A Blockchain-based Architecture for Collaborative DDoS Mitigation with Smart Contracts and SDN

ABSTRACT

The rapid growth in the number of portable and stationary devices makes Distributed Denial-of-Service (DDoS) attacks a top security threat to securely provision services and scale through automation. Existing strategies lack of resources and flexibility to cope with attacks by themselves. Emerging technologies such as blockchain, smart contracts and SDN (Software-Defined Networking) introduce new opportunities for flexible and efficient DDoS defense solutions. In this work, we propose the design of an architecture and smart contract deployed in a public blockchain, which facilitates a collaborative DDoS mitigation architecture across multiple network domains. The advantages of our architecture are the use of an already existing public infrastructure to distribute the rules without the need to build specialized registries or other distribution mechanisms and to enforce rules across multiple domains.

KEYWORDS

Smart Contract, Blockchain, Software-Defined Networking, Security, Distributed Denial-of-Service

1 INTRODUCTION

In the past years a rise in DDoS attacks could be observed [11]. DDoS attacks have the simple goal of interrupting or suspending services available on the Internet [25]. The motivations for such attacks range from personal grudges over blackmail to political reasons [11]. A recent example is the attack conducted against Domain Name System (DNS) servers responsible for domains such as Twitter, PayPal, and Spotify [21] in October 2016. As a consequence those services became unavailable to many US users for several hours. Besides the frequency, also the strength and duration of DDoS attacks is growing making them more effective and dangerous. One reason for the increasing size of attacks is the availability of countless reflectors, *i.e.*, weakly secured or configured IoT devices or home gateways [11, 20, 21].

By exploiting legitimate services on those devices, *e.g.* Simple Service Discovery Protocol, the power of a DDoS attack is amplified and the problem of defense is made more complex. Thus, the impact of DDoS varies from minor inconvenience to serious financial losses for enterprises that rely on their online availability [15]. Various mitigation techniques from both academia and industry have been proposed. However, only a few have been considered for widespread deployment because of their effectiveness and implementation complexities. An ongoing IETF (Internet Engineering Task Force) proposal discusses the development of a collaborative protocol called DOTS to advertise DDoS attacks [14]. However, the infrastructure of blockchains and smart contracts already provide the required instrumentation without tackling with the design and development complexities of such a new protocol. In a different direction, the adoption of DDoS protection services, offered by companies such as Akamai [1] or CloudFlare [3], is increasing [8]. Those cloud based solutions typically can absorb DDoS attacks by increasing capacity and taking the burden of detection away from the device under attack by exporting flow records from edge routers and switches. The further analysis is done in the cloud and packet filtering is used to balance, reroute, or drop the traffic inside the cloud. However, those solutions requires a third party DDoS Protection Service (DPS) provider implying in additional costs and a decrease in service performance.

This paper presents the architecture and design of a DPS mechanism using smart contracts and investigates the possibility of mitigating a DDoS attack in a decentralized manner. The objective is to create an automated, and easy-to-manage DDoS mitigation. Three major building blocks are identified to build such a mechanism:

Blockchains and Smart Contracts. This work presents an architecture and an implementation of an efficient approach to distribute these rules from a victim to ASes based on blockchains and smart contracts. The advantage of using smart contracts in a blockchain is: a) to make use of an already existing infrastructure to distribute the rules without the need to build specialized registries or other distribution mechanisms, b) to apply rules across multiple domains, which means that even if the AS of the victim is not applying these rules, some traffic can still be filtered, and c) the victim or its AS can control which customers get blocked. The only central element remaining is to show proof of IP ownership.

In addition, **SDN** is an effective solution to enable customizable security policies and services in a dynamic fashion. The centralized network control and its deployment based on the OpenFlow [12] protocol facilitates the enforcement of high-level security policies moving away from current approaches based on SNMP (Simple Network Management Protocol) and CLI (Command Line Interface). With SDN, flow-rules can be applied to block DDos attacks, and the closer these rules are applied and those malicious packets can be dropped the less DDoS traffic occurs. Therefore, once attacks reported in the blockchain are retrieved by the SDN controller, the verification and definition of flows to mitigate DDoS attacks is performed in a more agile fashion in the ASes.

