Community Cleanup: Incentivizing Network Hygiene via Distributed Attack Reporting

Yu Liu, Craig A. Shue
Worcester Polytechnic Institute
{yliu25, cshue}@wpi.edu

***Abstract*—Residential networks are difficult to secure due to resource constraints and lack of local security expertise. These networks primarily use consumer-grade routers that lack meaningful security mechanisms, providing a safe-haven for adversaries to launch attacks, including damaging distributed denial-of-service (DDoS) attacks. Prior efforts have suggested outsourcing residential network security to experts, but motivating user adoption has been a challenge. This work explores combining residential SDN techniques with prior work on collaborative DDoS reporting to identify residential network compromises. This combination provides incentives for end-users to deploy the technique, including rapid notification of compromises on their own devices and reduced upstream bandwidth consumption, while incurring minimal performance overheads.**

***Keywords*-Software-defined networking, residential network security, distributed denial-of-service attacks.**

I. Introduction

Modern residential network security faces multiple challenges: a lack of security expertise in each home, a rising number of consumer-grade Internet of Things (IoT) devices, and capacity improvements that make residential networks ripe for attack. Internet-connected embedded devices pose a unique challenge since they may be deployed for long periods but without maintenance from either the end-user or manufacturers. Attackers have recognized the potential of these devices, including in the Mirai botnet that launched distributed denial-of-service attacks in excess of 1 Tbps [1].

Residential networks typically lack dedicated security hardware. Instead, they typically leverage network address translation (NAT), which has the side-effect of blocking in-bound connections by default, as a security measure [2]. However, NAT was not designed for security and has substantial limitations when used in this fashion [3]. Further, attackers are well aware of NAT implementations and design their attacks to circumvent them [4].

The presence of vast swaths of insecure networks makes the Internet, as a whole, less secure. Mobile devices may be compromised on residential networks and later physically move into corporate networks where the contamination spreads [5]. Insecure networks also offer opportunities for adversaries to anonymize their traffic and attack with impunity.

Although distributed denial-of-service (DDoS) attacks are a challenge for residential networks, they also represent an opportunity to inform home users that their security has been breached. Typically, residential users are unaware when their devices are compromised. Owners of devices participating in the Mirai botnet had no way of knowing their systems were attacking others on the Internet. If informed of a compromised device, many owners would remediate the issue to avoid negative consequences for themselves and others. In DDoS attacks, a victim learns the IP addresses of a large set of attacking hosts, which are typically compromised machines organized in a botnet. If victims could easily report these attacks, the device owners could learn about the compromise and remediate the devices, lessening the flood and preventing the devices from being involved in future attacks.

To address these problems, the research community has explored ways to connect experts with home networks. Feamster [6] recommends outsourcing all security and network management to experts. With Project BISmark [7], a remote service provider is able to periodically measure network performance across a collection of residential networks. Taylor et al. [8] found that cloud-based software-defined network controllers would have acceptable latency for the vast majority of residential networks in the United States even when deploying fine-grained flow control. This suggests external providers could feasibly manage network access control for residential networks.

In their Active Internet Traffic Filtering (AITF) work [9], Argyraki and Cheriton proposed a cooperative filtering scheme in which victims could issue filtering requests to Internet Service Providers (ISPs) close to attacking hosts, preventing unwanted traffic from saturating bottleneck network links. A significant challenge for the AITF work was a lack of incentives for deployers: the filtering ISPs would have to modify their routing infrastructure to support AITF, but those ISPs did not directly receive benefits in return.

In this work, we explore opportunities to provide all

deployers with incentives to participate. We combine fine-grained SDN access control with AITF's cooperative traffic filtering techniques. We explore a system where SDN controllers from different service providers, which can either be hosted by ISP or third party, could automatically discover each other, report attacks, and cooperate on network filtering. In doing so, we make the following contributions:

- **Exploration of Incentives:** We examine how each of the stakeholders in the system benefit from this approach. These stakeholders include the owners of compromised devices, the attacked victims, and SDN service providers for both the attackers and victims (Section IV).
- **Architecture Design and Implementation:** We provide an overview of how the approach works, a protocol that allows the SDN controllers and consumer-grade residential routers to identify and verify each other, and implement a distributed attack reporting system (Sections III and V).
- **Security and Performance Evaluation:** We evaluate the approach's reporting from a security perspective and analyze the performance at each device (Section VI). We find that the approach can report and verify a compromised host within seconds and introduces few performance overheads in the controllers and routers.

