Fast Address Hopping at the Switches: Securing Access for Packet Forwarding in SDN

Sang-Yoon Chang*
Advanced Digital Sciences Center
sychg@adsc.com.sg

Younghee Park*
San Jose State University
younghee.park@sjsu.edu

Akshaya Muralidharan
San Jose State University
aksh3001@gmail.com

***Abstract*—To defend against network reconnaissance for unauthorized access of the packet forwarding path, we leverage software-defined networking (SDN) and build moving target defense (MTD) by randomizing network addresses. We distinguish our work from prior research by implementing MTD at the data plane and on all nodes along the forwarding path. Thus, our scheme is fast and lightweight in operation (significantly decreasing the controller communication overhead) and enables quicker security response to reduce the attack impact (as opposed to having the attack impact all the way to the endhost destination). We validate our work on an Open vSwitch-based testbed and show that the attacker's cost to achieve timely network reconnaissance increases by more than an order of magnitude than having the controller actuate the MTD.**

I. Introduction

In communication network, networked nodes send data traffic to each other. Since not all nodes are directly connected by a communication link, the relaying nodes forward the traffic on behalf of the source and the destination, and since there can be multiple forwarding paths to reach the destination, routing protocols determine which forwarding path to use.

In contrast to building protection at the network perimeters, e.g., filtering, some network security researchers address security vulnerabilities after the perimeter is breached and assume attackers operating within the network. We also consider attackers who have the physical access to the wired link along the forwarding path and can inject malicious packets to the network nodes. To make such injection effective, the attacker needs to achieve *network reconnaissance*, which process investigates the network for vulnerability prior to launching an attack, e.g., denial-of-service (DoS).

To thwart network reconnaissance, moving target defense (MTD) varies the network parameters, e.g., the address, the medium access, and the network configuration. By randomization, MTD increases the unauthorized user's cost in probing and achieving network reconnaissance (while the authorized users who control and know the randomization pattern have a significant edge to determine the network parameters). We build MTD based on the randomization of the nodes's IP addresses, and thus knowing the IP address randomization acts as implicit authorization to access the communication.

Our work offers contributions beyond the state-of-the-art that adopts MTD for networking in the following ways: first, we build defense at both the router switches and the endhosts securing access of the routing path at the link level and enabling quicker response at the first hop of the unauthorized access (in contrast, prior work only protects the endhosts [1], [2]); second, we consider an advanced threat model where the attackers compromised the network and can intelligently sense and monitor the traffic from the breach point; and third, we spread the controller communication information by using it to generate a random pattern to actuate the MTD randomization on the data-plane routers, making the actuation quicker and the operation lightweight in control overhead. Our evaluation shows that the attacker cost to achieve network reconnaissance increases by more than an order of magnitude against our scheme than the state-of-the-art MTD randomization.

We build our scheme on *software-defined networking* (SDN) architecture, which de-couples the decision-making at the control plane and the lower-level implementation of forwarding at the data plane. SDN offers a centralized infrastructure where the controller resides in the control plane and supports communication between the control and the data plane, e.g., via OpenFlow, and is typically used for intra-networking applications (where the network is governed and operated by a network manager). SDN thus facilitates the synchronization between the nodes along the forwarding path and the distribution of the randomization seeds in our work.

The rest of the paper is organized as the following. We review the relevant prior work and highlight our contributions beyond the state of the art in Section II. Afterward, we describe the system and threat model in Section III and our scheme in Section IV. Section V analyzes our scheme, and Section VI validates its effectiveness using an Open vSwitch-based testbed prototype. Lastly, Section VII concludes our paper.

II. Related Work and Our Work

Our work is inspired by prior work in wireless security to secure link access. As wireless communication is inherently broadcast and anybody equipped with a radio can access the communication, researchers and military experts have long been working on securing access; in contrast, wired network security has traditionally placed heavier focus on defending the network perimeter, e.g., filtering, so that attackers do not have the link access within the system. In Section II-A, we discuss about the body of work in wireless security that inspired our

*Doctor Chang and Professor Park are the corresponding authors.

978-1-5090-0223-8/16/$31.00

work (spread spectrum), related work in (wired) networking that adopts randomization, and then other relevant work that implements security at the data-plane routers. In Section II-B, we highlight our novel contributions beyond these prior work.