The paper is structured as follows. First, in Section 2 introduces collaborative DDoS mitigation strategies and related works. Then, in Section 3 our architecture is presented detailing its components and basic functioning, as well as describing the implementation details. In Section 4 we provide a preliminary evaluation of the development and results obtained so far. The work is concluded highlighting the major contributions and discussing our future work.

2 BACKGROUND AND RELATED WORK

This section presents an overview of the broad categories of DDoS attacks and their mitigation based on cooperative approaches (2.1), and SDN-based works implementing these DDoS strategies (2.2). Then, the concepts on Blockchains and Smart Contracts are described (2.3).

2.1 Cooperative DDoS Mitigation

There are four broad categories of defense against DDoS attacks according to [15]: (1) attack prevention, (2) attack detection, (3) attack source identification, and (4) attack reaction. (1) tries to prevent attacks before they become a problem, *i.e.* as close to the sources as possible. The obvious method to achieve this for amplified or reflected attacks is for the access provider to filter spoofed packets. Those are packets having the address of the victim as the source address in their IP header. (2) can be a difficult task since certain attacks mask themselves as legitimate user traffic or use various traffic types. Due to this complexity it can be hard to make a confident decision if traffic is part of an attack or special user behavior, *e.g.* a flash crowd. (3) is applied after an attack was detected. This step is important to efficiently contain or re-route the attack as close to its source as possible. (4) the final step involves taking concrete measures against the attack. The better the result from (3) the more efficiently this can be done.

Among the reaction techniques there are two main approaches using resource management to react against bandwidth attacks [15]. The first takes effect within the victims domain and the second within the domain of the victims ISP, *i.e.* the AS. Both techniques apply traffic classification and define specific actions for those classes. However, it is rather difficult to give an accurate classification as DDoS attacks can mimic any type of legitimate traffic. In this regard, a number of sophisticated techniques can be implemented to classify traffic. A cooperation of attack reaction strategies implemented both at the AS and the customer can be more efficient than implementing just one.

Other works exist for cooperative defense against DDoS attacks. However, it is still an open issue since DDoS attacks are growing in scale, sophistication, duration and frequency [11]. The IETF is currently proposing a standard protocol [14] covering both intra-organization and inter-organization DDoS situations to detect and mitigate attacks. Through a high-level comparison with the ongoing IETF proposal (the DOTS protocol) [14], instead of making use of an existing infrastructure such as the blockchain and smart contracts, the IETF proposes from scratch the development of such protocol with several requirements (*e.g.*, extensibility, resilience) to be deployed in a distributed architecture. In this sense, the protocol development becomes complex since it must be deployed in distributed and centralized architectures to support different types of communication (inter and intra domain, *i.e*, inside the domain of an AS and between ASes). Instead, we argue that some of the requirements can be inherited from the natural characteristics of blockchains, smart contracts and SDN, avoiding the complexities of development and adoption of new protocols.

In [26] the authors propose a gossip-based communication mechanism to exchange information about attacks between independent detection points to aggregate information about the overall observed attacks. The system is built as a peer-to-peer overlay network to rapidly disseminate attack information to other listening users or systems. A similar approach was presented in [22], formalizing a gossip-based protocol to exchange information in overlay network using intermediate network routers. A different approach is [17], proposing a collaborative framework that allows the customers to request DDoS mitigation from ASes. However, the solution requires an SDN controller implemented at customer side interfaced with the AS, which can change the label of the anomalous traffic and redirect them to security middle-boxes. In the approach presented in this paper customers and ISPs can take action to mitigate an attack by interfacing directly with a blockchain providing the required trust.