II. Background and Related Work

In this section, we provide background and discussion on distributed denial-of-service (DDoS) attacks, software-defined networking (SDN), and efforts to protect residential networks.

A. Distributed Denial-of-Service (DDoS) Attacks

DDoS attacks, in which attackers exhaust a victim's resources through a large volume of requests, are a common and on-going phenomenon [10]. Botnets of compromised machines can be large, encompassing millions of machines [11]. A recent botnet, Mirai, is designed to run on compromised Internet of Things (IoT) devices [1] and continues to evolve [10].

The research community has explored methods to detect and mitigate DDoS attacks. Prior detection efforts have included statistical and mathematical methods to distinguish DDoS traffic from benign traffic [12]–[14], an analysis of network distance between the destination and origin [15], the application of deep learning [16], and even entropy comparisons of flows [17]. Some approaches have looked at ISP-wide data to detect botnets [18], though botnets typically operate globally across ISPs [19].

Once an attack is detected, defenders can try to mitigate it. Some endpoint filtering approaches have used IP history and reputation to determine malicious and legitimate senders [20], [21]. However, with IP rotation and sharing in DHCP and NAT environments, IP history and reputation may have limited value. Other endpoint solutions include the work by Buragohain et al. [22], which performs flow-modeling on a SDN controller, and the work by Rebecchi et al. [23], which performs stateful anomaly detection on traffic at a router. A common limitation for these approaches is that filtering at the endpoint, or at network devices in the same LAN as the endpoint, is ineffective when the saturated bottleneck network link is between the LAN and its ISP. Such "last mile" bottleneck links are common on the Internet.

Other approaches have sought to employ packet filtering before the traffic reaches the victim's bottleneck link. Weniger et al. [24] propose a "moving target" mechanism in which ISP customers have multiple addresses and can unsubscribe from an address upon detecting an attack. This allows the upstream ISP to filter traffic to an unsubscribed address. Mahajan et al. [25] propose a method to send filtering requests to upstream routers. Likewise, in their work on AITF, Argyraki et al. [9] designed a filtering protocol for routers at both the victim and attacker networks to cooperatively filter traffic. These approaches have powerful filtering capabilities, but they require cooperation from a set of ISP routers, ideally some at high-traffic peering points. The reliance on ISP cooperation and lack of incentives for ISPs to deploy this approach have resulted in little adoption of these techniques. In this work, we shift the filtering load to SDN controllers operated by any service providers who have incentives to report and filter malicious traffic.

B. OpenFlow and Software-Defined Networking

The software-defined networking (SDN) paradigm separates control decisions from data plane processing, allowing a remote control system to manage the rules cached in network switches and routers. The OpenFlow [26] protocol allows routers and switches to elevate packets that do not match any locally cached rules to an OpenFlow controller. The controller can respond by providing new rules and instructions on how to handle each packet. Unlike the routers and switches, the controller has visibility across the network and runs on standard computer hardware, allowing it to perform more sophisticated traffic analysis. The flow rules can specify each criteria for the fields in common layer 2, 3, and 4 network headers along with an associated action. These fields can be wildcarded or require specified values for matching purposes. A flow rule is considered "fine-grained" if it fully specifies a flow consisting of the source and destination IP addresses, the transport layer protocol, and the source and destination transport layer ports. With careful caching strategies, OpenFlow

controllers can gain visibility into network traffic while limiting overheads with rule caching so that traffic is processed quickly.

A significant amount of SDN research has focused on data center or enterprise networks. However, residential networks typically have different characteristics, including network throughput, asymmetries between upload and download bandwidth, and management practices. While each residential network tends to be fairly small, they play an important role on the Internet. Recent surveys have shown that around 89% of households have multiple computers at home and around 82% of US households have Internet connections [27]. Around 76% percent of US home networks use a wireless router [28] to share network resources. These networks provide connectivity to a range of Internet-enabled devices, with some predictions of over 442 million connected home devices by 2020 in the United States alone [28]. Internet of Things devices represent a unique concern because of their long deployment periods and the fact that they may not be updated and patched as frequently as traditional computers, tablets, or smartphones.