A. Related Work

I) Spread spectrum In wireless medium access control, MTD is used for building link resiliency against jamming [3]–[5] and securing confidentiality against eavesdropping [6], [7]. Spread spectrum is a popular approach to realize MTD, and the access parameters are typically time, frequency, and code. For instance, frequency hopping spread spectrum (FHSS) in wireless communication dynamically varies the carrier frequency in order to prevent the attacker's access; fast hopping with short hopping duration thwarts reactive attackers who sense the channel use and adapt their strategy in real-time [8]. We emulate two properties of spread spectrum at the physical and link layers: first, the hopping is dynamic and quick (making it difficult for attackers to react to the protocol and adjust their parameters); second, the hopping is proactive and autonomous (as opposed to being reactive and triggered). These properties distinguish our work from prior work that deploys MTD at the network layer, which work we discuss later.

The aforementioned spread spectrum work assumes single-hop network, i.e., destination hosts are directly reachable by source hosts, and is typically applied to secure the last hop of communication link for mobile endhosts, e.g., which communicates with the access point. In contrast, to adopt MTD techniques is more challenging for communication that consists of multiple hops, in which case, the links across the hops must be synchronized, and the implementation spills beyond the source-destination pair (who are directly involved in the communication payoffs) and to the intermediate routers (who merely relay the packets).

II) Address space randomization We use the network address as our randomization parameter. Dynamic host configuration protocol (DHCP) is widely deployed for IP control but is not designed for security. In the security context, researchers have used dynamic addressing to prevent network reconnaissance [9]–[12]. An adaptation of network address translation (NAT) has also been proposed [13]. However, these prior work in address space randomization build security only at the endhosts (and not at the intermediate router switches) and thus the attackers can still access the path until the traffic reaches the endhosts. In contrast, we aim to prevent the attacker access early in the forwarding path.

III) Software-defined network We design our scheme on SDN. Since most intelligence is on the controller, e.g., routing control, many prior work proposes to offload the security implementation to the controller (e.g., detection and filtering based on flow analyses [14]–[16] and traffic analyses [17]–[19]) and build on secure communication between the router switches and the controller [20], [21]. In specific, Kampanakis et al. [2] sketches the adoption of the address space randomization technique on the SDN controller, and OF-RHM [1] describes an instantiation of such approach.

Our work is different from SDN-based MTD work in three aspects. First the security implementation scope is different, as these prior work in address randomization are implemented only at the endhosts and need to tolerate the attacker traffic until it reaches the destination endhost (much like the previously mentioned network address randomization work in non-SDN contexts in Section II-A-II). Second, they require substantial overhead as they require interacting with the controller whenever the network address changes; in contrast, our work spreads the information entropy delivered from the controller and significantly reduces the controller communication overhead; to achieve this, we delegate some intelligence (computations) to the routers; others have also proposed to offload responsibilities to the data-plane routers for increased security in other contexts [20], [22]. Third, in contrast to prior work, our work is proactive in hopping and is resilient to adaptive attackers that sense the traffic flow.

IV) Security at routers Our scheme implements security at the data-plane routers and aims to prevent unauthorized access at the first node that encounters the access. Other researchers proposed orthogonal approaches to achieve the same goal, e.g., against DoS, and adopt two classes of approach: filter-based approaches detect the unauthorized access based on traffic analyses (e.g., using source identities and anomalous behaviors) and assume the ability to distinguish the unauthorized traffic and the authorized traffic [23], [24] (we review the filter-based approaches that are specific to SDN environments in Section II-A-III), and the other capability-based approaches involve handshaking (typically with the destination endhost or a dedicated server) and explicit authorization for the path access [25], [26]. While our contribution shares similarities with such protocols in its objective and the fact that security is implemented at the router switches, we take a different approach and adopt MTD; unlike the aforementioned work, we do not require the correct classification of malicious traffic nor explicit handshaking per flow.

B. Our Work Implements Security at the Switches

We implement our randomization scheme at the data plane and on the router switches and thus the implementation scope is fundamentally different from the endhost-centric prior work, described in Section II-A-II and Section II-A-III. In other words, whereas prior work builds security at the endhosts and their communication with the network representative (e.g., SDN-controller or a DHCP server), the endhosts' operations are orthogonal to the data-plane networking and treat the actual routing and forwarding process as a black box when sending data packets. In contrast, our scheme lies within that blackbox and thus at the network layer in the OSI communication model; to provide a modular solution (while keeping the rest of the layers intact), we construct virtual switches at the endhosts in Section IV-A. Furthermore, since our scheme builds protection on the path itself and at the link-level, it enables quicker response and prevents attackers from overwhelming the links and the routers; in contrast, prior work in MTD does not implement security on the routing switches

(but only at the endhosts, e.g., [1], [2]) and is vulnerable to link-targeted DoS [27], [28].

