On the State of the Inter-domain and Intra-domain Routing Security

Mingwei Zhang
University of Oregon, Eugene, OR, USA
Email: mingwei@cs.uoregon.edu

***Abstract*—Routing is a key component for building an interconnected network architecture. There are inter-domain and intra-domain routing protocols. The inter-domain routing protocol has experienced increasingly frequent anomalies, such as IP prefix hijackings, route leaks, or impact from large-scale disruptive routing events. The intra-domain routing also suffers from various attacks originated from within an autonomous system, such as topology manipulation and host-based flooding attack. Security upgrades to the existing protocols and accurate detection mechanisms have therefore been proposed and experienced. In this study, we conduct a comprehensive survey on the existing security mechanisms for both inter-domain and intra-domain routing protocols. For inter-domain routing protocol, we study the *de facto* protocol – Border Gateway Protocol (BGP). For intra-domain routing protocol, we investigate the recent software-domain networking paradigm and the OpenFlow protocol. For each routing protocol, we investigate both attack prevention solutions and attack detection solutions. We summarize the strengths and weaknesses of every existing solution, and discuss the missing gaps that need further research.**

I. Introduction

The Internet consists of many domains, each of which has autonomous control over its own networking infrastructure. Such domains are also called Autonomous Systems (ASes). People designed routing protocols to connect hosts and routers within one domain and exchange information between domains. The routing protocols can be categorized into inter-domain and intra-domain protocols. The inter-domain routing protocols aim to exchange routing information between domains, allowing each domain to decide the routes toward any destinations on the Internet. The *de facto* routing protocol for inter-domain routing is Border Gateway Protocol (BGP) [1]. The intra-domain protocols exchange reachability between different networking devices within one domain (AS). Traditional intra-domain routing protocols include RIP [2], OSPF [3], IS-IS [4], EIGRP [5], etc. The recently emerged new routing paradigm, the software-defined networking (SDN) [6] and OpenFlow [7] are quickly getting adopted due to their features such as programmability, unified interface, and centralized control mechanisms. Network operators can also implement traditional routing protocols on a SDN platform. Such features make SDN and OpenFlow more preferable to the traditional protocols. Fig. 1 shows an example of inter-domain and intra-domain routing, where machine A talks to machine B, and the traffic travels through a set of routers. The routing from router 1 to 4 is within AS1 and is intra-domain routing; while the routing from router 4 to 7 is inter-domain routing.

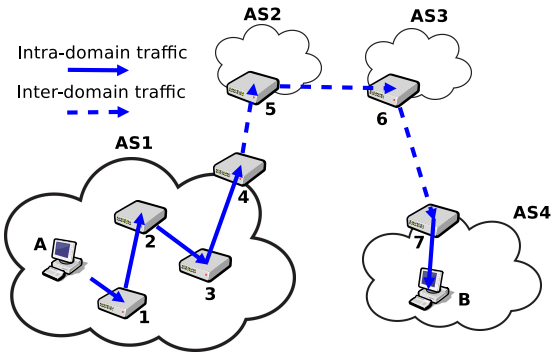


Fig. 1: An example of inter-domain and intra-domain routing.

As the Internet relies on the routing protocols for its normal operations, it is very important to ensure the routing protocols are secure against security threats. Unfortunately, neither intra-domain nor inter-domain routing protocols are bulletproof against various threats. In this report, we closely examine the security properties and the existing security solutions of both protocols. Specifically, we examine BGP as the main inter-domain routing protocol, and examine SDN and OpenFlow as the representative of the intra-domain routing protocol.

Regarding the security for inter-domain routing, originally BGP was not designed to carry many security properties. The Internet experienced a number of inter-domain anomalies such as IP prefix hijackings, large-scale route leaks, or Internet “earthquakes” caused by

various reasons. BGP is designed with the assumption that everyone on the Internet does not act maliciously, and it lacks sufficient verification mechanisms for the update messages. However, such an assumption does not hold anymore in today's Internet. Numerous routing incidents show that even some Internet providers at national level conduct malicious activities on the Internet and pose severe security and privacy threats to Internet users [8], [9], [10], [11].

Regarding the security for intra-domain routing, SDN also suffers from severe security problems. SDN architecture consists of the end-hosts, switches, controllers, and applications, and each component could suffer from security attacks. The applications running on top of the controllers may encompass security loopholes or malicious exploits; the controllers can be made unavailable to legitimate needs when receiving a large number of fake requests; the switches can be compromised to act maliciously when forwarding traffic; and the end-hosts can also exploit the loophole in OpenFlow and disrupt the SDN controllers.

In investigating BGP and SDN security solutions, we focus on analyzing their strengths and weaknesses, as well as their deployment status on the Internet. We categorize the security solutions into two general categories: the attack prevention solutions and the attack detection solutions (Fig. 2). The attack prevention solutions aim to proactively stop potential attacks through security upgrade of either the protocol design or the protocol operations. Meanwhile, the attack detection solutions aim to reactively detect abnormal events regarding the operation of the protocols, providing triggers for timely reaction to such events. The attack prevention mechanisms for BGP have been heavily studied for a long time. However, none of these mechanisms has been largely deployed to date, leaving the Internet still vulnerable to the inter-domain routing attacks. On the contrary, since SDN is still young, the majority of the SDN security work look at attack prevention, with the attack detection for SDN less explored.

This report is organized as follows. We survey the existing BGP attack prevention solutions in section III, and review the main BGP attack detection mechanisms in section IV. In section V, we look at the solutions that try to secure the SDN architecture and operations. Section VI is then focused on how SDN can be applied to solve other security problems. At last, in section VII we summarize the survey and discuss some related issues about the security of Internet routing.

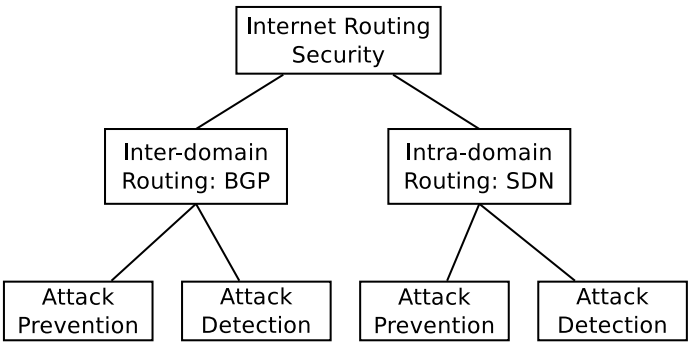


Fig. 2: Internet routing security taxonomy.

II. Background

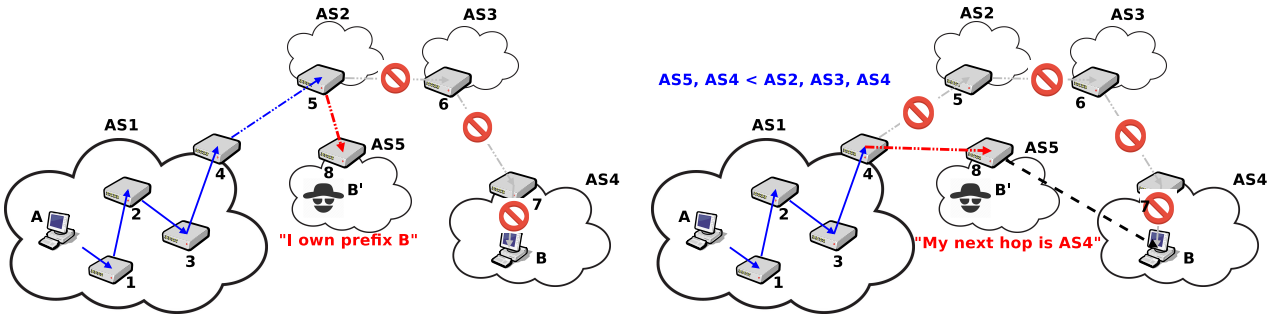
Internet routing security has been a hot research topics since late 1990s. There are many related projects that try to improve the security of the Internet routing, from the inter-domain (BGP) and intra-domain (SDN) perspectives. In this section, we provide some background of the BGP and SDN security research in general.

A. BGP Security

The Internet started with only a few connected networks for research and military purposes. Until late 1980s, there was no clear definition of the autonomous systems (ASes). This lack of domain-level hierarchy hindered the scalability of the Internet. In 1989, the first version of Border Gateway Protocol (BGP) was proposed. BGP clearly defines the concept of AS and the operations between ASes for exchanging routing information. Since then, the Internet started the exponential expansion.

However, the BGP is not perfectly secure. In 1998, Labvotiz et al. first studied the instability of the Internet routing. This paper is one of the earliest papers that studied the vulnerabilities of the Internet. Researchers also discovered two major attacks that can severely disrupt the Internet: prefix hijacking and AS path spoofing. Fig. 3 shows the examples of these two attacks. Since 2000, there were many projects focused on preventing such attacks on BGP (Fig. 4): S-BGP [12] in 2000, soBGP [13] in 2002, IRV [14] in 2003, SPV [15] in 2004, psBGP [16] in 2005. These projects showed that BGP operations can be secured with upgrades of the protocol and the operations.

However, all of these projects relied on certain infrastructure to distribute the routing information securely. It was not until the establishment of Resource Public Key Infrastructure (RPKI) in early 2010s do the BGP security projects have such a reliable infrastructure to submit and access verifiable routing information. With the deployment of RPKI, BGPsec was then proposed and quickly



(a) An example of a BGP prefix hijacking attack (b) An example of a BGP AS path spoofing attack

Fig. 3: Examples of BGP prefix hijacking and AS path spoofing attacks

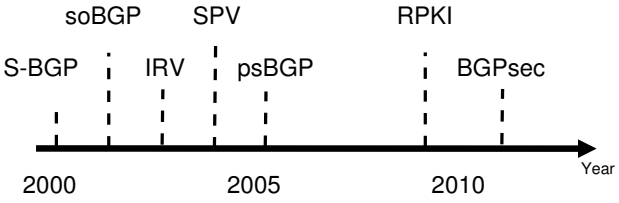


Fig. 4: Timeline of the major BGP attack prevention projects.

being deployed. Unfortunately, the deployment rate of RPKI and BGPsec is still far from sufficient to date. With the low deployment rate, the aforementioned BGP attack prevention solutions cannot effectively prevent the stop the attacks on BGP. As a result, it is very important for people to be able to detect and react to the attacks quickly and accurately.

B. SDN Security

Software-defined networking is a very new networking technology, and has not yet been recognized as the *de facto* approach for intra-domain routing. However, we believe SDN is the future of the intra-domain routing. First, it is fully compatible with all the traditional intra-domain routing protocols. Using the centralized approach for controlling the network, a network operator can implement any existing or new routing protocols as applications running on the controller. Second, the separation of the control logic and forwarding actions makes SDN can not only achieve the goal of the traditional routing protocols but also many other new tasks. These features make SDN very popular among the large networks, where the operators require maximum flexibilities of the networking functionalities as well as the centralized management over the entire network.

In terms of security, researchers have discovered several new attacks on SDN. In section V, we examine the main security solutions against the attacks. However, there are also many security aspects that are yet to be studied. Instead of securing SDN itself, there are several projects that use SDN for securing the Internet, such as conducting anomaly detection or defending against DDoS attacks. In section VI, we also investigates the applications of SDN on solving other security problems of the Internet.

III. BGP Attack Prevention

The design of Border Gateway Protocol (BGP) was based on the assumption that all the autonomous systems (ASes) are trustworthy. The assumption no longer holds as we have seen an increasing frequent appearance of the malicious attacks on the Internet carried out by ASes that exploit the loopholes in BGP. The current version of BGP allows ASes to announce origination of any prefixes without authentication (Fig. 3a), propagate routes with manipulated path information(Fig. 3b), or even send out entirely forged routing information. Due to the lack of verifiable global routing information, an AS cannot effectively verify the information received from other ASes, and can only rely on its own knowledge about the legitimacy of the updates, which has proven to be ineffective by the repeated occurrences of the malicious attacks and misconfigurations.

Researchers have proposed multiple solutions ranging from cryptographic to multi-party collaborative approaches to securing the BGP operations and preventing the attacks entirely. In the following subsections, we introduce the design and core ideas of the majority options of attack prevention mechanisms. For each mechanism, we also discuss its essential drawbacks and its deployability. At last, we discuss the overall future of the attack prevention mechanisms.