In a 2010 position paper, Feamster [6] proposed outsourcing residential security management to services operating in cloud infrastructure. Subsequent work explored the feasibility of doing so. Taylor et al. [8] found that 90% of residential networks in the United States were within a 50 millisecond round-trip of a public cloud data center. Those results suggest that cloud-based service providers could operate SDN controllers that monitor network flows while causing little delay. Other work examined the potential for ISP-hosted SDN controllers [18] since they would introduce lower latency. However, the reliance upon ISP cooperation has limited the deployment of these proposed approaches.

In this work, we explore an approach that could operate on an SDN controller in either public cloud data centers or in ISP-provided data centers. This grants independence from specific ISPs while still leveraging close network proximity, where available.

III. Architecture and Design

We now describe the architecture and components we will use to combine SDN techniques with distributed attack reporting. We describe the incentives for this approach in Section IV and describe the implementation details of this design in Section V.

In Figure 1, we show an example network configuration. In this diagram, we have an attack target, which we call the "victim host," that is is connected via a router to the Internet. The victim router can be a commodity consumer-grade wireless router, such as the variety used in 76% of households [28]. The router is configured to run Open vSwitch [29], a popular OpenFlow agent implementation. It communicates with an OpenFlow controller, labeled the "victim controller," which is run by a security service provider. The victim controller could run in a public cloud data center, ISP-hosted data center, or other location. In our model, we assume that the controller is not in the same LAN with the victim host or router. The victim router and controller are connected to the Internet via ISPs, which we merge together in a simple box labeled "Internet." We depict the bottleneck bandwidth link as being between the victim router and its ISP, since this is the common "last mile" link that constrains throughput.

Since network attacks are illegal in many jurisdictions, the actual perpetrator of a DDoS attack will often use compromised systems to actually send the attack traffic to the victim. We call the compromised device a "pawn" to indicate that it is working for an attacker (though the device owner is unaware that this is occurring). In our example network, we show a pawn host connected to its own router, which connects with its own controller. The pawn could share an ISP with the victim or use an ISP that is distant in the topology. Multiple pawns could be working in concert as part of a botnet. We consider the pawn to be fully controlled by the attacker. However, the pawn's router and pawn's controller are managed by a security service provider that works for a legitimate device owner who does not realize the device has been compromised. Accordingly, the pawn's router and controller may curtail the pawn's activities upon receiving evidence of a compromise.

In the event of an attack, the victim's controller will initiate communication with the pawn's controller to request filtering of the traffic. The two controllers may filter the traffic with an OpenFlow flow modification (`FlowMod`) message that indicates that packets matching the unwanted flow's IP addresses and ports should be discarded. The mechanisms to discover and verify the responsible controllers will be described later in this section.

A. Threat Model

The defender's goal is to report and stop unwanted traffic as quickly as possible. We assume the defender has an existing technique to determine which traffic is unwanted. The victim can request filtering from the victim router–explicitly or via continual connection reset messages–trusting the security service provider and considering the victim router and victim controller to be a trusted computing base (TCB).

The owner of the compromised device likewise trusts the security service provider operating the pawn router and pawn controller and considers them in a TCB. The pawn's owner wants to know if the pawn is ever compromised and to prevent it from engaging in attacks.

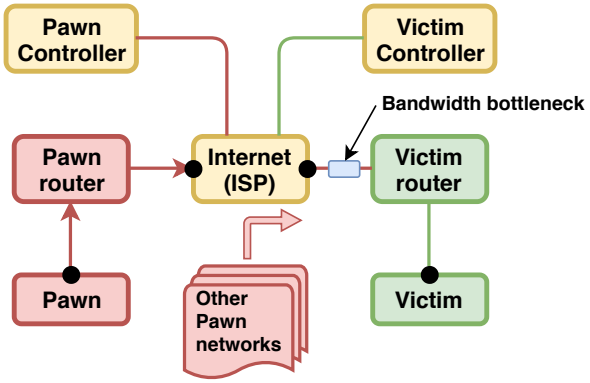

Fig. 1. An overview of the components in our system. The victim and pawns are connected via their respective routers and managed by their respective controllers. The black dots represent measurement points in our evaluation.