III. System Model

A. System Assumptions

We consider a connected multi-hop network and assume IP forwarding. We build our scheme on the SDN architecture (where a controller establishes routing and the router switches forward traffic accordingly), focus on intra-domain communication applications (typically governed by a single operator), and assume packet-level synchronization between the nodes, e.g., by timestamping the packets.

We also leverage a public key infrastructure (PKI) for the distribution of the seed values that will activate our scheme (in specific, we have two seeds in IP seed and PRG seed as we discuss in Section IV). The SDN controller distributes the seed values to the data-plane nodes. We assume the security of the control communication between the control and the data plane [20], [21] (for example, by using asymmetric encryption for confidentiality and digital signature for integrity) and the controller itself [22]; we focus on threats on the data plane.

In addition to using the source's and the destination's addresses on the packet's network header, the router switch checks its own address to determine the routing and the forwarding action, which we will describe further in Section IV-D. To incorporate this feature without requiring modification to the rest of the networking architecture, we use the Options field in the IP header (which field, along with other rarely used fields, are also used for packet marking and tracing in security research, e.g., [29]) and ensure that the switches supporting our scheme checks these fields, e.g., IPv6 supports an Hop-by-Hop Options extension that not only can carry the next-hop router's address but also triggers examination for all router switches along the forwarding path.

B. Threat Model

An attacker aims to access and use the forwarding path but is defined to lack the legitimate authorization for the access; we also consider compromised network with the attacker having physical access to the forwarding path. Our scheme defends such attack by thwarting network reconnaissance, which is to learn about the victim network and its vulnerability and is a pre-requisite for further exploit. If an attacker successfully achieves network reconnaissance, then it can not only distinguish between the packet flows (characterized by the source and the destination) flowing in the link and eavesdrop on the communication between the source and the destination but can also actively inject packets on the path for selfish reasons (e.g., to deliver its own traffic without the overhead of establishing the credentials and the routing path) and malicious reasons (e.g., to spoof the source user for slandering purposes, targeted DoS on the destination, and DoS on a link).

We do not consider the orthogonal threat of disrupting the routing and the forwarding process; even though the attacker has access to the path links, it does not have control over the nodes that perform the routing and forwarding process. For example, if a node along the forwarding path is compromised and disrupts the forwarding process, e.g., re-routing and packet dropping, then the routing protocol can choose another forwarding path that does not include that node. Thus, we do not consider compromised routers that have active control on the forwarding operations.

In contrast to prior work in routing security, we consider an advanced attacker that are dynamic and adaptive. They can both passively monitor and analyze the traffic and actively send queries and probe for network reconnaissance and, afterward, use the information to facilitate their unauthorized access (an analogy to wireless communication is a reactive jammer where the jammer performs cognitive-radio-like sensing of the spectrum to decide where to focus its jamming for maximum impact). The traffic sensing and the adaptation of the access strategy is done in real time. We can envision that, if the MTD randomization supports reactive trigger (e.g., IP changes when there is a misbehavior detection), such attacker can indirectly launch a MTD-specific DoS on the control communication by repeatedly and intentionally triggering misbehavior detection and forcing the controller to continuously update the outdated security parameters, overwhelming the network with control communication updates and blocking the actual goodput delivery. (We counter these threats by developing fast and proactive randomization.)

IV. Our Scheme

We deploy IP randomization across all nodes that are involved in the forwarding path. We generate IP addresses, based on a local seed and a random seed, and use them for randomization on the router switches. The SDN controller distributes the seeds and controls the routing operations (including the routing rule updates on the router switches), as discussed in Section III-A. Much like other routing schemes [25], [26], our work can be used to establish priority-based forwarding where the traffic that uses our scheme has higher priority in the path access and forwarding; the low-priority traffic which uses the static IP address without MTD can be processed in an unobtrusive manner to the high-priority traffic or can be banned altogether, depending on the priority scheme implementation. We describe our scheme in this section and analyze it in Section V.