A. Overview

From the main technology used, the main attack prevention system can be categorized into three types:

method	control-plane	crypto-based	overhead	data source	path	origin	deployment
BGPSec [9], [12], [17]	✓	✓	high	PKI	✓	✓	✓
soBGP [13]	✓	✓	high	peer	✓	✓	
psBGP [16]	✓	✓	high	peer	✓		
SPV [15]	✓	✓	low	registry	✓		
Listen & Whisper [18]	✓	✓	medium	peer	✓	✓	
RAVS [19]	✓	✓	low	peer	✓	✓	
IRR	✓		low	registry	✓		✓
IRV [14]	✓		medium	peer	✓	✓	
GTSM [20]			low	N/A			✓
TCP MD5 Sig. [21]			low	N/A			✓
IPSEC [22]			medium	N/A			✓

TABLE I: BGP Attack Prevention Mechanisms

1) control-plane cryptographic approaches,
2) control-plane non-cryptographic approaches, and
3) data-plane-based approaches.

The control-plane approaches aim to secure the control-plane information exchanged among ASes, while the data-plane approaches focus on securing the communication channel between the BGP routers. The control-plane-based solutions can also be coarsely categorized into cryptographic and non-cryptographic approaches. Table I lists the current main BGP attack prevention systems. The *method* column represents the name of every attack prevention mechanism; the *control-plane* column indicates if the method mainly operates on the control-plane; the *crypto-based* column shows if a method applies cryptographic approaches or not; the *overhead* column represents the operational cost of each mechanism, ranging from low to high; the *data source* column indicates the type of data sources used by these methods; the *path* and *origin* columns show that whether the approach can secure the AS paths and prefix origin of the BGP updates respectively; and the *deployment* column shows if the method is currently being deployed, regardless of the deployment ratio.

B. Prevention Without Cryptography from the Control Plane

We start our survey for BGP attack prevention methods by investigating the prevention mechanisms that do not heavily depend on cryptographic methods. Specifically, we examine two main methods in this area: the Internet Routing Registry (IRR), and Inter-domain Route Validation (IRV). IRR uses centralized trusted databases to maintain and offer access of correct routing information; IRV enables active queries for the correctness of BGP updates, trusting each AS to provide the accurate routing information about itself. In this following subsection, we closely investigate theses two methods and analyze their strengths and weaknesses.

1) Internet Routing Registry (IRR): People first built the **Internet Routing Registry (IRR)** to serve as general repositories of routing information, connectivity, and routing policies. IRR consists of several databases where network operators publish their routing policies and announcements so that other network operators can utilize the data. The information includes ASes' relationships with other ASes, the routes learned and propagated from other ASes, the preferences if multiple routes exist, etc. Such information is structured into data objects using the Routing Policy Specification Language (RPSL) [23], [24]. Network operators can verify each BGP update against the known routing information obtained from IRR databases.

However, IRR suffers from out-of-date information. The information in IRR databases may be accurate at the time of submission, but this may not be true by the time users access the information. The organizations do not have enough motivation to keep their IRR records up to date, especially for those ASes that update routing information frequently. Users of IRR information thus cannot confidently decide if the suspicious BGP updates contains anomalous information or simply newer legitimate information. Despite the weakness, researchers still use IRR on various topics [25], [26], [27], [28]. Signaos et al. even evaluated the efficacy of using the inaccurate IRR and claimed that it is still very helpful [29]. However, such weakness makes IRR less reliable in terms of verifying BGP information. As a result, people have to seek other approaches to obtaining authentic BGP information.

2) Inter-domain Route Validation (IRV): In the global registry model used by IRR, ASes do not have enough motivation to update a third-party registry of regarding their routing information. To address this shortcoming, Goodell et al. proposed the **Inter-domain Route Validation (IRV)** [14] architecture that extends the existing model into per-AS routing registry. IRV provides out-of-band verification information using a query-based ap-

proach. It defines an information distribution protocol for ASes to exchange routing data with each other without the involvement of a third party. Each IRV-enabled AS implements an IRV server that stores all local routing information. The routers that receive BGP updates can query the IRV servers of the ASes on the path to validate the information. Upon receiving a query, the IRV server will respond based on its local policy. Since every IRV server resides within each AS, the information can be kept up-to-date with minimum cost.

IRV also has a set of issues that were not specified in the paper. First, the discovery of an IRV server within an AS is not clear. A querying AS does not necessarily know every IRV servers' IP addresses. However, the authors did not introduce any mechanism for an AS to discover the IP addresses of other ASes' IRV servers, Second, the message authentication of the IRV query and reply was not specified, leaving it possible for attackers to forge illegitimate IRV messages. Third, in partial deployment scenarios where only a set of ASes on the Internet enables IRV, a querying AS can only verify a portion of the AS path.

C. Prevention with Cryptography from the Control Plane

The majority of the BGP attack prevention systems try to secure the assets (prefix origin and AS paths) through cryptographic approaches. The main argument behind this is that there are hardly any trustworthy verification sources that an AS could refer to, leaving fewer options but to establish and use cryptographically-secured data sources. In this subsection, we survey the main attack prevention solutions that use cryptographic approaches. To date, BGPsec is the only BGP security upgrade that has been deployed on the Internet. Though the deployment rate is still bleak, we could foresee a better scenario in near the future. In the rest of this subsection, we look at other attack prevention solutions proposed prior to BGPsec, but not deployed on the Internet.

1) RAVS: Kim et al. proposed a solution with verifiable search called **Identity-based Registry with Authorized and Verifiable Search (RAVS)** [19]. RAVS features the following capabilities that out-performs the IRR method. First, it enables public key exchange cryptographically transform the AS number to the public key of each AS. This allows every AS to easily authenticate itself to the RAVS system without requiring a globally deployed public-key infrastructure. Second, RAVS uses Search Permission Generator to control search permissions based on AS credentials. Only the authorized ASes can query the registry. Third, every search result can be verified cryptographically. All entries in the RAVS database are signed with the private key of the owner ASes, allowing other ASes to verify them with the public key of the owner ASes. Such scheme allows RAVS to provide verifiable routing information to the ASes. Unfortunately, RAVS has never been adopted on the Internet.

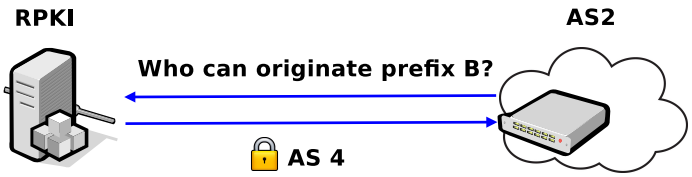


Fig. 5: An example of querying RPKI for prefix ownership information.

2) RPKI: As another attempt to construct a trustworthy database for routing information, people proposed and built **Resource Public Key Infrastructure (RPKI)**. The main purpose of RPKI is to provide a centralized repository for all resource-related information with cryptographic protection. One of the main types of resource is the ownership information of IP prefixes.

As discussed in section III-A, one of the main threats toward the Internet routing is prefix hijacking (Fig. 3a). Exploiting the lack of verification mechanism in BGP protocol design, the attackers can send announcements to claim the ownership of any prefixes, or to change the AS-level path toward a target prefix. In an ideal scenario where every AS on the Internet knows the legitimate owner of every IP prefix, forged announcements from the attackers will not be propagated. However, in real world scenarios, it is hard, if not impossible, to obtain the correct and up-to-date prefixes ownership information.

RPKI is a public key infrastructure specifically designed to store and provide information of the resources (or assets of ASes) on the Internet. RPKI is not intended to replace the current IRR system, but to provide extra security property for the information. For example, when an BGP router received an announcement of a prefix B originated from AS X, the router can query RPKI repository (or a local cache) for the ownership information of this prefix, and then verify the correctness of the information (Fig. 5). The result could be valid, invalid, or unknown. Based on the verification result and local security policy, the receiving router can then make the routing decision. Wahlisch et al. [30], [31] described the procedure of detecting suspicious prefix ownership changes, and Huston et al. [9] also provided a more comprehensive description of RPKI architecture and its usage.

However, RPKI is also facing a number of problems. The first problem is the scaling issue. To date, RPKI

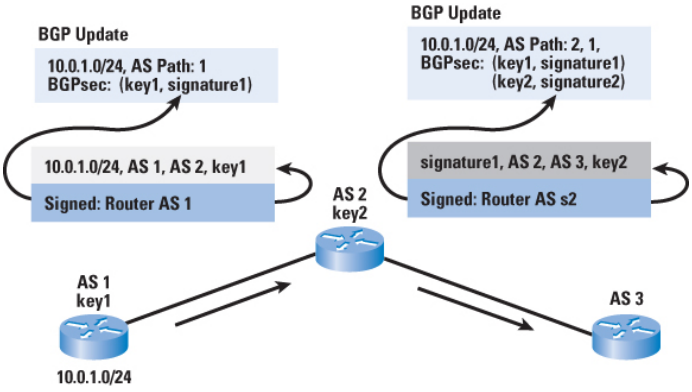


Fig. 6: An BGPsec AS path protection example.

only covers less than 10% of the IP space. Through a set of calculation, Osterweil et al. estimated that the size of RPKI with full deployment will consist of 650,000 encrypted objects [32], which will incur a more than 4 day time overhead for a full synchronization. If consider the key rollover cases, the overhead would be even higher. The second problem is that RPKI system itself can also be abused to take down prefixes. The revocation of any objects does not need any acknowledgement from the current prefix owner [33]. In [34], Heilman et al. proposed a set of countermeasures to maintain countable RPKI operations with explicit acknowledge responses for every potential damaging operations. Nonetheless, RPKI is still considered to be the best information source for BGP security mechanisms.

3) BGPsec: With RPKI protecting the prefix origin information, **BGPsec** (originally S-BGP) is proposed to further ensure that the path toward any prefixes is also protected [9], [12], [17]. First, in every BGP update announcing a new AS path to a prefix, the path segments in the AS path are protected with a signature from each hop along the path that the update propagated. A router receiving a BGP update can check the signature-packed "BGPsec_Path" attribute. Each signature any hop appends also contains a "Subject Key Identifier" that uniquely represents a router or AS's identity in the RPKI, which the receiving party can use to verify the signature. Fig. 6[1] shows an example where AS1 originates an update for prefix 10.0.1.0/24, propagates the update to AS2, and AS2 propagates it to AS3. On each propagation, the AS in question will sign the AS path content. The receiving AS (AS3) can validate all the signatures from every AS on the path and validate the entire AS path.

BGPsec is considered the best and most practical solutions toward securing BGP; however, it still faces several challenges. Lychev et al. argued that with the unavoidable stage of partial deployment, BGPsec provides *"only meagre benefits over origin authentication when these popular policies are used"* [35]. To accommodate its legacy next-hop routers, a router running BGPsec has to *downgrade* its protocol to legacy BGP, and thus lose all the cryptographic protections provided by the previous hops. Once the downgrade happened at one hop along the propagation, previous signed signatures will no longer be available to the downstream ASes, neither can the downstream entities continue to use BGPsec to partially sign the path. In [36], Li et al. presented two types of attacks that work even when BGPsec fully deployed: the wormhole attack and the mole attack. The wormhole attack shows that BGPsec cannot tell or defend fake BGP links created by tunneled BGP sessions. With the help from the others, an attacker can effectively announce a totally legitimate path to the target victim with shorter path length. The mole attack exploits the fact that some ISP would rent IP prefixes from its provider and not utilize them with a default forwarding path in place. An attack could exploit such situations by simply sending traffic to the unutilized prefixes to generate a loop of traffic. Such loop of traffic can eventually saturate the link between the victim AS and its provider.

4) Other Cryptographic Solutions: As discussed previously, the only BGP security upgrade that has been deployed to date is BGPsec. There are several other solutions proposed before BGPsec that have not been adopted, including soBGP [37], psBGP [16], and SPV [15]. Unfortunately, none of these approaches have been widely deployed to date. We investigate these methods and in the rest of this section.