The pawn's owner does not consider anyone else to be trusted. If a given destination will block communication with its own hosts anyway, the pawn's owner gains upload bandwidth capacity by filtering that traffic at its local router when participating in an attack.

The security service providers for the victim and pawn may be mutually distrusting. In that case, the providers will need a way to securely identify and verify each other. A service provider only needs to confirm that the other service provider is an agent for the remote communicating party. In other words, if the other service provider can prove it can intercept packets destined to the remote communicating party, then it is allowed to request that filtering be done closer to the source. The service providers assume the links between the two controllers, between the two routers, and between the routers and controllers are uncompromised. If this assumption is violated, fake reports and filtering requests could be introduced. However, those on-path adversaries would already have the capability of simply discarding traffic.

B. Protocol Design

In Figure 2, we provide a sequence diagram of communication between a pawn and a victim. When the pawn initiates its connection to the victim, the request reaches the pawn's router. Since this is the first entry in the flow, and because the pawn's router uses fine-grained flow entries, the pawn router does not find a matching flow entry and elevates the request to the pawn controller. The pawn controller decides whether to allow the traffic or not. If it is allowed, the pawn controller generates a special value, `SRC_COOKIE`, that can be used by recipients or other controllers to prove that it received the packet. The controller then sends the packet back to the pawn router using a standard OpenFlow `PacketOut` along with any `FlowMod` messages needed to authorize the flow.

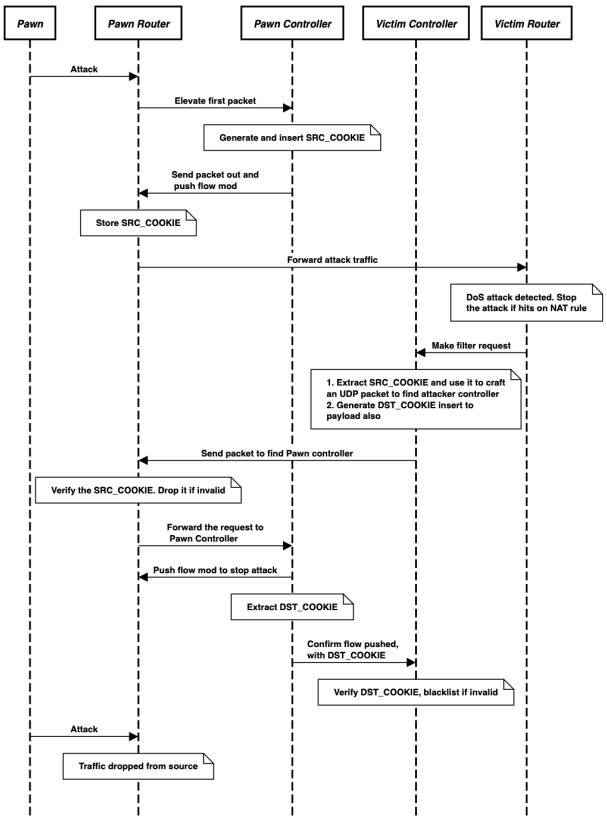

Fig. 2. A sequence diagram overview of the reporting protocol

Before transmitting the packet as ordered, the pawn router first inspects the message in the `PacketOut` to search for any cookie values the controller may have inserted. If it finds one, it stores the `SRC_COOKIE` in a local database for future consultation. It then transmits the packet towards its destination.

The packet is then transmitted across the Internet until it reaches the victim router. The victim router detects any new flows and searches for any cookie values associated with them. If one is detected, it records the `SRC_COOKIE` to a local database. If it ever determines that the flow needs to be filtered in the future, it can use the `SRC_COOKIE` as an authenticator value.

When a victim router recognizes that traffic should be filtered for a victim host, it first filters the packet locally by adding a drop rule. It then contacts the victim controller and reports the filtering request, along with the `SRC_COOKIE`, if any, that the victim router previously stored when the flow was being created. The filtering request contains the full flow information (the IP addresses of the pawn and victim, the transport protocol and associated transport layer ports). The victim router sends this request to a pre-defined port on the victim controller.