2.2 SDN and DDoS Mitigation

The SDN characteristics provide better network visibility by decoupling the control plane from the data plane and control by centralized management to perform tasks such as network diagnosis and troubleshooting [10]. In addition to SDN, the OpenFlow protocol [12] leverages network management by providing a programmable and standardized interface between the data plane and the control plane. It has been recognized that the decoupling of the data plane and the control plane makes SDN a promising solution to enable the enforcement of customizable security services and policies. To deal with DDoS attacks various SDN-based solutions have been proposed [17, 26]. A survey on these issues is provided in [18]. However, each security/concern category can be sub-divided in finer-grained aspects *e.g.*, authentication, integrity, network communications, in the following are presented mainly research efforts addressing DDoS attacks in SDN networks.

To analyze impacts on network performance of DDoS attacks, works of [19] and [9] have showed how such attacks may impact on several parameters like the control plane bandwidth (*i.e.*, controller-switch channel), latency, switches flow tables and the controller performance. Other works as [23] and [4] use the SDN capabilities to implement schemes that allows to detect and mitigate DDoS attacks through packet analysis and filtering. These solutions reduce the impact of attacks, but they may cause an overhead in the flow-tables and the SDN controller. Also, they do not provide any solution to address these specific SDN performance issues as proposed in [5] (*e.g.*, flow-tables and controller overloading). Furthermore, they also do not consider DDoS attacks and the collaboration with AS customers as [17].

SDN-based solutions allows greater agility to enforce decisions that requires a global network view. Therefore, intra-domain security policies and mechanisms to prevent and react to DDoS attacks can be made more agile. By combining the intra-domain capabilities provided by SDN and the inter-domain advantages provided by blockchains and smart contracts, the efficiency to mitigate DDoS attacks in both inter- and intra-domains can be improved.

2.3 Smart Contracts and Blockchains

A smart contract is a piece of software made to facilitate the negotiation or performance of a contract, being able to be executed, verified or enforced on its own. A smart contract alone is not ”smart” as it needs an infrastructure that can execute, verify, and enforce the negotiation or performance of a contract by respective computer protocols. It has gained attention in the context of blockchains that provide a fully decentralized infrastructure to run, execute, and verify such smart contracts [2]. Therefore, smart contracts need to run on a blockchain to ensure (a) its permanent storage and (b) high obstacles to manipulate the contractfis content. A node participating in the blockchain runs a smart contract by executing

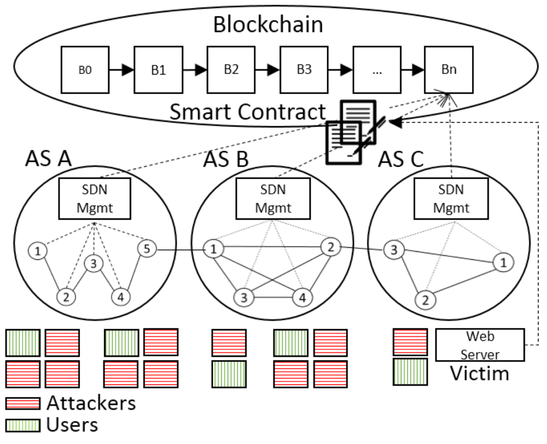

Figure 1: Application Scenario

its script, validating the result of the script, and storing the contract and its result in a block.

Although the Bitcoin [13] blockchain was the first fully decentralized distributed ledger, it is primarily designed for transfer of digital assets and it is not a Turing-complete (*e.g.*, it does not support loops). Such a Turing-complete contract language allows to define rules to block IP addresses or route traffic that can be interpreted by an SDN controller. Thus, a blockchain can be used to store and distribute rules in a fully decentralized manner. While several projects try to address these issues, the Ethereum [24] blockchain is the most popular to supports a Turing-complete contract language. In Ethereum, smart contracts runs in a sand-boxed Ethereum Virtual Machine (EVM) and every operation executed in the EVM has to be paid for to prevent Denial-of-Service (DoS) attacks.

In the literature, there exist a few blockchain applications for managing security issues in networks. Securechain [16] is a solution that uses Ethereum as a blockchain-based approach for enhancing the network security in two cases, when a) adding new devices and b) to generate alerts and a forensically auditable log. A white-list of authorized entities is stored in the blockchain for verifying trusted entities able to modify the network (*e.g.*, include or remove a device). Therefore, transactions/requests from a white-listed wallet, with a valid instruction, are validated and the instruction can be implemented in the network. In addition, it generates logs of requests and attempts which ones can be verified in the blockchain.