In 2003, White et al. [37], [13] proposed **soBGP** that uses cryptographic certificates to prevent forged prefix origin announcements and invalid AS path updates. Every AS that deploys soBGP should obtain an "EntityCert" certificate to authenticate its own identity to others. To secure the prefix origin information, soBGP uses cryptographic certificates, "AuthCert"s, to provide verifiable announcement of prefix ownerships. To announce the ownership of a prefix, an AS needs to obtain an "AuthCert" from a trusted third party. The AS can then announce the prefix with the corresponding "AuthCert" attached to the update, and sign the announcement with its own private key. A receiving router can verify the announcement by validating the signatures of the update and the "AuthCert" attached. To enable the validation of the AS path updates, soBGP requires all enabled ASes to broadcast AS relationship information using another

[1]http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_14-2/142_bgp.html

certificate, "ASPolicyCert". The relationship information of an AS includes the identities of neighbor ASes, and the policy for each neighbor AS. An "ASpolicyCert" essentially asserts the feasibility of an AS to forward traffic to other ASes. Each soBGP enabled AS will build its own Internet topology based on the "ASPolicyCert" obtained from the broadcast, and then validate the AS path updates against this topology. For example, AS A must first announce that it connects to AS B using "ASPolicyCert" before it can propagate any BGP updates containing the link between A and B..

Comparing to BGPsec or BGPsec, the advantage of soBGP is the relatively low overhead. After building the topology and the prefix ownership information, an AS can then validate all BGP updates using its own data without active query even during partial deployment. However, the solution relies on the assumption that ASes can reliably distribute (or broadcast) the policy information as well as the identity information of the ASes (such as the public keys of the ASes). Without a reliable information distribution mechanism like RPKI, soBGP is not able to deploy on the Internet.

In 2005, Wan et al. introduced **Pretty Secure BGP (psBGP)** [16] that utilizes a decentralized trust model for verifying IP prefix ownership. To announce the ownership of a prefix, an AS needs to send out an ownership assertion signed with its own public key and distribute to its neighbor ASes. When receiving an assertion, an AS will decide whether to propagate such assertion to its neighbors based on its own judgment. The number of assertions from the prefix owner and its peers indicates its level of authenticity. Similar to BGPsec, the path verification is done by validating a set of signatures attached by the ASes along the propagation paths. Different from BGPsec, psBGP allows partial path signatures using a confidence value for the validation, which reflects how likely a path is valid.

psBGP is essentially built upon a AS-level reputation system, which assumes the infeasibility of constructing a hierarchical PKI system for resource assertions. However, the reputation system would result in indeterministic decisions in many cases. The verification of the AS paths in BGP updates is also dependent on the decision logic of the confidence system, and potentially could be manipulated by the resourceful attackers. Thus, psBGP could be applied as a secondary route verification mechanism, but not a reliable method preventing the routing attacks.

In 2004, Hu and Sirbu introduced **Secure Path Vector (SPV)** [15], proposing to use symmetric cryptography to secure the BGP updates. The main goal of SPV is to secure BGP updates against AS path fabrications, including forging whole AS paths or modifying partial AS path segments. SPV uses tree-authenticated hash values for AS path validation. First, the prefix owners needs to have the knowledge of the private key associate with the prefix. The distribution of the prefixes' public/private keys is proposed to done in places like ICANN. Then, the prefix originator announces the prefixes with a set of one-time signatures together with the private keys for them. During each propagation, the sending AS sign itself into the ASPATH using the private key for the signature. The receiving AS can verify the ASPATH with all the one-time signatures through a hash-tree style authentication. Since the private keys was used and removed, the attacker cannot recreate the key and thus cannot replace a previous AS number with its own. To ensure the security of the constructed verification tree, SPV requires the originator periodically re-announce the prefixes.

Comparing to the BGPsec, the authors claim that SPV achieves significantly performance improvement by changing nested digital signature authentication with hash-tree-authentication. The performance improvement comes from the computational complexity difference between symmetric and asymmetric cryptography used in SPV and BGPsec. Though with some performance improvement against BGPsec, SPV still suffers some severe problems. First, the re-announcement frequency could greatly affect the overall traffic and computational load on the BGP routers, which was not taken into consideration in their evaluation. Second, the "epochs" of verification trees require a higher level of time-synchronization. Also, as discovered in [38], SPV cannot fully protect BGP against route forgery and eavesdropping.

D. Prevention from the Data Plane

There are some other methods that prevent attacks from data-plane level, including IPSEC, TCP MD5 field, and Generalized TTL Security Mechanism (GTSM). These methods focus on protecting the data-plane communications between different routers, and are BGP content agnostic.

TCP MD5 Signature Option [21] is a light-weight method to protect the integrity of the packets. In the case of BGP operations, TCP MD5 signature could prevent Each communication will start with a generation of secrets between the two parties. Then they will send the TCP packets with a MD5 signature of the packet using the secret. When receives a packet, the router will calculate the MD5 signature and compare it with the one attached in the packet. Without the correct secret,

the signatures will not match. Therefore, an attacker cannot easily impersonate another router using forged BGP packets.

IPSEC [22] is a suite of protocols that operate at the IP level to secure the communications between end-hosts. It is a relatively heavy-weight toolkit comparing to the MD5 option. The suites include the cryptographic protection on both the packet header and the packet content. Using IPSEC, a router could talk with the peers without being vulnerable to eavesdropping or packet manipulation. However, the processing overhead of using IPSEC to detect malicious packets is much higher than MD5 [39]. This high computational overhead could be exploited for conducting denial of service attack. An attacker can send a large volume of forged IPSEC packets to overload a BGP router, and thus downgrade or even disrupt the normal operations of the router.

The **Generalized TTL Security Mechanism (GTSM)** [20] is another type of security method that protects the communication from topological perspective. Knowing the distance of a BGP peer router, GTSM configures each BGP packet with TTL to be 255. Any inbound packets with TTL less than specific threshold will be discarded. In the case of directly peered BGP routers, any incoming packets with TTL of 254 or less should be discarded. This method effectively protects the BGP session from the potential intrusions. However, in the case of remote peering, the BGP routers are not directly connected and forcing TTL restriction could affect the legitimate connections if any routing changes happen.

IV. BGP Attack Detection

Ideally, with full deployment of some aforementioned methods such as BGPsec, the inter-domain routing could be fully secured. However, it has been shown that none of the security solutions is bulletproof against attacks. Besides, the cost of operations, the uncertainty about the effectiveness, and the lack of motivation also hindered the overall deployment progress. Before a more secure BGP security solution is widely deployed on the Internet, network operators and users have to keep fighting with the existing and new security concerns of BGP. In particular, it is important to be able to detect all kinds of attacks and react to them timely. Table II presents a taxonomy of the current solutions on BGP attack detection. From their data sources, we categorize the solutions into three types: control-plane-based mechanisms, data-plane-based mechanisms, and hybrid mechanisms where both control-plane and data-plane information is utilized to obtain a higher detection accuracy.

The majority of the solutions are control-plane based in that they obtain the input data from the control-plane message collectors or through querying on third-party repositories. We define the passive monitoring solutions as the methods that detect attacks by passively collecting and processing the control-plane information. On the contrary, the active monitoring solutions require active queries to third-party registries and construct knowledge databases from different data sources.

In this section, we survey the major inter-domain routing attack detection methods, including the control-plane, data-plane, hybrid methods and a number of attack verification mechanisms as well. We summarize each method and describe their key idea, strength, weakness, performance, and their relationship with other methods.

A. Passive Monitoring from the Control-Plane

Most BGP attack detection solutions monitor and detect attacks by passively listening to the control-plane information. Based on the types of attacks they try to detect, these attack detection solutions can further be categorized into prefix hijacking detection solutions and BGP dynamics attack detection solutions. The prefix hijacking detection solutions examine each individual prefix path updates and detect suspicious changes of paths toward or origination of specific prefixes. On the other hand, rather than focusing on each individual prefix, the BGP dynamics anomaly detection solutions detect unusual changes of routing dynamics on the Internet. Such changes may indicate the impact from large-scale disruptive events for Internet routing. In this subsection, we will look at the detection solutions on both types.

1) Prefix Hijacking Detection: Lad et al. introduced the prefix hijack alert system [53], i.e., **PHAS**. The authors argue that the prefix owner is the only one who can accurately distinguish between legitimate changes and prefix hijackings. Thus, PHAS was designed to provide the prefix owners the quick notifications on any suspicious prefix origin changes. PHAS uses a registration system that allows the prefix owners to obtain the information about any changes to the prefix origins. The authors use the concept of *origin set* to accommodate the traffic engineering needs for the users. Each *origin set* consists of a set of origins observed from the monitors within a time window t. Any changes to the *origin set* will trigger alerts to the users. The authors also mentioned several modifications to the algorithm to reduce the total amount of the notifications to the subscribers. Overall, PHAS is a simple solution to detect AS origin changes and relies on the prefix owners to determine the

method	control-plane		data-plane		anomaly verification
	active	passive	active	passive	
NeighborWatch [29]	✓	✓			
TERRAIN [40]	✓	✓			
subMOAS [25]	✓	✓			
Kruegel et al. 2003 [41]	✓	✓	✓		
Buddyguard [42]		✓			
I-Seismograph [43]		✓			
Geo-location [44]		✓			
Argus [45], [46]		✓	✓		✓
Wavelet [47], [48]		✓			
BGP-lens [49]		✓			
BGPeye [50]		✓			
Concurrency-based [51]		✓			
Machine-learning-based [52]		✓			
PHAS [53]		✓			
PGBGP [54]		✓			
rcc [55]		✓			✓
Topology-based [56]		✓			
TAMP [57]		✓			
Listen and Whisper [18]		✓		✓	✓
Host-signature based [58]		✓	✓		✓
Hop-distance based [59]			✓		
iSPY [60]			✓		
Crowd-based [61]				✓	
LDC [62]				✓	
PurgePromote [63]					

TABLE II: Taxonomy of the attack detection methods

natures of the changes. Though easy to implement and deploy, the notification could be only useful and verified by the prefix owners. This method also fails to detect any malicious AS path segment changes other than the prefix origin changes. For example, any AS path shortening attempts by the attacker will not be detected by PHAS, allowing the attackers to "steal" the traffic to the prefixes without changing the originating ASes.

Pretty Good BGP (PGBGP) solution in [54] was another attempt to detect suspicious prefix origin changes by building and utilizing a prefix-origin binding database. During the training phase of consecutive h days, the system learns from a router's routing table for all existing prefixes and their originating ASes. All the prefix-origin pairs will be accepted by PGBGP and logged into a database. During the operational phase, all the new prefix-origin pairs unseen before will be labeled as suspicious for a period of s. If the route stays in the RIB after a period of s, the route will be labeled as normal and logged into the database. A suspicious path should have the lowest priority in the path selection of a PGBGP-enabled router. Thus, a suspicious path's propagation will be delayed for at least a period of s. Comparing to PHAS, PGBGP further protects BGP by delaying suspicious updates rather than just generating notifications. Similar to PHAS, this solution does not consider the AS path forgery attacks. The quarantine mechanism also potentially poses a long delay for any legitimate change of prefix origin. Considering the chain of delay along the propagation, the overhead is not negligible, making this method less favorable for deployment.

Qiu et al. proposed a control-plane-only detection mechanism that looks further beyond the prefix-AS binding, and consider AS-AS pairs as another important metric [56]. The basic observation is that the prefix and origin AS bindings and the peering ASes bindings are relatively stable over time. The authors believe that the prefix ownership and topological information learned from the BGP RIB table can be applied to identify bogus routes. For each path change, the system will look at the prefix-origin pair and the directional AS-AS pairs. Only when both types of bindings have been seen before can this change be valid. Similarly, the **Argus** system proposed by Xiang et al. also uses the prefix-AS and AS-AS bindings to detect suspicious BGP updates.

There are two main problems for both methods. First, both methods rely on the assumption that there are no attacks during training period. However, there is no reliable way to identify training period without any attacks. If there are attack updates during the training period, these updates will not be identified as suspicious during monitoring periods. Second, it requires a long training period for the methods to build a relatively complete

database for prefix-AS and AS-AS bindings. Study by Rexford et al. shows that the vast majority of prefixes of the popular websites have the update interval of five or more days on average [64]. To capture the BGP updates of such prefixes, the systems need to have training period for weeks or even month. This significantly increases the time overhead of running these detection systems.