A. Modular Design

To provide a modular solution, we construct virtual switches within the endhost nodes, so that the packets are processed for IP hopping after the network-layer operations and before the physical-layer mappings. The operations of the virtual switches (within the endhosts) are the same as the actual switches (separate nodes from the endhosts), which are described in Figure 1, except that the source (the beginning of the path) does not perform IP de-hopping and the destination (the end of the path) does not perform IP hopping.

B. IP Hopping: Address Generation & Randomization

Unlike other schemes which implement security at the controllers [14], [15], [17]–[19], [30] including the work that

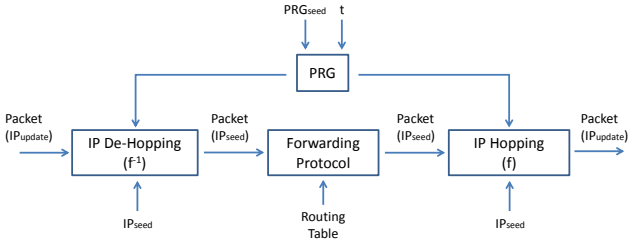

Fig. 1: Operational flow for switches

builds MTD [1], [2], we offload some control computation to the data plane and on the router switches in our work. However, the operations on the router switches are marginal in intelligence as they perform routine protocol and computations (with the control plane providing the input that drives the computation). The nodes along the forwarding path (both the endhosts and the router switches) locally implement pseudo-random generator (PRG), which is uniform and agreed across the nodes.

The endhosts and the router switches update the IP addresses from the two seeds: the IP address seed (unique for each nodes) and the PRG seed (provided from the control plane and used to drive the PRG). The local IP address seeds of the nodes are static and do not need to be private information; for example, the static IP addresses (which have been used before the activation of our scheme or is used to send the low-priority traffic) can be used as this seed; all nodes involved in the forwarding know each other's IP address seeds. The variation for randomization comes from the PRG; the function and the seeds are uniform across the nodes involved in the path (including the source and the destination endhosts, equipped with hopping-capable virtual switches). Nodes not involved in the path forwarding do not have the PRG seed, and thus the PRG output appears random to them.

The updated IP address (IP_{update}) is a function (f) of the IP seed (IP_{seed}) and the PRG output (PRG), which in turn is a function of the PRG seed (PRG_{seed}) and time (t):

$$IP_{update} = f(IP_{seed}\ ,\ PRG(PRG_{seed}\ ,\ t)) \tag{1}$$

where both f and PRG are some deterministic functions. While f^{-1} (given either IP_{seed} and PRG output) is easy to compute, PRG^{-1} and PRG (without PRG_{seed}) are difficult to compute. Also, as discussed in Section III-A, time is synchronized and t is shared across the nodes.

C. IP Hopping: Our Instantiation

We instantiate the IP hopping by applying the followings: the randomization is at the IP addresses of the nodes (e.g., in our implementation, we use Class A private addresses with subnetwork address of 10.X.X.X and subnet mask of 255.0.0.0, leaving us 24 bits for randomization[1]); the static IP address, e.g., for low-priority traffic, acts as the IP_{seed}; we use a linear feedback shift register (LFSR)-based[2] PRG (with t increment triggering the shift and $t = 0$ corresponding to no shift and being in the state of PRG_{seed}); and f is cyclic addition for each decimals (representing one byte each). Thus, both f and PRG are linear in our implementation.

The IP seed (which will also be used in Section IV-D and Section V-A) can be found by reversing the operation and using f^{-1}, i.e., cyclic subtraction by the PRG output. For the packet at time t, the legitimate users that forward the packets know the PRG output (as they are using the same PRG, PRG_{seed}, f, and t).

We illustrate the f of decimal-based cyclic addition with an example: if a user has a static address of $10.0.240.25$ and the PRG output for time $t = t'$ is $10.0.17.25$, then the updated IP address is $IP_{update} = 10.0.2.50$, and that user will use the IP address of $10.0.2.50$ at time $t = t'$. Other nodes, e.g., router switches, also compute PRG output of $10.0.17.25$ at $t = t'$ and cyclic-subtract it from the received $10.0.2.50$ to retrieve the IP seed of $10.0.240.25$.

D. IP De-Hopping & Forwarding

The route is established based on the IP_{seed} addresses (in specific, the source, destination, and the relaying switch's addresses), and the routing table is static (the routing table lists the routing rules and policies and is only updated when directed by the SDN controller). As discussed in Section III-A, we use the IP header (which is accessed by the router switches by design) to enable the router switches to check its own IP address for path access validation.