Keyless Signature Infrastructure (KSI) [7] uses a similar approach to ensure integrity and audibility of every operation in an SDN network. The authors propose a private blockchain architecture in which the participants can sign the transactions (*e.g.*, traffic rule changes, configurations, policies) in two cases. First, by the time a transaction is produced and second, while retrieving and committing transactions to other network components. Therefore, it possible to track down to the root of every record of transaction in the network.

3 PROPOSED SYSTEM ARCHITECTURE

This section presents the design principles considered in the architectural design. First, Section 3.1 exemplifies a scenario in which our architecture can be used. Section 3.2 provides a detailed description of its main components. Implementation details are presented in Section 3.3.

3.1 Application Scenario

To illustrate the architecture application, a scenario is presented in Figure 1. In this scenario, a web server hosted at AS C is under a DDoS attack from devices hosted at various domains (ASes A, B and C). With a non-collaborative DDoS mitigation approach, the web server relies on defense mechanisms that are implemented at the AS where it is allocated, which in many cases may be distant from the origin of the attack traffic and therefore overloading several domains with attack traffic.

In this scenario, participants of the collaborative defense (ASes and web server) first need to create a smart contract, that is promptly linked with a registry-based type of smart contracts. Therefore, when attackers of domain A, B and C overload web server, the customer or the AS under attack stores the IP addresses of the attackers in the smart contract. In an Ethereum blockchain a new block is created every 14 seconds, so subscribed ASes will receive updated lists of addresses to be blocked and confirm the authenticity of the attack by analyzing the traffic statistics and confirming the authenticity of the target's address.

Once other ASes retrieve the list of attackers and confirm the attack, different mitigation strategies can be triggered according to the security policies and mechanisms available in the domain. Also, it can possibly block malicious traffic near of its origin. Near source defense is ideal for the health of the Internet because it can reduce the total cost of forwarding packets which, in the case of DDoS attacks mostly consist of useless massive attack traffic [14].

3.2 Architectural Design

As DDoS attacks continue increase and vary in their patterns, the need for coordinated responses also increases to efficiently detour the attacks. However, it is important to note that only the collaboration between customers and ASes is an additional approach to existing defense mechanisms. The architecture depicted in Figure 2 is composed by three components:

- **Customers**: may report DDoS attacks to the Ethereum blockchain via smart contracts;
- **ASes**: may report DDoS attacks and retrieve lists containing reported IP addresses and may implement their own DDoS mitigation mechanisms;
- **Blockchain/Smart Contract**: the public Ethereum blockchain (Ethereum Virtual Machine nodes) running Solidity smart contracts, which comprises the logic to report IP addresses in the blockchain.

The architecture is built considering the following principles:

(1) DDoS detection and mitigation countermeasures are provided as an on-demand services by either the ASes or third-party services;
(2) To efficiently aid coordinated attack responses, Blockchain DDoS Mitigation modules are running on the entities (customers or ASes) writing IP addresses and listening to the blockchain;

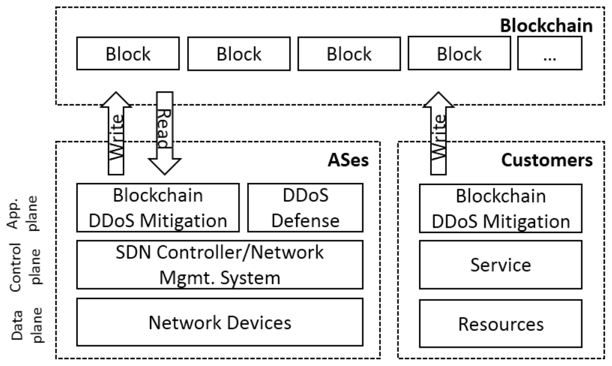

Figure 2: Proposed System Architecture

(3) Only customers or ASes with proof of ownership of their IP may report the attack to the smart contract;
(4) Different domains implement different security policies as well as different underlying management systems. Once notified of a DDoS attack in which the customer has its authenticity confirmed, countermeasures are defined according to the domain security policies and available actions.