Li et al. [42] introduced a system called **Buddyguard**, applying the "buddies" concept in detecting malicious prefix origin changes. Instead of looking at the prefix-AS relationship, Buddyguard tries to search for connections between prefixes. The main observation is that whenever there is a path update of a prefix, there are some other prefixes that will also have new paths around the similar time. They call these prefixes the "buddies" of the target prefix, and found that there are fate-sharing properties among theses prefixes. By using the best "buddy" candidates, Buddyguard can detect unusual changes of the path toward the target prefix if the changes do not come with the changes to the buddies. Buddyguard system is resilient against intelligent adversaries that try countermeasures to avoid detection. To counter the detection from Buddyguard, a prefix hijacker needs to simultaneously hijack the majority of the buddies of the target prefixes, which would be impractical given that any prefix would have at least hundreds of buddies from different ASes.

Khare et al. in [51] described a mechanism that detects anomalies that involve hijacking of multiple prefixes simultaneously. Their basic observation is that "simultaneously originating prefixes of many other networks is highly likely to be a real hijack." They developed a scheme that detects "concurrent" hijacks by comparing the previous knowledge of prefix origins with the simultaneous announcements. The system first collects some basic knowledge about the prefix ownerships using RouteViews [65], then look for the ASes that offending the ownerships within a small time window. They discovered about 5 to 20 such events each year from 2003 to 2010, and confirmed overall 53 events from 2008 to 2010 through direct emails with the network operators. However, this method is limited to detect only the events with multiple simultaneous hijackings. An attacker can avoid the detection by targeting only a small amount of prefixes at a time. This method also assume that the training period is clean from any attacks, which could be false in many cases. Besides, this method also does not consider the AS path manipulation, where the attackers can hijack traffic by announcing forged shorter AS paths toward the target prefix. These limitations greatly narrow the application scenarios of this method.

2) Routing Dynamics Anomaly Detection: Researchers also look at the dynamics of BGP and try to detect abnormal changes from a global perspective. Zhang et al. [47] applies wavelet transformation to capture the normal status of BGP dynamics. Wavelet transformation can reveal the temporal structures in signals carried by the dynamics of BGP streams: the individual updates could be viewed as high-frequency signals, and the group (or bursty) updates could be viewed as low-frequency signals. By using clustering algorithms on top of the wavelet representation, the proposed system can reveal different categories of the BGP dynamics. Using the learned clusters, the system could detect any outliers that does not fit into the normal clusters and thus detect the anomalies. Similarly, Yuan et al. [48] applied the same wavelet-based approach for BGP dynamics anomaly detection. They assigned deviation score to indicate local noisy levels and the outliers. Prakash et al. [49] used wavelet transformation to create the "tornado" and "clothline" figures to visually represent the BGP dynamics and detect anomalies.

In [43], Li et al. introduced an Internet monitoring system, **I-seismograph**, that goes beyond hijackings and include all different kinds of events. The system collects all prefixes updates and look for various attributes of BGP updates such as withdrawal-announcement-duplicates (WAs), announcement-announcement-duplicates (AAs) for prefixes. The attributes are then aggregated to databins for each minute during training and monitoring periods, each represented by a 10-dimension data vector. To train the normality model, the system uses all the vectors collected during the training phase and derive the majority cluster from a hierarchical clustering procedure. Databins in the cluster collectively represent the normal values for the ten attributes. During monitoring phase, the system will compare databins of very minute during the phase with the trained normality model and calculate the deviation vector of each database. This mechanism detects large-scale anomalies that generate a global impact on BGP.

B. Active Monitoring from the Control Plane

In this subsection, we survey a set of monitoring mechanisms that apply active monitoring for attack detection. The active monitoring approaches require active queries to third-party registries and construct knowledge databases from different data sources. Different from passive monitoring approaches, active monitoring approaches do not need to process BGP information from the past for training.

Siganos et al. [29] argued that the IRR information can still be a good data source to use, despite its negative reputation in information correctness and update frequency. The authors systematically examined all the IRR databases, and used them as the main data source their proposed attack detection system (**Neighborhood Watch**). The experiment revealed 0 to 3 flags per hour, infrequent enough for administrators to act on them when they show up. With careful processing, the information obtained from the IRR registries is enough for detection. The authors also observed that with more attention to improve the registry information, the security of BGP routing can be greatly enhanced with the existing registry based algorithms. As another observation, the authors also found that the reaction time for a large-scale event is as long as hours.

Schlamp et al. proposed to use IRR data and end-hosts information in the target IP blocks to detect subprefix hijacking attacks [25]. A subprefix hijacking is more complicated to detect since usually there is no BGP announcement for that specific prefix before. In this paper, the authors utilized IRR data to obtain prefix ownership information and construct AS-level topology. Both the ownership information and topology can be used to determine illegitimate sub-prefix path changes. To further confirm the sub-prefix hijackings, the authors compares the fingerprints of the machines within the sub-prefix obtained from different periods. An inconsistent comparison result shows the changes of the physical machines within the network. The authors assume that the web-servers' IP addresses are stable, and the SSL/TLS certificates could be used as the fingerprints to identify the servers. During the training period, system obtain the public keys of a set of HTTPS web servers reside in the target sub-prefix. When suspicious changes detected, the system will probe HTTPS servers within the prefix and compare the obtained certificates. An inconsistent comparison results indicates an ongoing prefix hijacking. The methodology proposed by the authors, especially the certificate comparison metric is novel in the attack detection field. However, running such a system requires a constant Internet-wide traffic scanning, which would be rather hard if not impossible in real-world cases. The web-server operators also conduct certificate rollovers from time to time for security reasons, which would result in false positive of the detection results.

Kruegel et al. [41] proposed to validate AS path updates based on topological and geographical information. The method classifies the ASes into "core" and "periphery" nodes based on their degree of connectivity in the AS topology. By removing the core nodes in the topology, the methods can generate a topology with

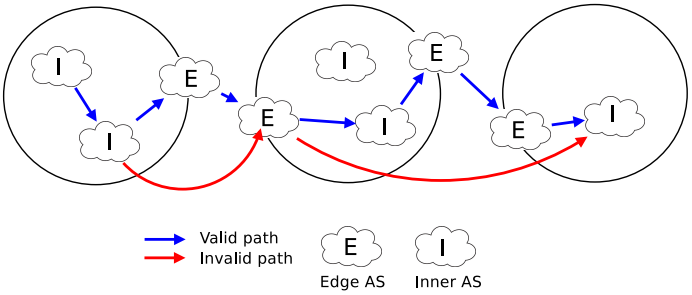


Fig. 7: An example of detecting BGP anomalous paths with topological and geographical information.

clusters of periphery nodes. The authors claimed that the *geographical distances* between ASes within a single cluster are small. The author proposed that a valid AS path must satisfy the following two requirements. First, the AS path may only contain one single subsequence of core ASes, conforming to the valley-free routing principle from Gao et al. [66]. Second, the periphery ASes must be either in one cluster or on the edge that connects two clusters. This means for an AS to reach another AS in a different cluster, this AS has to travel through a set of ASes that connects the clusters. The AS link jumps across multiple clusters would look suspicious, and thus will not be automatically accepted.

In [40] Sriram et al. proposed a comprehensive evaluation framework, **TERRAIN** (Testing and Evaluation of Routing Robustness in Assurable Inter-domain Networking), that compares the existing algorithms for BGP attack detection. TERRAIN investigates the performance of both active and passive monitoring attack detection methods. The authors proposed to enhance the active monitoring algorithms by validating the consistency of the data objects in the registry. They assigned the following four labels onto the registry entries based on the information consistency across data objects: fully consistent, only prefix information consistent, only prefix origination consistent, or not consistent. Based on each domain's security policy, the operators can then decide how to use the labeled information entries.

The authors also proposed an enhanced passive monitoring algorithm. The algorithm classifies prefixes into stable and unstable types based on the frequency of updates observed during training period. When suspicious changes to the prefixes are detected, the algorithm will decide the quarantine period length for the suspicious updates based on the type of the prefixes, assigning longer quarantine time to the changes for stable prefixes. To achieve the best accuracy, the authors propose to combine the enhanced passive and active monitoring approaches as a hybrid solution.

C. Passive Monitoring from the Data Plane

We have discussed plenty of mechanisms that utilize different types of control-plane information to detect anomalies of inter-domain routing. There are also a number of solutions that use only the data-plane information to detect inter-domain routing anomalies. One type of these data-plane-based solutions is the passive monitoring solutions. A passive monitoring solution utilizes data-plane information obtained through passive monitoring to detect routing anomalies, which does not need to send any query or probing packets during the procedure. In the following subsection, we survey the state-of-the-art passive monitoring solutions.

Liu et al. proposed a system [62] that detects prefix hijackings by traffic **load distribution changes (LDC)**. LDC monitors the data-plane traffic load distribution from any sources on the Internet to the target prefix and detect unusual load shifts. The load distribution is defined as the ratio between the total number of AS paths that pass through any specific node and the total number of AS paths available toward target prefix. The authors used a clustering algorithm to define the normal load patterns for each prefix at each AS. Any significant deviation of the traffic load on any AS will trigger alert to notify a potential prefix hijacking. LDC requires at least one monitor to be deployed at the provider of the target prefix to be able to detect anomalies of the prefix, which limits the deployability of this method.

Subramanian et al. designed the **Listen** algorithm [18] to detect routing anomalies using prefix reachability information obtained through passive monitoring from the data plane. It passively listens to all the TCP flows, and cherry-picks the segments that are specifically related to the establishment of a TCP connection. If Listen observes more than N incomplete TCP flows within any specific prefix, and within a predefined time period T, then it will alert the operators of the prefix (Fig. 8a). A flow is incomplete if it begins with a TCP SYN packet but never receives any SYN-ACK responses. To further reduce the amount of false positives, Listens applies two additional actions to verify the aliveness of any TCP flow. First, it can actively drop random packets from a specific TCP flow, and observe the retransmission of those dropped packets. If no retransmission is observed, it will raise alarm. Second, it can also passively sample the traffic for a specific flow and observe the ratio of the retransmission. If more than 50% of the packets are retransmitted, it will also raise alarm. The two extra steps prevent adversary from faking a TCP flow. However, this approach relies on a main assumption that the detection system is able to listen to the traffic of the monitored prefix. This assumption requires that the system has to be deployed at an AS (or ASes) where it carries a large portion of the traffic toward the monitored traffic. This requirement significantly hurts the deployability of this method.

Hiran et al. in [61] proposed a passive data-plane anomaly detection system that applies crowd-sourcing approach for information gathering. Each end-host that participate in the detection framework will collect round-trip time (RTT) information about all the IPs it interacts with. The information will be shared with other hosts to create a larger picture of the RTTs from and to different parts of the Internet. Using the aggregated RTT information, the system can visualize and detect suspicious RTT changes of any IPs over time. The problem with this method is that RTT is highly sensitive to any network changes and the nature of the detected anomalies could be non-malicious. The authors also acknowledged that this method should only be used as a supplementary evidence for the attack detection methods. Another problem is that the system requires participations from many end-hosts, which could be very hard to achieve. There is no clear motivation for end-hosts to participate, and the overhead would go up when the nodes become larger.

D. Active Monitoring from the Data Plane

Other than passively monitoring the traffic from the data plane, there are other BGP attack detection methods that apply active probing from the data plane to detect anomalies. Comparing to passive monitoring approaches, the active monitoring approaches only need to conduct active probing at a number of external vantage points, and do not need to have access to the traffic information of the monitored prefix.

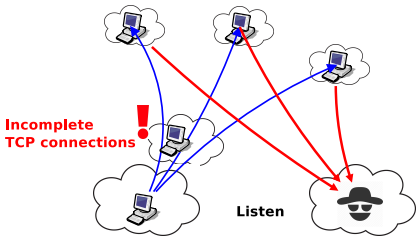
Zheng et al. proposed a distance-based attack detection system from the data plane [59]. The authors define the distance between two networks as the count of hops (or routers) a packet travels from one network to the other. The system detects the suspicious changes of the distances from a set of vantage points to the monitored prefix. For every prefix, the system selects a set of the best vantage points that are topologically dispersed from each other. The system continuously monitors the distance from every vantage point to the target prefix. When a suspicious network distance change is detected, the system will then further verify the detection results. During the training phase, the system selects a set of "reference points" of the target prefix, which are different prefixes that are topologically close to the target prefix. The data-plane path from the vantage points to the

reference points should be similar to the path toward the target prefix. Upon a potential hijacking is detected, the system probes the target prefix as well as the reference points of the prefix. If the paths to the prefix and to its reference points deviate significantly, then the system confirms the hijacking and alert the users. The authors suggest that the system can use the prefixes for the direct provider of the monitored prefix as reference points. However, an adversary can also hijack the reference points as well to avoid detection. The authors did not further investigate how the system should properly locate the reference points.