As depicted in Figure 1, given the routing table, the forwarding nodes locally decrypt the IP randomization by performing f^{-1} with the PRG output to check whether the traffic is addressed correctly (and hence legitimate). Once packets arrive, the forwarding nodes first check its own (updated) IP address, the destination's, and the source's; by computing f^{-1} of cyclic subtraction, the forwarding node learns the IP_{seed}s of the addresses and identify the routing path according to them. If the IP addresses are valid and the corresponding path exists in the routing table, the user identifies the IP_{seed} of the next hop, compute the IP_{update} of the next-hop user, and forwards the packet while addressing the next hop with the computed IP_{update}; otherwise, the packet is dropped. All router switches perform this process.

V. Analyses

A. Hopping Collision for Multiple Flows

We consider the case of multiple traffic flow in a link. If multiple flows coexist and the path carries traffic from multiple sources, there can be collisions in IP_{update} in Equation 1. However, if f conditioned on IP_{seed} is injective with PRG (i.e., different PRG yields distinct $f \mid IP_{seed}$), then we can resolve collision and distinguish packets that are addressed to

[1]The entropy is the same as the size of the host address space. Thus, adopting our scheme with IPv6 will significantly increase the randomness.

[2]LFSR is popularly used for military applications such as cryptography and scrambling (e.g., DSSS) and is known to generate maximum entropy against brute-force threats, as it produces maximal length sequences (i.e., if the bit length is l, it produces $2^l - 1$ distinct states before it repeats itself).

the same IP_{update} by coupling the information of IP_{update} and IP_{seed}. In other words, if traffic from different flows address the router switch with the same IP address at the same t, then the two packets can be distinguished by computing f^{-1} and identifying the IP_{seed} of the respective flows.

B. Security Analyses

Attackers who wish to compromise a forwarding path need to know that the routing path based on the IP_{seed} is established in the routing table and the corresponding IP_{update}s of the source, destination, and the relaying switch node. The security of our scheme relies on the security of PRG_{seed} and consequently the PRG output. To defeat reactive attackers who observe the packet traffic and compromise the relevant set of IP_{update}, we have the IP update more quickly than the attackers' reaction time. Because the IP generation and updates are being processed within the host (as opposed to via control communication with a separate entity, the controller [1], [2]), our scheme is fast and significantly increases the attackers' cost, as we will see in Section VI-B.

To protect PRG_{seed}, we use $t > 0$ for the IP randomization to avoid using the plaintext PRG_{seed} as the PRG output with the initial packet (to defeat lurking attackers who capture the network-layer information at the first packet). However, we do not rely on the secrecy of f and PRG themselves; both f and PRG are public information by Kerchoff's principle. Our main contribution does not lie in building a secure PRG, but we rely on the security properties of mature and well-adopted techniques for such functions (for example, it is difficult to compute PRG without PRG_{seed} and t); thus, we use LFSR-based scrambling for PRG.

Our scheme is secure against an attacker that monitors the traffic over time. Such an attacker may want to gather the history of the utilized IP addresses to acquire the PRG_{seed}, but linking the individual packets to the flow and the source-destination pair (to learn the per-flow packet sequences) is challenging, especially when multiple users and flows are using the link (where the attacker compromise resides). Even when the attacker is successful in gathering and storing the history of the IP address updates using some side channel information (for instance, it has the assurance that there is only one traffic flowing), the attacker needs to break the PRG to learn the PRG_{seed} and access the path.

VI. Implementation Evaluation

We build a OVS-based testbed prototype to evaluate our IP hopping scheme; Section VI-A describes the prototype implementation. After the seed distribution, we take benchmark measurements to show the performance gain of our scheme and the increased security against a reactive attacker, described in Section III-B. Since the performance values are sensitive to the system implementation details [31], [32], we describe the performance in gains over references. Section VI-B measures the packet delivery latency and compares our IP hopping scheme with the prior state-of-the-art IP randomization scheme (that involves communication with the controller for every IP update), and Section VI-C discusses how our fast IP hopping defeats the network reconnaissance attack. The delay measurements are taken over 10,000 samples.