To mitigate DDoS attacks and reroute traffic (1) different techniques can be used upon the DDoS detection by customers, which typically involves analyzing Internet traffic with complex attack detection algorithms, followed by filtering. In this regard, a collaborative approach decreases the overhead in the detection phase. Blockchain DDoS Mitigation modules (2) on the customer side are made simpler as Ethereum is a public and already available blockchain technology, which can be used to perform rapid and widespread DDoS advertisement across multiple domains using smart contracts. To proof IP ownership (3), services with challenge/response authentication can be used by an AS to ensure that the IP address of the customer reporting the attack is actually the customer under attack, and to enforce the necessary countermeasures (4) in accordance with the security policies implemented in the domain.

The smart contract logic illustrated in Figure 3 is deployed as complement to existing DDoS mitigation mechanisms. Using blockchain technology enables fast mitigation across multiple domains. Therefore, any domain who implements the system should consider the aforementioned principles in its design. First, any domain (*e.g.*, customers or ASes) participating must create a smart contract identified with an IP address or range of addresses certified by an authority. Then, the smart contract is registered in a registry-based type of smart contract so that participation can be easily tracked and thus relevant smart contracts can be identified.

Traffic arriving at both the customer and AS can be analyzed and filtered using existing monitoring and measurement tools (*e.g.*, NetFlow, sFlow, custom SDN implementations). Components of the controller and the Blockchain DDoS Mitigation appliances to retrieve and report the IP addresses are provided so that both customers and ASes can securely communicate with the smart contract. The analysis of traffic in a gateway is facilitated by SDN and therefore it is intended to use a monitoring framework based on the OpenFlow protocol and the Ryu controller. Also, solutions proposed in [5] to avoid overburdening flow-tables with several blocking flow-rules and the communication channel between switches and controller are considered.

3.3 Implementation Details

Listing 1 and 2 outlines the main features of the smart contract to store source IP addresses that should be blocked. For simplicity, only IPv4 addresses are shown here.Either the customer or the AS can create the smart contract. In any case, a certificate of IP ownership is required. For the customer, the certificate can be created with an automated challenge-response system, while the AS requires a certificate matching their entry in the AS registration.

The one that created the smart contract (owner of the account that created the contract) can add other addresses that are also allowed to add IPs to block. Before such an address is added, it is checked if the address matches its parent subnet. Both AS and customer can store src IP with an expiration time. The time is measured in blocks and the access to the stored data is public and can be accessed by anyone.

Before retrieving the list of IP pairs (source/destination) that should be blocked, the verifyIP() function needs to be called in order to make sure that the destination IP address has a proof of ownership. The issuing of a certificate (certOwnerIPv4) is the only remaining central entity in the architecture. After that any AS (does not need to be the customers AS) can use these IPs to block traffic in its network.

The smart contract needs first to register itself in an other smart contract registry, which stores all relevant smart contracts that should be watched. Thus an AS listens for these changes and any addition can be monitored and assessed against the network properties of the AS and apply a blocking rule if necessary.

```
contract SDNRulesAS {
  struct DropIPv4 {
    uint32 expiringBlock;
    uint32 src_ip;
    DstIPv4 dst_ip;
  } DropIPv4[] drop_src_ipv4;

  struct DstIPv4 {
     uint32 dst_ipv4;
     uint8 dst_mask;
  } DstIPv4 dstIPv4;
  bytes certOwnerIPv4; address owner;
  mapping (address => DstIPv4) customerIPv4;

  function SDNRulesAS(uint32 dst_ipv4, uint8
       dst_ipv4_mask, bytes _certOwnerIPv4) {
    owner = msg.sender;
    certOwnerIPv4 = _certOwnerIPv4;
    dstIPv4 = DstIPv4(dst_ipv4, dst_ipv4_mask);
    //TODO: register in a registry contract
  }
  //suicide and deregistering function here

  function createCustomerIPv4(address customer, uint32
       dst_ipv4, uint8 dst_ipv4_mask) {
    if(msg.sender == owner &&
        isInSameIPv4Subnet(dst_ipv4, dst_ipv4_mask)) {
      customerIPv4[customer] =
        DstIPv4(dst_ipv4, dst_ipv4_mask);
    }
  }
```