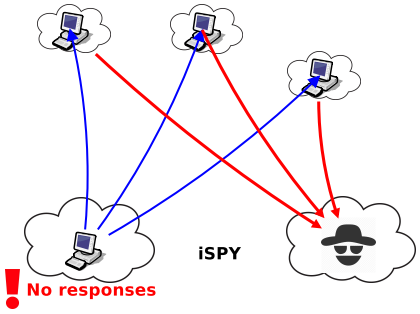
iSPY [60] is another data-plane based attack detection system proposed by Zhang et al. that detects the prefix hijackings that reduce the reachability of the victim prefixes. An iSPY system runs within the monitored prefix, and sends probing messages (such as ping messages) from the monitored prefix to a number of external networks. When there is no attack, iSPY expects to receive all the return messages of the probes it sends. During an prefix hijacking event, if a external is affected by the attack, the reply to the iSPY's probing message will arrive at the attacker's network, thus iSPY will not be able to receive it (Fig. reffig:bgp-detection-ispy). If the loss of return messages exceeds a threshold, iSPY will alert the prefix owner of a potential prefix hijacking.

However, there are three problems of this method. First, the unreachability from external network does not necessarily indicate prefix hijacking. There are many other types of events can also cause the reply of a iSPY probing cannot return, such as routing disruption, link failure, packet loss, etc. Second, to protect one prefix, the iSPY system for this prefix needs to send probes to about 3000 external networks, which leads to high network overhead. Moreover, iSPY fails to detect prefix interception attack, where the attacker hijacks the prefix and then continue to forward all the traffic to the victim. In this type of attack, the victim prefix does not lose reachability from any external network, and the iSPY probing packets will return as usual. As a result, iSPY will not be able to detect such an attack.

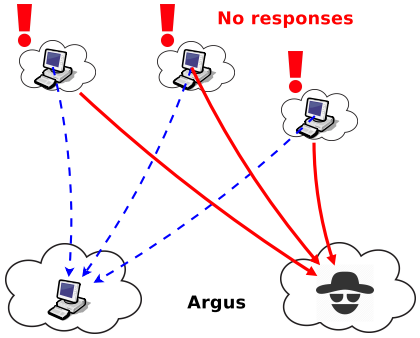
Xiang et al. proposed **Argus** [46], an active data-plane probing system that can verify detected routing anomalies. As discussed in section IV-A1, Argus applies a simple approach that alerts users of unseen origin or AS path segments for the monitored prefix. Argus then proposed a data-plane-based active probing system to verify the detection results (Fig. reffig:bgp-detection-argus). The verification system consists of multiple view points deployed across the Internet, and the view points are divided into the ones that see the "anomalous" updates, and the one that do not for specific event. Argus



(a) An example of Listen detecting loss of reachability through passive listening to TCP messages.



(b) An example of iSPY detecting loss of reachability from within the victim AS.



(c) An example of Argus detecting loss of reachability from external ASes.

Fig. 8: Example of the prefix hijacking detection method for Listen, iSPY, and Argus.

sends out ping messages to the monitored prefix from all the view points, collects the probing replies from both types of view points, and gathers the results into two vectors. Argus then calculates the correlation coefficient between the two vectors, the value of which ranges from -1 and 1. If the result is close to 1, it means most view points that see the updates cannot successfully ping the prefix, while the ones that do not see the updates can still get ping responses back. This case shows a strong evidence of a blackhole prefix hijacking. On the contrary, if the result is close to -1, it means most view points that do not see the update cannot get the response back. This indicates that the updates were

most likely related to route migration, and should not be malicious. The verification mechanism of Argus takes a big step on differentiating the legitimate changes from malicious ones. However, this could still only applies to the blackhole attacks where the attackers would drop the hijacked packets. In case of intercepting attack, where the attacker will eventually forward the hijacked traffic to the victim prefix, Argus cannot detect any changes in the probing results due to the fact that the reachability to the prefix remains the same.

Similarly, Hu et al. designed an active fingerprinting mechanism to verify attack detection results [58]. Upon the detection of a potential hijacking, the system will actively probe a set of end-hosts within the target prefix from multiple vantage points. The probing procedure will obtain the fingerprint information such as host OS properties, IP identifiers, TCP timestamps, ICMP timestamps, etc. The system then compares the fingerprints of the same set of end-hosts from different vantage points. If results between different vantage points are different, it is very likely that the prefix was hijacked. To obtain high accuracy, this system needs deploy a large number of vantage points from different ASes, avoiding the situation where all the vantage points are affected by the hijack. This requirement puts high deployment overhead on this system. The other problem of this method is that the probing messages may not get through to the target prefix from certain vantage points. Some ASes put restrictions on the probing messages and sometimes filter out such messages. This could also cause inconsistencies in the probing results, which does not necessarily indicate a prefix hijacking.

V. Security of Software-Defined Networking

Software-defined networking (SDN) is a recent paradigm for intra-domain networking. Traditionally, each domain (or AS) runs its own set of routing algorithms on top of a number of devices from different vendors with different implementation. The heterogeneous nature of the devices and their software complicates the management of the intra-domain networking. For example, a network operator has to communicate with routing devices using different vendor-specific interfaces. To unify the control of networking devices, SDN separates the control plane and the data plane. The SDN controller processes control-plane messages and makes routing decisions based on local routing policies, and the switches forward the traffic based on the forwarding rules decided by the controller. The separation simplifies the design of the data-plane devices, i.e., the switches, and provides more flexibilities for control-plane policies.

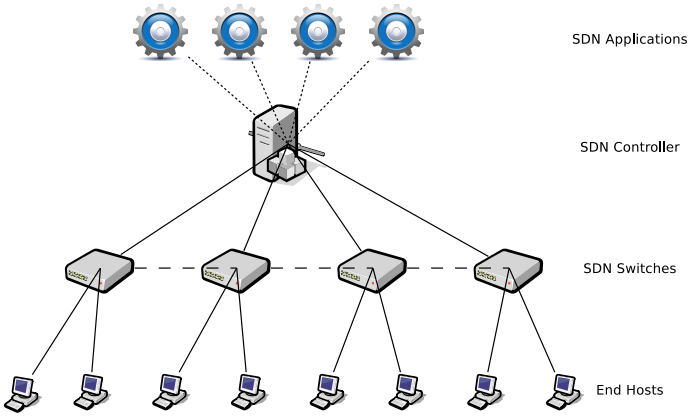


Fig. 9: Software-defined networking basic architecture.

There are several proposed protocols to unify the communication between a controller and switches, such as OpenFlow[7], OVSDB[67], ForCES[68], and POF[69]. These protocols define the messages between a controller and switches and provide implementation specifications for SDN switch manufacturers. Among these protocols, OpenFlow [7] is the most popular and widely deployed. We will use OpenFlow as the default SDN control-plane protocol in the following sections.

There are four types of components in an SDN environment (Fig.9): end-hosts, switches, controllers, and applications running on the controllers. The **end-hosts** are the machines and servers connected to the SDN switches. The **SDN switches** forward the traffic to and from their connected end-hosts. Every SDN switch is connected to one SDN controller, from which the switch can learn the forwarding rules for each traffic flow. When an incoming flow does not match any forwarding rule of a switch, the switch will send a query to its controller and ask for the actions on this flow. A **SDN controller** is a centralized network management entity that runs the SDN applications and communicates with the switches. When receiving a flow query from a switch, a controller will respond to the switch with one or more forwarding rules that match the flow and contain a set of actions on the flow, such as dropping the packets or forwarding the traffic to a specific port on the switch. The responses are decided by the SDN applications running on the controller. **SDN applications** are the control-plane software that contain all the routing policies and generate forwarding rules for the switches of an SDN network.

Unfortunately, all the four types of components could be compromised and used for launching attacks. The SDN applications may encompass security loopholes or malicious exploits from outside the network; the controller could be compromised by malicious operators to degrade the performance of the network; the switches

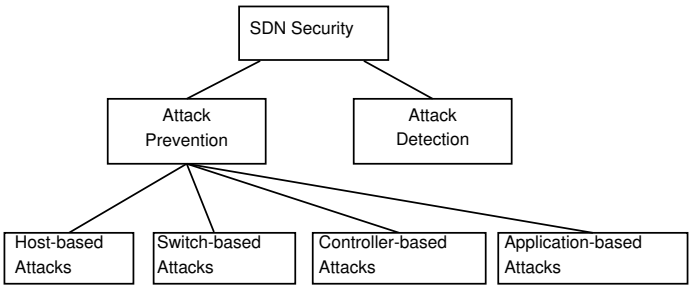


Fig. 10: Security solutions taxonomy for software-defined networking.

can be compromised to act maliciously when forwarding traffic; and malicious end-hosts can also exploit the loopholes in OpenFlow and disrupt the SDN controllers. The majority of the efforts on securing SDN focus on discovering security loopholes and designing mechanisms to prevent them from happening. We can categorize the security mechanisms for SDN by the types of the attacks to prevent, i.e., the solutions against end-host-based, switch-based, controller-based, or application-based attacks. In the following subsections, we discuss the threats that come from each type of the components, and survey the existing solutions toward the threats.

A. Securing SDN against Malicious Controllers

An SDN controller is the brain of an SDN network and hosts the applications and communicates with the switches. Whereas people focused more on the applications that run on the controller, the controller itself could also be compromised. A malicious controller could essentially take over the control of the entire network, thus requiring serious attention from the research and industry communities.

Matsumoto et al. described a system called **Fleet** that defends SDN networks against compromised controllers [70]. The authors define the compromised controllers as the ones that are controlled by malicious administrators. The authors state that malicious administrators can apply erroneous configurations on the controllers, the misconfigurations can then degrade the performance of a network or even disrupt the normal operation of the network. To defend against this attack, the authors assume that there are multiple controllers in the SDN network, and in order for a controller to install a new rule on an SDN switch, it needs to obtain acknowledgements from the other controllers. Fleet uses Shamir's secret sharing scheme [71] to ensure that at least k out of n controllers need to validate the rule (Fig. 11). An SDN switch in this system needs to verify the rule before installing it. On the other hand, an

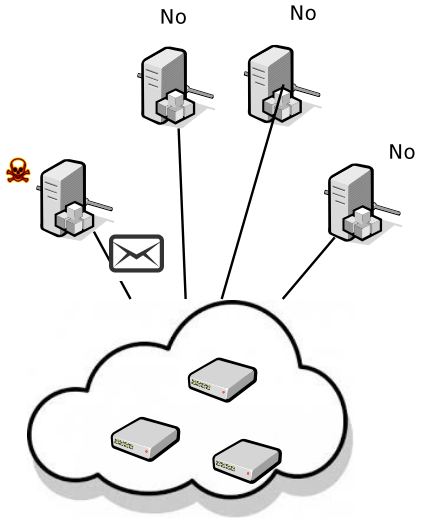


Fig. 11: An example of Fleet preventing malicious controller from sending out SDN rules.

attacker needs to compromise at least k controllers to modify forwarding rules in an SDN network. However, the requirement of multiple controllers in the SDN network makes Fleet impractical in many networks. There are many SDN controller software that only support single controller setup for a network, such as NOX [72], POX [73], Floodlight [74], or Ryu [75]. The authors also assume that all the controllers share the same set of configurations and policies, which makes the multiple-controller setup less cost-effective.

B. Securing SDN against Malicious Applications

The threats can also come from the applications running on the controllers. There are increasingly more applications designed and shipped to various SDN controllers. However, the qualities and the management of the applications are still in question. Different applications may generate the rules that overlap with each other or even with conflicting forwarding actions. Similar to software security, it is not easy to implement and enforce strong security policies during the design of SDN applications. Thus, the controller or third-party security auditors needs to monitor the operations of the SDN applications to ensure that these applications do not produce conflicting forwarding rules.

