illegitimate flows. For future work, we intend to improve precision of our ECS by using a machine learning methods.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] EthereumFoundation. [Online] Available: https://www.ethereum.org/.

[3] Blockchain for business. [Online] Available: https://www.r3.com/.

[4] Hongyu Gao, Vinod Yegneswaran, Jian Jiang, Yan Chen, Phillip Porras, Shalini Ghosh, and Haixin Duan," Reexamining DNS From a Global Recursive Resolver Perspective", IEEE Transactions on Networking ,Vol. 24, No. 1,February 2016.

[5] "DNS","http://dns.measurement-factory.com/surveys/sum1.html.

[6] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in Proc. IEEE INFOCOM,2013.

[7] Reza Mohammadi, Reza Javidan, and Mauro Conti, "SLICOTS: An SDN-Based Lightweight Countermeasure for TCP SYN Flooding Attacks", IEEE Transactions on Network and Service Management,Vol. 14, No. 2,June 2017.

[8] Yan, Q., Yu, F.R., Gong, Q., and Li, J.,'Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges', IEEE Commun. Surv. Tutor., 18, pp. 602–622, 2016.

[9] Mathis Steichen, Stefan Hommes and Radu State," ChainGuard - A Firewall for Blockchain Applications using SDN with OpenFlow", IPTComm, 2017.

[10] Wang R, Jia Z, Ju L. An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking. IEEE Trustcom/BigDataSE/ISPA. 2015:310-317.

[11] C. Sun, B. Liu, and L. Shi, "Efficient and low-cost hardware defense against DNS amplification attacks," in Global Telecommunications Conference, pp. 1-5. IEEE GLOBECOM ,2008.

[12] F. Guo, J. Chen, and T. Chiueh, "Spoof detection for preventing DoS attacks against DNS servers," in IEEE ICDCS, 2006, pp. 37–37.

[13] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, "A fair solution to DNS amplification attacks",Digital Forensics and Incident Analysis. Second International Workshop on. IEEE, pp. 38-47, 2007.

[14] Zaalouk, A., R. Khondoker, R. Marx and K, Bayarou. "OrchSec: An Orchestrator-Based Architecture For Enhancing Network-Security Using Network Monitoring And SDN Control Functions". Network Operations and Management Symposium, May 5-9, 2014.

[15] P. Phaal ,"https://www.ietf.org/rfc/rfc3176.txt",2001.

[16] "REST API"," http://www.sflow-rt.com/reference.php".

[17] "sFlow-RT", http://www.sflow-rt.com".

[18] H.J. Landau,J.E. Mazo,S. Shamai," Shannon theory: perspective, trends, and applications special issue dedicated to aaron d. wyner", IEEE Transactions on Information Theory,2002.

[19] Mininet. [Online] Available: http://mininet.org.

[20] Floodlight. [Online] Available:http://www.projectfloodlight.org/ .

[21] Scapy,"http://www.secdev.org/projects/scapy ".

[22] NodeJs," "Nodejs", "https://nodejs.org/en//".