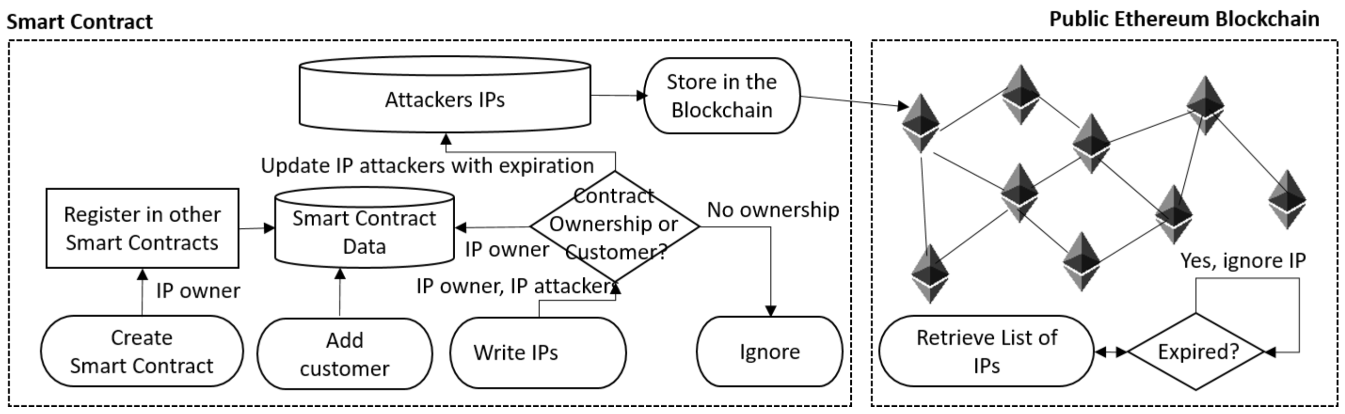

Figure 3: Proposed System Flowchart

```

  function isInSameIPv4Subnet(uint32 dst_ipv4, uint8
      dst_mask) constant returns (bool) {
    // true if customer IP is in same subnet
  }
  ...
}
```

Listing 1: Smart contract structures and core functionality

```
...
function blockIPv4(uint32[] src, uint32 expiringBlock) {
  if (msg.sender == owner) {
    for (uint i = 0; i < src.length; i++) {
      drop_src_ipv4.push(DropIPv4(
        expiringBlock, src[i], dstIPv4));
    }
  }
  DstIPv4 customer = customerIPv4[msg.sender];
  if(customer.dst_ipv4 != 0) {
    for (i = 0; i < src.length; i++) {
      drop_src_ipv4.push(DropIPv4(
        expiringBlock, src[i], customer));
    }
  }
}

function verifyIP(bytes pubKey) constant returns (bool) {
  //check if signature in certOwnerIPv4 is correct
}

function blockedIPv4() constant returns (uint32[]
    src_ipv4, uint32[] dst_ipv4, uint8[] mask) {
  uint32[] memory src; uint32[] memory dst; uint8[]
      memory msk;
  for (uint i = 0; i < drop_src_ipv4.length; i++) {
    if(drop_src_ipv4[i].expiringBlock > block.number) {
      src[src.length] = drop_src_ipv4[i].src_ip;
      dst[dst.length] = drop_src_ipv4[i].dst_ip.dst_ipv4;
      msk[msk.length] = drop_src_ipv4[i].dst_ip.dst_mask;
    }
  }
  return (src, dst, msk);
}
```

Listing 2: Smart contract blocking functions

4 PRELIMINARY EVALUATION

The use of the Ethereum Virtual Machine (EVM) allows the multiple domains involved in an attack scenario to invoke functions in a smart contract reporting attacks. This public and already available structure provides many benefits when compared to existing collaborative DDoS mitigation implementations. First, the use of an existing and distributed storage infrastructure reduces the complexity in the development and adoption of the approach as it supersedes the development and standardization process of a gossip-based protocol, which needs to be embraced by the various ASes and customers. Also, the EVM smart contracts supports in a decentralized and native way the logic to control who is reporting an attack and who are the attackers.