Canini et al. proposed a tool called **NICE** to uncover bugs in OpenFlow programs through modeling checking and symbolic execution [76]. The authors showed that an OpenFlow application that works correctly most of the time can misbehave under certain states. To uncover bugs in OpenFlow applications, NICE applies modeling checking [77] to explore system execution paths, and

apply symbolic execution to reduce the space of inputs. Through model checking, NICE is able to locate erroneous states and find out the root causes. Combining with symbolic execution, the computational overhead of exploring all possible states is largely reduced. Based on the exploration results, NICE further provides a set of APIs for users to express their desired correctness properties of the programs. Verifying the states with the correctness properties, NICE can produce verification results for any given OpenFlow application. Similarly, **ConfigChecker** [78] and **FlowCheker** [79] encode OpenFlow flow tables into binary decision diagram (BDD) and use model checking to verify the security properties. They convert the forwarding rules on switches into boolean expressions to build the binary decision diagram. With the constructed BDD, the system runs queries using computation tree logic to detect conflicts among rules.

Porras et al. proposed **FortNOX**, a security enforcement kernel for the NOX controller [72], to proactively detect application conflicts [80]. Instead of looking at each flow rule individually, FortNOX converts the similar rules into *alias set* according to their inter-dependency. The FortNOX then detects conflicting alias sets rather than individual rules. With this level of abstraction, FortNOX is able to largely reduce the overhead for detecting conflicts, and is able to detect conflicts for 1000 rules in less than 10 ms. To resolve the conflicts, FortNOX prioritizes different applications with the concept of *authorization level*. The rules generated by applications with a higher authorization level will be applied when there is a conflict.

FlowVisor [81] provides a way to split a physical network into multiple virtual networks, each with a dedicated controller. By splitting the network into smaller *slices*, each controller may only control a subset of devices based on the topology design. Each slice of the network is a logically independent virtual network, and operators utilize any previously mentioned security mechanisms to further secure this virtual network. The slicing of networks provides extra scalability and security features for the SDN controllers and applications.

FlowGuard [82] is a firewall framework for SDN networks that can detect and resolve firewall policy violations in real time. The authors categorize the security policy violations into two types: entire violation and partial violation. If all possible flows covered by a flow rule were against a firewall policy, the flow rule is an entire violation; otherwise it is a partial violation. Based on different types of violation, FlowGuard defined multiple methods, as opposed to a simple pass/drop decision, including dependency breaking for resolving legitimate but overlapping rules; update rejecting for resolving entire violations rules; and packet blocking for resolving partial violation rules. Unlike FortNOX's proactive verification approach, FlowGuard is able to verify new flows in real time. Evaluation showed that FlowGuard can check tens of thousands of new flows in milliseconds.

NetPlumber [83] detects policy violations from a graph perspective. It applies header space analysis (HSA) to construct rule dependency graph called "plumbing graph." In this graph, a node represents a forwarding rule, and an edge represents the dependency between the rules. NetPlumber maintains an up-to-date dependency graph and uses it to determine security policy violations. NetPlumber incrementally updates the graph on each relevant change of the network, such as adding a new rule, deleting rules, link status changes, or modification of forwarding tables. The incremental updating scheme enables NetPlumber to conduct violation detection in real time. NetPlumber also includes a high-level descriptive language for users to express their security policies. NetPlumber translates the policies from the descriptive language into forwarding rules for deployment. Overall, NetPlumber extends the original HSA method, and enables real-time detection and descriptive languages for expressing policies.

VeriFlow [84] verifies the OpenFlow rules against the security policies using a graph approach. VeriFlow summarize the forwarding flows into *equivalence classes* which represent sets of flows with identical forwarding actions. Utilizing the equivalence classes, VeriFlow can effectively confine the search space for conflicting rules. VeriFlow generates a forwarding graph for each equivalence class, showing how packets matched by this class will be forwarded throughout the network. Based on the forwarding graphs, VeriFlow further provides interfaces for the users to query about the reachability, loop-freeness, rule consistencies of the network. The content of the queries depends on the specific applications or tasks. Similar to NetPlumber, VeriFlow also adopts the incremental updating mechanism for the graphs to enable real-time verifications. The authors assume that the existing rules are benign during the equivalence classes building phase. The policy violations could thus stay undetected if they exist before VeriFlow starts running.

C. Securing SDN against Malicious Switches

SDN switches are responsible for forwarding all the traffic in an SDN network. It is thus important to make sure that the switches operate correctly and consistently. However, due to the differences in implementations,

SDN switches from different vendors may behave differently on the same input. Switches can also be compromised and conduct various malicious actions.

Kuzniar et al. presented a testing framework for SDN switches called **SOFT** that can detect inconsistencies between different OpenFlow implementations [85]. The authors proposed to use symbolic execution [86], [87], [88], [89] to explore input spaces systematically across multiple OpenFlow switches. SOFT can construct sequences of test inputs that cover all possible executions for each OpenFlow switch. The system will then compare the running results from different agents to identify inconsistencies. This paper is the first to present a systematical and comprehensive method for verifying SDN switch software.

Chi et al. proposed a system to detect compromised switches [90]. The authors first defined the following four types of misbehaviors of a compromised switch:

- *incorrect forwarding* that forwards traffic flows to incorrect ports;
- *duplicate forwarding* that duplicates traffic flows to multiple ports;
- *packet manipulating* that modifies content of the packets;
- *traffic weight adjusting* that alters traffic priorities.

This paper introduced two algorithms that detect forwarding anomalies and weight adjusting anomalies. The two algorithms use the same "active probing" idea. The controller constructs special flow rules (or "honey rules"), installs them on SDN switches, and sends out probing packets that match the rules. The switch can then detect the switches on the forwarding path that behave differently than the expected correct forwarding behavior. The system can apply these two algorithms to detect incorrect forwarding and weight adjusting. However, the duplicate forwarding and packet manipulation were not discussed in the paper.

D. Securing SDN against Malicious End-hosts

In software-defined networking environment, the end-hosts could also be malicious. The hosts can either exploit the switch-controller communication mechanism to flood the controller, or forge control-plane messages to deceive the controllers and influence the network topology. In this subsection, we will look at both angles and review the related works.

1) Host-based Flooding Attack: Shin et al. introduced the **AVANT-GUARD** system that protects the SDN controller from end-host-based control-plane saturation attack [91]. As previously discussed, the SDN switches

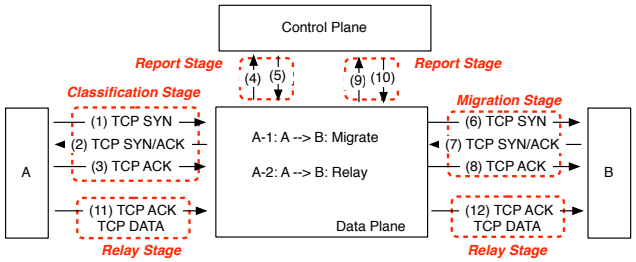


Fig. 12: Example connection migration scenario of AVANT-GUARD (figure from [91])

send any unmatched packets to the controller for handling. The control-plane saturation attack exploits this mechanism to generate a large amount of query messages using TCP SYN flood mechanism. First, an attacker floods the SDN switches with unique and unmatched TCP SYN packets. Each packet will trigger the switch to generate a query message to the controller. As a result, a large volume of query messages will arrive at and eventually overload the controller. To stop the flood, AVANT-GUARD acts as a proxy between switch and the controller, and reply the TCP SYN packets on behalf of the destination end-hosts. Only when the TCP connection is fully established (i.e., the sender and receiver both acknowledged the connection) will AVANT-GUARD forward the query to the controller and then migrate the connection to the true destination machines. Fig. 12 shows an example of this procedure where machine A tries to initiate a TCP connection with machine B. AVANT-GUARD first establishes the connection with machine A and then migrate the connection to B when the TCP handshakes are complete.

However, AVANT-GUARD suffers from the following problems. First, the proxy-style behavior requires a switch to have a large memory space to cache the connection status for every flow before the TCP connection is complete. The overhead of caching connections under flooding attack could be prohibitive. Second, the solution only deals with the TCP SYN flood attacks and the other types of attack are still not handled.

Wang et al. introduced **FloodGuard** to solve the control-plane saturation attacks in a proactive manor [92]. The authors proposed a static program analyzer to proactively generate flow rules to ensure the major functionality of the network work. If attack traffic manages to bypass the proactive rules, the packets will again trigger control-plane message floods. In this case, FloodGuard will generate rules to forward packets to a data-plane cache server and limit the rate of PACKET_IN messages. The FloodGuard provides

a more generic protection against various attacks with the proactive rule generation. However, the rate-limiting approach of processing traffic will significantly slow down the overall traffic speed, and trigger high memory and storage overhead to cache the data-plane flooding traffic.

2) Host-based Control Message Forgery Attack: Hong et al. in [93] presented two types of new network topology poisoning attacks, and proposed a system, **TopoGuard**, to secure the SDN control plane against such attacks. The first type of attack is *host-location hijacking attack*, where the hijacker forges packets to deceive the controller to believe that the victim has relocated to the hijacker's location, and thus hijacks all the traffic toward the victim. Specifically, the controller uses Host Tracking Service (HTS) to keep track of the hosts in the network, and identifies the hosts using a set of known identifiers (MAC, IP, VLAN ID, etc.). An attacker can forge packets with the same identifiers of the victim, and convince the HTS that the host is now in a new location. To stop such attacks, TopoGuard forces the controller to verify the change of location from the connected switch. For a change of location to be valid, the controller must first receive a Port_Down packet from the switch previously connected to the host. TopoGuard also actively probes the host's previous location to ensure that the original location is indeed not being used.

The second type of attack is *host-based link fabrication attack*. The controller dynamically discovers the topology of the network using LLDP (Link Layer Discovery Protocol) packets sent from the switches. The attacker exploits two facts here to conduct a link fabrication attack. First, the switch will forward the whole packet to controller as a query message when a new flow does not match any rule. Second, the controller does not have authentication for accepting the LLDP packets. The attacker then could forge a LLDP packets with fake topology information and send it to the switch. With a carefully designed packet header, the forged packets will be forwarded to the controller, and the controller will accept the LLDP packets. TopoGuard proposes to stop this type of attack by dropping all the LLDP packets from end-hosts. The end-hosts can be identified by the traffic types. However, this method cannot handle the cases where the attacker stays "quiet" before the attack, in which case the TopoGuard does not know whether the port is connected to a host or a switch. The assumption of the lack of authentication for LLDP packets also does not hold for many modern controller software (e.g., OpenDaylight [94]).

Dhawan et al. proposed **SPHINX** that also tries to secure the topological information on the controller [95]. SPHINX is an attack detection application that sits between the SDN controller and SDN switches. It aims to construct a trust-worthy topology graph, or "flow graph", by using only messages from the controller, and detect anomalous messages from the SDN switches. The authors believe that only the OpenFlow messages sent from the controller are trustworthy. Thus, SPHINX constructs the flow graph using only the "FLOW_MOD" messages sent from the controller. Any changes caused by untrusted entities, or changes that violate administrative policies will trigger alerts to the users.

VI. Security using Software-Defined Networking

Via the separation of control-plane and data-plane, SDN provides flexibilities for the operators or applications to fully control the packet forwarding in a centralized way. This new routing management paradigm enables a set of new security applications using SDN. In particular, there are two major directions where using SDN can help the existing security practices: traffic monitoring and traffic filtering. SDN enables easier traffic monitoring over the entire network, and can improve the effectiveness of the current *traffic anomaly detection* systems. With a unified control-plane protocol, SDN also boosts traffic filtering capability within one AS or across multiple ASes, which is specifically beneficial in the defense against *distributed denial-of-service* (DDoS) attacks. In this section, we investigate security mechanisms that use SDN on both traffic anomaly detection and DDoS defense.

A. SDN for Anomaly Detection

The basic idea for software-defined networking, or programmable networking, has been discussed long before the development of OpenFlow. Casado et al. proposed a centralized traffic management architecture called Ethane [96] that can conduct fine-grained flow level policy enforcement. Similar to the current SDN architecture, Ethane's architecture consists of end-hosts, Ethane switches, and an Ethane controller (Fig. 13). The controller accepts the policy inputs from the operators and translates high-level policies to switch-level rules and enforces the rules at the Ethane switches. To initiate a flow from within an Ethane network, a user needs to first authenticate itself to the controller. On detecting a new flow, an Ethane switch asks the controller to decide the forwarding actions, then installs a flow rule based on the reply from the controller. Despite its innovative architecture, Ethane is limited to only addressing the

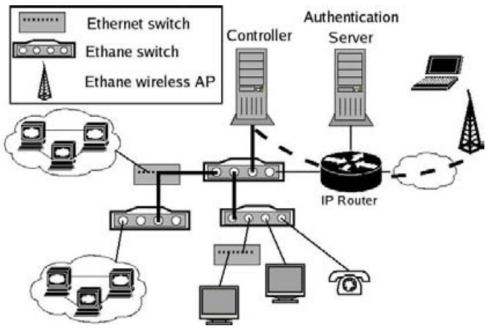


Fig. 13: Architecture of Ethane (figure from [96])

problems of user authentication and forwarding rule generation.