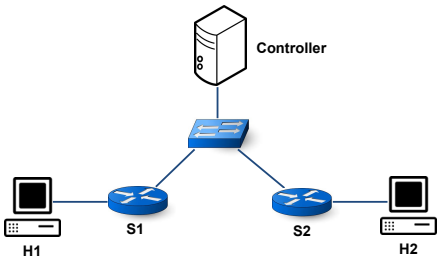

Fig. 2: Our experimental setup

A. Prototype Implementation

To construct a local network for our experiment, we use five computers (Intel Core 2 Duo, 2×2.20GHz CPU with 2.0GB RAM) and an ethernet switch for the physical devices. There are two endhosts (the source and the destination), two router switches, and a controller server. The router switches are software-based in Open vSwitch 2.3.0, and the controller is based on OpenDaylight Helium supporting OpenFlow 1.3. Unlike the router switches, the physical ethernet switch device does not support any virtualization and has the sole purpose of bridging the wired connections. Figure 2 depicts the network topology and the roles of the computer hosts; we focus evaluating our scheme given a forwarding path.

We implement a prototype for our scheme. As PRG has comparable data format as IP addresses, we use DHCP protocol for the PRG seed distribution with the DHCP server implemented at the controller. Then, the nodes locally randomize their IP addresses, as described in Section IV-C, and send packets with the updated IP addresses. Only when the IP addresses are correct do the packets go through, as discussed in Section IV-D. For example, when the host H1 directs its packets to other hosts with incorrect IP addresses, e.g., because it is unauthorized and does not have the correct IP update, then the packets get dropped at the next hop at S1.

B. Packet Delivery with IP Hopping

We compare our scheme to a baseline reference. The *baseline* scheme is to involve the SDN controller for the IP randomization of the nodes, which is adapted from the state-of-the-art mechanism to build MTD on the hosts in SDN [1], [2]; however, the baseline scheme differs from these prior work in that they randomize the addresses of all nodes on the forwarding path as opposed to just the endhosts. The gain of our scheme over the baseline comes from the fact that the randomization operations are done locally within the nodes as opposed to involving the controller. After the PRG seed is distributed to the endhosts and the routers, we generate 64-byte ICMP packets (ping), apply randomization for each packet, and compare the packet latency between the two different MTD approaches. We also isolate the operations and measure the latency when the randomization/update is done locally via LFSR computation and when it is done via the interactions

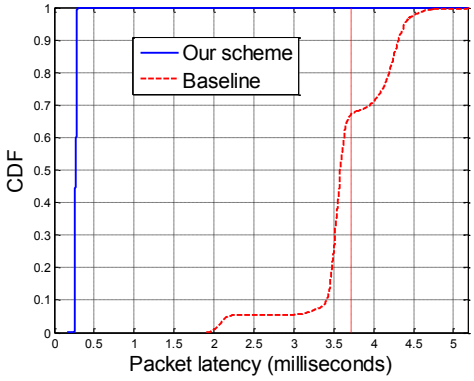

Fig. 3: The measurement distribution of our scheme and the controller-driven baseline. For baseline, we also show the average in a vertical line.

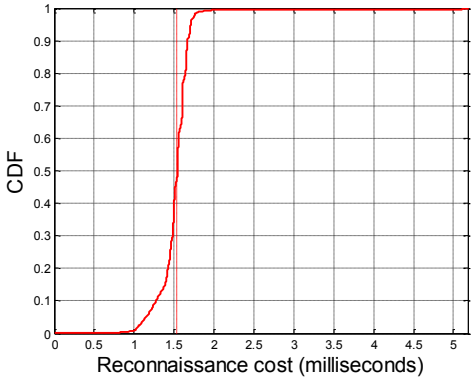

Fig. 4: The measurement distribution of attacker's cost in achieving network reconnaissance. The horizontal axis is aligned to Figure 3.

with the controller. The latency performance for our scheme is 0.2680 milliseconds per packet delivery (averaged over 10,000 measurements) and the IP generation/update takes 14.58 nanoseconds (averaged over 10 cycles or $10\times(2^{24}-1) \approx 1.678 \times 10^8$ IP addresses), which accounts for a marginal $\sim 0.0054\%$ of the total packet latency. On the other hand, the controller-involved baseline scheme has an average latency of 3.7159 milliseconds per packet delivery where the controller overhead for IP update accounts for $\sim 93\%$ of the latency. The latency distribution of our measurements are shown in Figure 3. Thus, our scheme compares favorably to the baseline, as the attacker needs to improve its reaction time by more than an order of magnitude to successfully breach the updated IP address.