However, this smart contract works well for small number of attacks, while for large-scale attacks, our approach is currently costly the contract size but we will address this issue in a future work to make reference to a larger list of IP addresses. Therefore, in order to keep the complexity of the architecture low, only the data (*e.g.*, IP addresses) should be stored in the contract, and it may become necessary to add a reference as shown in Listing 3, where the full list of addresses can be retrieved. The cost of adding 50 source IPs directly in a freshly deployed contract is 2.5 mio gas (gas is the internal pricing for running a transaction or contract in Ethereum) at the current gas price [6] of 20 gwei, which is 0.05 ETH at the current market price of 9.3 USD is in total 0.46 USD, while 100 source IPs cannot be mined in one contract and multiple contracts have to be used as it exceeds the 4 mio gas limit.

```
struct DropIPv4 {
uint32 expiringBlock;
uint32 src_ip;
//e.g. https://example.com/blockedips.txt
string src_ipv4_ref;
DstIPv4 dst_ip;
}
DropIPv4[] drop_src_ipv4;
```

Listing 3: Storing references

5 SUMMARY AND FUTURE WORK

In this paper a collaborative architecture using smart contracts and blockchain is proposed to enable DDoS mitigation across multiple domains. Being a distributed and essentially public storage, the blockchain determines a simple and efficient structure to develop a collaborative approach towards DDoS attacks mitigation. The proposed and implemented architecture can be deployed as supplementary security mechanism to existing DDoS protection schemes. Combined with existing mechanisms this process can be fully automated, and the reaction time against DDoS attacks involving multiple AS domains can be reduced. This mechanism provides ASes the possibility to deploy their own DPS and generate added value for their customers without transferring control of their network to a third party. The main contributions are summarized as: the design and development of an architecture making use of an existing structure to mitigate DDoS attacks in multiple domains, the simplicity in the adoption of the approach since Ethereum and smart contracts are publicly available, and the agility to enforce rules on the ASes-side by the use of SDN.

To support a larger number of addresses, future work will investigate ways to compress the list, *e.g.*, with a bloom filter and its advantages and disadvantages will be evaluated. Another limitation is that blocking of destination IPs should be only for static IPs. Thus, the automatic service that issues these certificates of proof of IP ownership need to check for dynamic IPs first, *e.g.* using services such as SORBS (dul.dnsbl.sorbs.net). Also the the current smart contract supports only one hierarchy, thus, createCustomerCertIPv4() in Listing 1 needs to be extended to allow more hierachies to map subnets accordingly.

REFERENCES

[1] Akamai. 2016. How to Protect Against DDoS Attacks - Stop Denial of Service. https://www.akamai.com/us/en/resources/protect-against-ddos-attacks.jsp. (2016). [Online, accessed 2017-1-10].

[2] Thomas Bocek and Burkhard Stiller. 2017. *Smart Contracts - Blockchains in the Wings.* Springer, Tiergartenstr. 17, 69121 Heidelberg, Germany, pp. 1–16.

[3] CloudFare. 2016. CloudFlare advanced DDoS protection. (2016). https://www.cloudflare.com/static/media/pdf/cloudflare-whitepaper-ddos.pdf

[4] Nhu-Ngoc Dao, Junho Park, Minho Park, and Sungrae Cho. 2015. A feasible method to combat against DDoS attack in SDN network. In *2015 International Conference on Information Networking (ICOIN)*. pp. 309–311. DOI: http://dx.doi.org/10.1109/ICOIN.2015.7057902

[5] Lobna Dridi and Mohamed Faten Zhani. 2016. SDN-Guard: DoS Attacks Mitigation in SDN Networks. In *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*. pp. 212–217. DOI: http://dx.doi.org/10.1109/CloudNet.2016.9