Mehdi et al. studied the feasibility of porting the existing anomaly detection systems into SDN environment [97]. Traditionally, the anomaly detection systems (ADS) are designed to run on dedicated machines and have no direct influence on networking. As the networking devices become more intelligent, the authors proposed to study whether the traditional ADS could be ported onto the SDN environment. In this work, the authors mainly focused on comparing the performance of deploying SDN-based ADS at different locations: home, small office home office (SOHO), or ISP. Specifically, they studied four different popular algorithms: threshold random walk with credit based rate limiting [98], regular rate limiting [99], [100], maximum entropy detector [101], and NETAD [102]. The authors first showed the feasibility of running all four algorithms in an SDN environment. The authors then collected data from real networks and compared the performance of these algorithms at different locations. The evaluations showed that deploying the solutions at home environment would have the highest detection rate and the fastest processing speed. However, this work does not compare the performance with the same solutions deployed on the traditional network environments, thus weakening the motivation of this work. This work also did not take into consideration the deployment cost at home environment.

To enable SDN for more general network security applications, Shin et al. proposed **FRESCO** [103], a programming framework for security applications in SDN environment. It provides a set of reusable modules as a library and enables developers to create security functions quickly and easily. The security functions include firewalls, scan detectors, intrusion detection systems, etc. FRESCO also provides APIs to enable legacy applications to trigger the security modules. This capability allows FRESCO to work with existing security applications such as deep packet inspection (DPI) boxes with minimum changes of the code. The authors demonstrated several applications generated using FRESCO and showed that FRESCO introduces reasonable overhead. However, FRESCO itself does not directly address any existing security problems.

B. SDN for DDoS Defense

The distributed denial-of-service (DDoS) attacks have been troubling the Internet for more than a decade now. With the fast development of SDN technology, researchers started to develop DDoS defense solutions for the SDN environments. Compared to using the traditional networking scheme, using SDN for DDoS defense has many benefits. First, SDN allows operators to describe the DDoS defense logics in a standard way using OpenFlow [7] with no vendor-specific interfaces required. Using unified communication protocol allows operators to deploy DDoS defense rules on any OpenFlow enabled devices with little compatibility concern. Second, the centralized control of SDN architecture enables operators to quickly modify the network topologies and forwarding rules of the entire network with minimum efforts. Third, multiple DDoS defense solutions can easily work together with the help of SDN. For example, SDN can enforce traffic flow to go through specific paths or equipments, and enable different defense components to process the traffic in a specific order. For example, a solution could require the traffic first flow through a generic traffic filter, then a deep-packet-inspection box, then a rate-limiting cache, and finally the destination. Overall, the SDN technology allows network operators to have much more flexibility without sacrificing the forwarding speed. In this section, we look at several DDoS defense solutions that utilize SDN as their main networking architecture.

Giotis et al. introduced a system that utilizes SDN to enhance the traditional DDoS defense solutions [104]. Traditionally, the Remote Triggered Black-Hole (RTBH) [105] method is used to automatically propagate blacklist flow rules for stopping unwanted traffic. RTBH uses BGP to exchange filtering information, and alters the next hop address of a matched prefix to achieve the black-holing goal. Because RTBH relies on BGP, it can only filter traffic on per-prefix level. RTBH is also limited to only dropping the matched traffic. To address the problems and utilizing the capability of SDN network, the authors proposed to modify the RTBH behavior, forwarding the matched traffic to a SDN network instead of dropping, and letting the SDN network to further process the traffic. By doing so, the

system separates the traffic steering and traffic filtering procedures, and enables the usages of SDN without largely changing the existing protocols. There are three components in the system: an anomaly detection system that triggers RTBH function, the RTBH function that forwards the traffic to a SDN network, and the mitigation functions used in the SDN network for filtering unwanted traffic. The system differentiate DDoS traffic from benign traffic using a bidirectional count sketch algorithm. The algorithm counts the packets and detects the highly asymmetric communication patterns, which the DDoS traffic usually has. Overall, the system is able to utilize SDN for filtering unwanted traffic with the help of BGP RTBH.

Lim et al. presented a system that can mitigate the DDoS attack and protect the desired services through resource migration [106]. When a DDoS is detected, the system provides new IPs for the servers under protection and redirects legitimate traffic to the new addresses. The system applies CAPTCHA [107] (or RECAPTCHA) to prevent the bots from knowing the new IPs. In the mean time, the system measures the frequency of accessing the protected resources for each client, and marks the clients with high access frequency as bots. All the traffic from the clients marked as bots will be dropped. However, there are several problems that this solution did not address. First, the system can only work with HTTP server scenario, where the redirection can be done automatically with explicit HTTP redirecting command. For other types of applications, such as FTP or SSH, the redirection of traffic will require modification of code on all clients. Second, the CAPTCHA methods cannot effectively stop bots from knowing the new addresses of the protected server. It requires only one manual access to learn the new addresses, and the attackers can quickly reconfigure the bots to attack the new addresses.

To provide a more proactive defense solution against DDoS attack, Jafarian et al. proposed a moving target technique called **OpenFlow Random Host Mutation (OF-RHM)** [108]. OF-RHM randomly and frequently mutates the IP addresses of the end-hosts to prevent attackers from learning the IP assignments of the network. To enable the transparent mutation, OF-RHM system keeps the original IP of the end-hosts and associates each host with a random *virtual IP*. When a flow enters the network, OF-RHM translates the destination IP from the real IP to the virtual IP. When a flow leaves the network, OF-RHM also translates the source IP from the virtual IP to the real IP. Moreover, the system ensures high mutation rate and randomness on virtual IP assignments. However, this system implicitly assumes that the network has a large range of unused IP addresses to use, which could be impractical in real-world cases, especially under the IPv4 address depletion situation.

VII. Conclusion

In this report, we examined the security problems for the inter-domain and intra-domain routing protocols, and surveyed the existing solutions. Specifically, we focused on Border Gateway Protocol (BGP) for inter-domain routing, and software-defined networking (SDN) for intra-domain routing. We investigate both BGP and SDN security solutions, with a focus on their strengths and weaknesses as well as their deployment status on the Internet.

We categorized the security solutions into two general categories: the attack prevention solutions and the attack detection solutions. The attack prevention solutions aim to proactively stop potential attacks through security upgrades of the protocol design or operations. Meanwhile, the attack detection solutions aim to reactively detect abnormal events regarding the operation of the protocols in order to trigger timely reaction to such events. The attack prevention mechanisms for BGP have been heavily studied [8], [9], [10], [11]. To date, none of these mechanisms has been largely deployed to date, leaving the Internet still vulnerable to the inter-domain routing attacks. We focused on examining the attack detection mechanisms for BGP, which most of the recent BGP security work have concentrated on. On the other hand, since SDN is still young, the majority of the SDN security work look at attack prevention, with less on attack detection.

For inter-domain routing (i.e., BGP), we surveyed attack preventions solutions of three types: the control-plane-based cryptographic approaches, the control-plane-based non-cryptographic approaches, and the data-plane-based approaches. However, the majority of the these solutions have not been adopted on the Internet, making the accurate detection of attacks more important. For existing attack detection mechanisms for BGP, we categorize them by their main methods: some detection systems use active probing to acquire information, while the others rely on passive monitoring. Based on the information source, the attack detection systems can be also categorized into control-plane-based, data-plane-based, or hybrid mechanisms. Compared to the attack prevention mechanisms, the attack detection systems usually incur much less overhead and easier to deploy, making them more preferable in real-world deployment scenarios.

For intra-domain routing (i.e., SDN), we mainly examined the security mechanisms from two perspectives: the

methods that secure the SDN itself, and the applications of SDN on solving other network security problems. Based on the types of attacks, we investigated the security solutions that protect SDN against attacks originated from SDN controllers, applications, switches and end-hosts. We also reviewed how researchers have utilized SDN to defend against other network security problems. However, because SDN is a new development in intra-domain routing, the overall security of SDN or using SDN is still at its early stage and many works remain to be done. Especially, while there are plenty of work on anomaly detection of BGP, little has been investigated on SDN anomaly detection.

ACKNOWLEDGEMENT

I would like to thank my research advisor Professor Jun Li for his expert advice and encouragement throughout the whole research, as well as Professor Reza Rejaie and Professor Hank Childs for their insightful guidance on the writing the report.

This material is based upon work supported by the USA National Science Foundation under Grant No. CNS-1118101. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," Tech. Rep., 2005.

[2] G. S. Malkin, "RIP version 2," 1998.

[3] J. Moy, "OSPF version 2," 1997.

[4] D. Oran, "OSI IS-IS intra-domain routing protocol," 1990.

[5] R. Albrightson, J. Garcia-Luna-Aceves, and J. Boyle, "EIGRP–A fast routing protocol based on distance vectors," *Interop 94*, 1994.

[6] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[8] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security issues and solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, Jan. 2010.

[9] G. Huston and R. Bush, "Securing BGP with BGPSec," *Internet Protocol Journal*, vol. 14, no. 2, pp. 2–13, 2011.

[10] S. Goldberg, "Why is it taking so long to secure internet routing," *Education for Health*, vol. 15, no. 3, pp. 291–293, 2002.

[11] H. Shiravi, A. Shiravi, and A. a. Ghorbani, "A survey of visualization systems for network security," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1313–1329, Aug. 2012.

[12] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 582–592, 2000.

[13] R. White and R. White, "Deployment Considerations for secure origin BGP (soBGP)," 2003.

[14] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. Mcdaniel, and A. Rubin, "Working Around BGP : An Incremental Approach to Improving Security and Accuracy of Interdomain Routing," *The Annual Network and Distributed Systems Security Symposium (NDSS)*, 2003.

[15] Y.-c. Hu and M. Sirbu, "SPV : Secure Path Vector Routing for Securing BGP," *ACM SIGCOMM*, pp. 179–192, 2004.

[16] T. Wan, E. Kranakis, and P. Van Oorschot, "Pretty Secure BGP (psBGP)," *Annual Network and Distribution Systems Security Symposium (NDSS)*, no. February, pp. 1–23, 2005.

[17] M. Lepinski, "Bgpsec protocol specification," 2015.

[18] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz, "Listen and whisper: Security mechanisms for BGP," 2004.

[19] E.-y. Kim, K. Nahrstedt, L. Xiao, and K. Park, "Identity-based Registry For Secure Interdomain Routing," *ACM Conference on Computer and Communications Security (CCS)*, pp. 321–331, 2006.

[20] V. Gill, J. Heasley, and D. Meyer, "The Generalized TTL Security Mechanism (GTSM)," *Request For Comments (RFC)*, no. 3682, pp. 1–16, 2004.

[21] A. Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option," *Request For Comments (RFC)*, no. 2385, 1998.

[22] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," no. 2401, pp. 1–101, 1998.

[23] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra, "Routing Policy Specification Language (RPSL)," Tech. Rep., 1999.

[24] L. Blunk, J. Damas, F. Parent, and A. Robachevsky, "Routing Policy Specification Language next generation (RPSLng)," Tech. Rep., 2005.

[25] J. Schlamp, R. Holz, O. Gasser, A. Korsten, Q. Jacquemart, G. Carle, and E. Biersack, "Investigating the Nature of Routing Anomalies: Closing in on Subprefix Hijacking Attacks," in *Traffic Monitoring and Analysis*, ser. Lecture Notes in Computer Science, M. Steiner, P. Barlet-Ros, and O. Bonaventure, Eds. Springer International Publishing, 2015, vol. 9053, pp. 173–187.

[26] P. Vervier, O. Thonnard, and M. Dacier, "Mind your blocks: On the stealthiness of malicious BGP hijacks," no. February, pp. 8–11, 2015.

[27] G. Siganos and M. Faloutsos, "Analyzing BGP policies: methodology and tool," *Proceedings of IEEE INFOCOM*, 2004.