The latency performance is highly dependent on the system and the SDN implementation [31], [32] and the network topology, and the performance optimization or more sophisticated analysis is beyond the scope of this paper. Nevertheless, our performance gain is conservative and the actual latency for the baseline scheme in real world is likely higher because of the following reasons: there is no other traffic in our setup (in contrast, real-life scenarios with multiple traffic flow introduce queuing delays); and our network topology is latency-friendly, as there is one-hop connection between the controller and the routers and two-hop connection between the controller and the endhosts. The randomness across packets will also increase outside of the lab setting; Figure 3 shows greater randomness in the measurements when the controller is involved (baseline) than having the randomization performed within the nodes (our scheme) even in our controlled lab environment.

C. IP Hopping Against Network Reconnaissance

To understand the attacker's perspective against our scheme, we instantiate a network reconnaissance threat by having a malicious node inquire for the node's IP address via traceroute once it encounters the traffic. This approach provides an immediately feasible implementation of the attack; this attack can be defeated (e.g., although often enabled by default, enabling traceroute is not compulsory and the router can decide not to engage with the attacker), but it provides an estimate of the attacker's cost in our implementation setup. As discussed in the following, our scheme can defend against such network reconnaissance attack.

Figure 4 shows the delay cost of the attacker, which is 1.5462 milliseconds on average. In contrast, our scheme costs 14.58 nanoseconds for IP generation, as the update is locally computed within the node (without requiring communication with outside of that node), and the average packet latency is 0.2680 milliseconds, as discussed in Section VI-B. Therefore, the attacker's reconnaissance delay is more than five times greater than the packet delivery latency when using our scheme. In other words, if we restrict the liveness of the IP update hop by five packets or less, then the information becomes outdated by the time the attacker achieves reconnaissance (and begins exploiting it). However, unlike our scheme defending against reconnaissance, the controller-driven IP randomization (the baseline scheme in Section VI-B) is not fast enough to defend against the implemented reconnaissance attack even if the IP hopping is implemented for every packet, leaving about 1.9 milliseconds for the attacker to use and exploit the vulnerability from reconnaissance.

VII. Conclusion

We build IP-address-based MTD in SDN to secure the forwarding path access against network reconnaissance. The defense is implemented at the data plane and at the link level and thus prevents the unauthorized access from the initial point of breach. Considering an advanced reactive attacker, we observe that our scheme (offloading the randomization operations at the switches) outperforms the baseline scheme (having the controller randomize the nodes's addresses) and increases the attacker's requirement cost for timely reconnaissance by more than an order of magnitude in our controlled lab environment; we project the gain to be greater in a less controlled environment.

VIII. Acknowledgements

This research is supported by the research grant for the Human-Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A⋆STAR) We would also like to thank the anonymous reviewers for their feedback.

REFERENCES

[1] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 127–132. [Online]. Available: http://doi.acm.org/10.1145/2342441.2342467

[2] P. Kampanakis, H. Perros, and T. Beyene, "Sdn-based solutions for moving target defense network protection," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*, June 2014, pp. 1–6.

[3] B. Awerbuch, A. Richa, and C. Scheideler, "A jamming-resistant MAC protocol for single-hop wireless networks," in *PODC*, Aug. 2008, pp. 45–54.

[4] S.-Y. Chang, Y.-C. Hu, and N. Laurenti, "Simplemac: A jamming-resilient mac-layer protocol for wireless channel coordination," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 77–88. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348556

[5] S.-Y. Chang, Y.-C. Hu, and Z. Liu, "Securing wireless medium access control against insider denial-of-service attackers," in *Proceedings of the IEEE Conference on Communications and Network Security*, ser. CNS '15. IEEE, 2015.

[6] B. Muntwyler, V. Lenders, F. Legendre, and B. Plattner, "Obfuscating ieee 802.15.4 communication using secret spreading codes," in *Wireless On-demand Network Systems and Services (WONS), 2012 9th Annual Conference on*, Jan 2012, pp. 1–8.