[6] Ether Fund. 2016. Ether Unit Converter. (jan 2016). http://ether.fund/tool/converter

[7] Guardtime. 2016. Use of a globally distributed blockchain to secure SDN. (jan 2016). http://www.ciosummits.com/Guardtime_KSI_Use_of_a_globally_distributed_blockchain_to_secure_SDN_whitepaper_1602.pdf

[8] Mattijs Jonker, Anna Sperotto, Roland van Rijswijk-Deij, Ramin Sadre, and Aiko Pras. 2016. Measuring the Adoption of DDoS Protection Services. In *Proceedings of the 2016 ACM on Internet Measurement Conference (IMC '16)*. Santa Monica, California, USA.

[9] Rajat Kandoi and Markku Antikainen. 2015. Denial-of-service attacks in OpenFlow SDN networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, pp. 1322–1326.

[10] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. 2015. Software-defined networking: A comprehensive survey. *Proc. IEEE* 103, 1 (2015), pp. 14–76.

[11] Steve Mansfield-Devine. 2015. The Growth and Evolution of DDoS. *Network Security* 10 (2015), pp. 13–20.

[12] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 38, 2 (2008), pp. 69–74.

[13] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[14] K. Nishizuka, L. Xia, J. Xia, D. Zhang, L. Fang, and C. Gray. 2016. *Inter-organization cooperative DDoS protection mechanism.* Draft. https://tools.ietf.org/html/draft-nishizuka-dots-inter-domain-mechanism-02 Draft.

[15] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. 2007. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys (CSUR)* 39, 1 (2007), pp. 3.

[16] Reply. 2016. Securechain Blockchain-based Security for Software-Defined Networks (SDN). (jan 2016). http://www.reply.com/Documents/Securechain-Brochure_Sytel_Reply_ENG.pdf

[17] Rishikesh Sahay, Gregory Blanc, Zonghua Zhang, and Hervé Debar. 2015. Towards autonomic ddos mitigation using software defined networking. In *SENT 2015: NDSS Workshop on Security of Emerging Networking Technologies*. Internet society.

[18] Sandra Scott-Hayward, Gemma O'Callaghan, and Sakir Sezer. 2013. Sdn security: A survey. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For.* IEEE, pp. 1–7.

[19] Seungwon Shin and Guofei Gu. 2013. Attacking software-defined networks: A first feasibility study. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, pp. 165–166.

[20] Deutshe Telekom. 2016. Deutsche Telekom hack part of global internet attack. (nov 2016). http://www.dw.com/en/deutsche-telekom-hack-part-of-global-internet-attack/a-36574934

[21] The Associated Press. 2016. Hackers Used 'Internet of Things' Devices to Cause Friday's Massive DDoS Cyberattack. http://www.cbc.ca/news/technology/hackers-ddos-attacks-1.3817392. (22 Oct 2016). [Online, accessed 2017-1-10].

[22] Thaneswaran Velauthapillai, Aaron Harwood, and Shanika Karunasekera. 2010. Global detection of flooding-based DDoS attacks using a cooperative overlay network. In *Network and System Security (NSS), 2010 4th International Conference on.* IEEE, pp. 357–364.

[23] Lei Wei and Carol Fung. 2015. FlowRanger: A request prioritizing algorithm for controller DoS attacks in Software Defined Networks. In *2015 IEEE International Conference on Communications (ICC)*. IEEE, pp. 5254–5259.

[24] Gavin Wood. 2016. Ethereum: A secure decentralised generalised transaction ledger. (jan 2016). http://bravenewcoin.com/assets/Whitepapers/Ethereum-A-Secure-Decentralised-Generalised-Transaction-Ledger-Yellow-Paper.pdf

[25] Saman T. Zargar, James Joshi, and David Tipper. 2013. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys Tutorials* 15, 4 (Fourth 2013), pp. 2046–2069. DOI: http://dx.doi.org/10.1109/SURV.2013.031413.00127

[26] Guangsen Zhang and Manish Parashar. 2006. Cooperative defence against ddos attacks. *Journal of Research and Practice in Information Technology* 38, 1 (2006), pp. 69–84.