[28] ——, "A Blueprint for Improving the Robustness of Internet Routing," 2005. [Online]. Available: http://www.cs.ucr.edu/

[29] ——, "Neighborhood watch for Internet routing: Can we improve the robustness of Internet routing today?" *Proceedings of IEEE INFOCOM*, pp. 1271–1279, 2007.

[30] M. Wählisch, O. Maennel, and T. C. Schmidt, "Towards detecting BGP route hijacking using the RPKI," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, p. 103, 2012.

[31] M. Wählisch, F. Holler, T. C. Schmidt, and J. H. Schiller, "Updates from the Internet Backbone:An RPKI/TRT Router Implementation, Measurements, and Analysis," *Network and Distributed System Security Symposium (NDSS)*, pp. 1–11.

[32] E. Osterweil, T. Manderson, R. White, and D. McPherson, "Sizing estimates for a fully deployed RPKI," Verisign Labs, Tech. Rep., 2012.

[33] D. Cooper, E. Heilman, K. Brogle, L. Reyzin, and S. Goldberg, "On the risk of misbehaving RPKI authorities," *Proceedings of ACM Workshop on Hot Topics in Networks (HotNets)*, pp. 1–7, 2013.

[34] E. Heilman and S. Goldberg, "From the Consent of the Routed : Improving the Transparency of the RPKI," *Sigcomm 2014*, pp. 51–62, 2014.

[35] R. Lychev, S. Goldberg, and M. Schapira, "BGP Security in Partial Deployment Is the Juice Worth the Squeeze?" *ACM SIGCOMM*, pp. 171–182, 2013.

[36] Q. Li, Y.-C. Hu, and X. Zhang, "Even rockets cannot make pigs fly sustainably: Can bgp be secured with bgpsec?" 2014.

[37] R. White, "Securing BGP through secure origin BGP (soBGP)," *Business Communications Review*, vol. 33, no. 5, pp. 47–53, 2003.

[38] B. Raghavan, S. Panjwani, and A. Mityagin, "Analysis of the SPV secure routing protocol," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, p. 29, 2007.

[39] "HP-UX IPSec vA.03.00 Performance and Sizing White Paper," http://h20565.www2.hpe.com/hpsc/doc/public/display?sp4ts.oid=4164811&docId=emr_na-c02035735, 2009, [Online].

[40] K. Sriram, O. Borchert, O. Kim, P. Gleichmann, and D. Montgomery, "A Comparative analysis of BGP anomaly detection and robustness algorithms," *Proceedings of Cybersecurity Applications and Technology Conference for Homeland Security (CATCH)*, pp. 25–38, 2009.

[41] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Topology-based detection of anamolous BGP messages," *Recent Advances in Intrusion Detection*, vol. 6, 2003.

[42] J. Li, T. Ehrenkranz, and P. Elliott, "Buddyguard: A buddy system for fast and reliable detection of ip prefix anomalies," in *IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2012, pp. 1–10.

[43] J. Li and S. Brooks, "I-seismograph: Observing and measuring internet earthquakes," in *Proceedings of IEEE INFOCOM*. IEEE, 2011, pp. 2624–2632.

[44] G. Theodoridis, O. Tsigkas, and D. Tzovaras, "A Novel Unsupervised Method for Securing BGP Against Routing Hijacks," vol. 62, pp. 353–358, 2010.

[45] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Detecting prefix hijackings in the internet with argus," *Proceedings of the 2012 ACM conference on Internet measurement conference (IMC)*, p. 15, 2012.

[46] Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Argus: An accurate and agile system to detecting IP prefix hijacking," *Proceedings of International Conference on Network Protocols (ICNP)*, pp. 43–48, 2011.

[47] J. Zhang, J. Rexford, and J. Feigenbaum, "Learning-based anomaly detection in BGP updates," in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM, 2005, pp. 219–220.

[48] L. Yuan, J. Mai, and C.-N. Chuah, "Bgp anomaly detection using wavelet analysis," University of California, Davis, Tech. Rep., 2004.

[49] B. A. Prakash, N. Valler, D. Andersen, M. Faloutsos, and C. Faloutsos, "BGP-lens: Patterns and Anomalies in Internet Routing Updates," *Kdd*, pp. 1315–1323, 2009.

[50] S. T. Teoh, S. Ranjan, A. Nucci, and C.-N. Chuah, "BGP eye: A new Visualization Tool for Real-time Detection and Analysis of BGP Anomalies," in *Proceedings of the International Symposium on Visualization for Computer Security (VizSec)*, 2006, p. 81.

[51] V. Khare, Q. Ju, and B. Zhang, "Concurrent prefix hijacks: Occurrence and impacts," in *Proceedings of the ACM conference on Internet measurement conference (IMC)*. ACM, 2012, pp. 29–36.

[52] N. M. Al-Rousan and L. Trajkovic, "Machine learning models for classification of BGP anomalies," *2012 IEEE 13th International Conference on High Performance Switching and Routing (HPSR)*, vol. 513, pp. 103–108, 2012.

[53] M. Lad, D. Massey, D. Pei, and Y. Wu, "PHAS: A prefix hijack alert system," *Proceedings of USENIX Security*, 2006.

[54] J. Karlin, S. Forrest, and J. Rexford, "Pretty good BGP: Improving BGP by cautiously adopting routes," *Proceedings - International Conference on Network Protocols (ICNP)*, pp. 290–299, 2006.

[55] N. Feamster, N. Feamster, H. Balakrishnan, and H. Balakrishnan, "Detecting BGP configuration faults with static analysis," *Proceedings of Networked Systems Design and Implementation (NSDI)*, pp. 49–56, 2005.

[56] J. Qiu, L. Gao, S. Ranjan, and A. Nucci, "Detecting bogus BGP route information: Going beyond prefix hijacking," *Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks (SecureComm)*, vol. 1, pp. 381–390, 2007.

[57] T. Wong, V. J. V. Jacobson, and C. Alaettinoglu, "Internet routing anomaly detection and visualization," *International Conference on Dependable Systems and Networks*, pp. 172–181, 2005.

[58] X. Hu and Z. M. Mao, "Accurate real-time identification of IP prefix hijacking," *Proceedings - IEEE Symposium on Security and Privacy*, no. 2, pp. 3–17, May 2007.

[59] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A light-weight distributed scheme for detecting ip prefix hijacks in real-time," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, p. 277, Oct. 2007.

[60] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "iSPY: Detecting IP Prefix Hijacking on My Own Zheng," *ACM SIGCOMM*, p. 327, 2008.

[61] R. Hiran, N. Carlsson, and N. Shahmehri, "Crowd-based Detection of Routing Anomalies on the Internet," *Computer and Network Security (CNS)*, 2015.

[62] Y. Liu, J. Su, and R. K. C. Chang, "LDC: Detecting BGP prefix hijacking by load distribution change," *Proceedings of IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 1197–1203, 2012.

[63] Z. Zhang, Y. Zhang, Y. C. Hu, and Z. M. Mao, "Practical defenses against BGP prefix hijacking," *Proceedings of the ACM CoNEXT Conference*, p. 1, 2007.

[64] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment workshop - IMW '02*, p. 197, 2002.

[65] D. Meyer *et al.*, "University of Oregon Route Views Project," 2005.

[66] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transactions on Networking (ToN)*, vol. 9, no. 6, pp. 733–745, 2001.

[67] B. Pfaff and B. Davie, "The open vSwitch database management protocol," 2013.

[68] E. Haleplidis, S. Denazis, O. Koufopavlou, D. Lopez, D. Joachimpillai, J. Martin, J. H. Salim, and K. Pentikousis, "ForCES applicability to SDN-enhanced NFV," in *European Workshop on Software Defined Networks (EWSDN)*. IEEE, 2014, pp. 43–48.

[69] H. Song, "Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane," in *Proceedings of the ACM SIGCOMM workshop on Hot topics in software-defined networking (HotSDN)*. ACM, 2013, pp. 127–132.

[70] S. Matsumoto, S. Hitz, and A. Perrig, "Fleet : Defending SDNs from Malicious Administrators," pp. 103–108, 2014.

[71] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[72] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[73] "POX," http://www.noxrepo.org/pox/about-pox/, [Online].

[74] "FloodLight," http://www.projectfloodlight.org/floodlight/, [Online].

[75] "Ryu," http://osrg.github.io/ryu/, [Online].

[76] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, "A NICE Way to Test OpenFlow Applications," *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[77] R. Jhala and R. Majumdar, "Software model checking," *ACM Computing Surveys (CSUR)*, vol. 41, no. 4, p. 21, 2009.

[78] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration Analysis and Verification of Federated OpenFlow Infrastructures," *Proceedings of the ACM workshop on Assurable and usable security configuration (SafeConfig)*, p. 37, 2010.

[79] E. Al-Shaer and M. N. Alsaleh, "ConfigChecker: A tool for comprehensive security configuration analytics," *Symposium on Configuration Analytics and Automation*, pp. 1–2, 2011.

[80] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," p. 121, 2012.

[81] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[82] H. Hu, W. Han, G. Ahn, and Z. Zhao, "FLOWGUARD: building robust firewalls for software-defined networks," *Proceedings of the ACM SIGCOMM workshop on Hot topics in software-defined networking (HotSDN)*, 2014.

[83] P. Kazemian, M. Change, and H. Zheng, "Real Time Network Policy Checking Using Header Space Analysis," *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 1–13, 2013.

[84] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "Veriflow: verifying network-wide invariants in real time," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 467–472, 2012.

[85] M. Kuzniar, P. Peresini, M. Canini, D. Venzano, and D. Kostic, "A SOFT way for OpenFlow switch interoperability testing," *Proceedings of the International conference on Emerging networking experiments and technologies*, pp. 265–276, 2012.

[86] J. C. King, "A new approach to program testing," in *ACM SIGPLAN Notices*, vol. 10, no. 6. ACM, 1975, pp. 228–233.

[87] S. Bucur, V. Ureche, C. Zamfir, and G. Candea, "Parallel symbolic execution for automated real-world software testing," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 183–198.

[88] C. Cadar, D. Dunbar, and D. R. Engler, "KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs," in *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, vol. 8, 2008, pp. 209–224.

[89] V. Chipounov, V. Kuznetsov, and G. Candea, "The S2E platform: Design, implementation, and applications," *ACM Transactions on Computer Systems (TOCS)*, vol. 30, no. 1, p. 2, 2012.

[90] P.-w. Chi, C.-t. Kuo, J.-w. Guo, and C.-l. Lei, "How to Detect a Compromised SDN Switch," *Proceedings of the IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–6, 2015.

[91] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 413–424.

[92] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks," *Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2015.

[93] S. Hong, X. Lei, H. Wang, and G. Gu, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures," pp. 8–11, 2015.

[94] "OpenDaylight," https://www.opendaylight.org/, [Online].

[95] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting security attacks in software-defined networks," in *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, 2015.

[96] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking Control of the Enterprise," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, p. 1, 2007.

[97] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting Traffic Anomaly Detection using Software Defined Networking," *Entropy*, vol. 6961, pp. 1–20, 2011.

[98] J. Jung, S. E. Schechter, and A. W. Berger, "Fast Detection of Scanning Worm Infections," *System*, pp. 1–14, 2004.

[99] M. M. Williamson, "Throttling Viruses: Restricting propagation to defeat malicious mobile code," in *Proceedings of the Computer Security Applications Conference*, no. HPL-2002-172, 2002, pp. 61–68.

[100] J. Twycross and M. M. Williamson, "Implementing and Testing a Virus Throttle," *Proceedings of the USENIX Security Symposium*, no. August, pp. 285–294, 2003.

[101] Y. Gu, A. McCallum, and D. Towsley, "Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation," in *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, 2005, p. 32.

[102] M. V. Mahoney, "Network traffic anomaly detection based on packet bytes," *Proceedings of the ACM symposium on Applied computing (SAC)*, p. 346, 2003.

[103] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks." vol. 2, no. February, 2013.

[104] K. Giotis, G. Androulidakis, and V. Maglaris, "Leveraging SDN for Efficient Anomaly Detection and Mitigation on Legacy Networks," *European Workshop on Software Defined Networks*, pp. 85–90, 2014.

[105] D. Turk, "Configuring BGP to block Denial-of-Service attacks," 2004.

[106] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," *International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 63–68, 2014.

[107] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Advances in CryptologyEUROCRYPT*. Springer, 2003, pp. 294–311.

[108] J. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," pp. 127–132, 2012.