[7] S.-Y. Chang, J. Lee, and Y.-C. Hu, "Noah: Keyed noise flooding for wireless confidentiality," in *Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, ser. Q2SWinet '15. New York, NY, USA: ACM, 2015, pp. 141–148. [Online]. Available: http://doi.acm.org/10.1145/2815317.2815329

[8] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, "Short paper: Reactive jamming in wireless networks: How realistic is the threat?" in *Proceedings of the Fourth ACM Conference on Wireless Network Security*, ser. WiSec '11. New York, NY, USA: ACM, 2011, pp. 47–52. [Online]. Available: http://doi.acm.org/10.1145/1998412.1998422

[9] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, "Defending against hitlist worms using network address space randomization," in *Proceedings of the 2005 ACM Workshop on Rapid Malcode*, ser. WORM '05. New York, NY, USA: ACM, 2005, pp. 30–40. [Online]. Available: http://doi.acm.org/10.1145/1103626.1103633

[10] P. Mittal, D. Kim, Y. Hu, and M. Caesar, "Towards deployable ddos defense for web applications," *CoRR*, vol. abs/1110.1060, 2011. [Online]. Available: http://arxiv.org/abs/1110.1060

[11] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront, "Mt6d: A moving target ipv6 defense," in *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, Nov 2011, pp. 1321–1326.

[12] M. Albanese, A. De Benedictis, S. Jajodia, and K. Sun, "A moving target defense mechanism for manets based on identity virtualization," in *Communications and Network Security (CNS), 2013 IEEE Conference on*, Oct 2013, pp. 278–286.

[13] D. Kewley, R. Fink, J. Lowry, and M. Dean, "Dynamic approaches to thwart adversary intelligence gathering," in *DARPA Information Survivability Conference amp; Exposition II, 2001. DISCEX '01. Proceedings*, vol. 1, 2001, pp. 176–185 vol.1.

[14] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "Flowguard: Building robust firewalls for software-defined networks," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 97–102. [Online]. Available: http://doi.acm.org/10.1145/2620728.2620749

[15] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on {SDN} environments," *Computer Networks*, vol. 62, no. 0, pp. 122 – 136, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613004003

[16] Y. Park, S.-Y. Chang, and L. M. Krishnamruthy, "Watermarking for detecting freeloader misbehavior in software-defined networks," in *Proceedings of the International Conference on Computing, Networking, and Communications*, ser. ICNC '16. IEEE, 2016.

[17] R. Jin and B. Wang, "Malware detection for mobile devices using software-defined networking," in *Proceedings of the 2013 Second GENI Research and Educational Experiment Workshop*, ser. GREE '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 81–88. [Online]. Available: http://dx.doi.org/10.1109/GREE.2013.24

[18] J. Li, S. Berg, M. Zhang, P. Reiher, and T. Wei, "Drawbridge: Software-defined ddos-resistant traffic engineering," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 591–592. [Online]. Available: http://doi.acm.org/10.1145/2619239.2631469

[19] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 29–42. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482631

[20] 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*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 413–424. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516684

[21] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, Oct 2010, pp. 408–415.

[22] S. Matsumoto, S. Hitz, and A. Perrig, "Fleet: Defending sdns from malicious administrators," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 103–108. [Online]. Available: http://doi.acm.org/10.1145/2620728.2620750

[23] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer dos defense against multimillion-node botnets," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 195–206, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402981

[24] K. Argyraki and D. R. Cheriton, "Active internet traffic filtering: Real-time response to denial-of-service attacks," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247360.1247370

[25] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing internet denial-of-service with capabilities," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 39–44, Jan. 2004. [Online]. Available: http://doi.acm.org/10.1145/972374.972382

[26] A. Yaar, A. Perrig, and D. Song, "Siff: a stateless internet flow filter to mitigate ddos flooding attacks," in *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, May 2004, pp. 130–143.

[27] M. S. Kang and V. D. Gligor, "Routing bottlenecks in the internet: Causes, exploits, and countermeasures," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 321–333. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660299

[28] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, ser. SP '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 127–141. [Online]. Available: http://dx.doi.org/10.1109/SP.2013.19

[29] Y. Xiang, W. Zhou, and M. Guo, "Flexible deterministic packet marking: An ip traceback system to find the real source of attacks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 4, pp. 567–580, April 2009.

[30] M. Shtern, R. Sandel, M. Litoiu, C. Bachalo, and V. Theodorou, "Towards mitigation of low and slow application ddos attacks," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, March 2014, pp. 604–609.

[31] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, ser. Hot-ICE'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228283.2228297

[32] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of sdn/openflow controllers," in *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, ser. CEE-SECR '13. New York, NY, USA: ACM, 2013, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/2556610.2556621