The victim controller listens for filter requests on its pre-defined port. It examines each part of the filtering

request and then attempts to contact the controller associated with the pawn. It does so by sending a packet to the pawn's public IP address using another predefined, globally-known port value for filtering requests. The victim controller supplies all the information it received from the victim router and generates its own value, `DST_COOKIE`, that will serve as a nonce and authenticator for the pawn controller. It then sends the request to the pawn router.

The pawn router listens for filtering requests on the well-known request port. If it sees requests to its own IP address (if the router uses NAT) or to IPs associated with its own hosts (when NAT is not in use), it extracts the packet and determines whether it is a valid filtering request. It examines the packet to determine the flow that should be filtered and the associated `SRC_COOKIE`. If it finds the extracted cookie in its database, and the cookie is associated with the indicated flow values, it knows the filtering request is valid and sends it on to the pawn controller. If there is a mismatch in cookie value and flow, or if either of the fields is invalid, the router simply drops the filter request packet.

Upon receiving the filter request packet from the pawn router, the pawn controller also verifies the `SRC_COOKIE` and flow details match. If so, it sends an OpenFlow `FlowMod` to the pawn router that discards any subsequent packets in the flow. It then uses the victim controller's IP address, which is the source of the request, to send an acknowledgement of the filtering request. By including the `DST_COOKIE` that the victim controller supplied, the pawn controller confirms its agency for the pawn. When the victim controller receives this acknowledgment, it can verify the `DST_COOKIE` and know that the pawn controller has received the report and may start filtering the traffic.

IV. Exploration of Incentives

While residential users care and are willing to make sacrifices to improve their computer security [30], [31], they need tangible benefits to encourage their use of security technology. For security service providers to help these residential users, they need a way to prove their value to these end users to justify their service charges. In this section, we explore the incentives for both these stakeholders.

A. Incentives for Residential Users

Home users want to have confidence that they can trust their computing devices (i.e., they are not compromised). With our approach, any victim under attack has an automated way to report the incident. This allows security service providers to link attack reports and inform device owners of any allegations. In essence, this model crowdsources attack intelligence and can quickly notify device owners about problems.

Home users also want high bandwidth network connections so they spend less time waiting for online services. When a device is compromised and participates in a DDoS attack, it uses upstream bandwidth, which tends to be more limited for residential users [32]. With filtering requests from victims, the attacker's router blocks the traffic from compromised systems locally, preventing the attacks from consuming upstream capacity between the user's router and ISP.

While it may also appear that a victim of a DDoS attack would gain all the benefits of remote filtering (a goal of the original AITF paper), that outcome is spoiled by any pawns that do not deploy the technique. Unfortunately, perfect deployment may be elusive on the Internet. Instead, the costs of a DDoS attack will become higher for adversaries as adoption expands, since more bots will be blocked by source filtering and the unfiltered bots will command a price premium. Over time, organizations may begin prioritizing traffic for hosts that operate in compliant networks, further incentivizing adoption. This may affect the economics of DDoS attacks as a whole, but are unlikely to yield tangible benefits for the victim of any given attack.

B. Incentives for Security Service Providers

Service providers need ways to provide evidence that their services are worthwhile to customers. With distributed attack intelligence, they can show customers the number of remote parties each device interacted with and the number of parties that deploy the technique. They can report problems and track the number of reported issues. By showing users they are collecting reports, they assure users that compromises are likely to be identified quickly.

If a device does become compromised and the service provider must filter its flows, the service provider can provide meaningful feedback about who is reporting the attack, the device involved, and the number of complaints. End-users may have difficulty understanding the technical details of a compromise or network attack, but quantified data about the number and variety of reports may be more intuitive and compelling. Service providers can then provide guidance on remediation and an analysis on whether other devices may have likewise been compromised. Further, service providers can use attack reports to identify the command-and-control infrastructure of botnets by analyzing traffic that precedes an attack and to perform detailed analysis on traffic when reports are received.

V. Implementation

To explore the distributed reporting approach, we create an implementation on consumer-grade residential network routers. We flash three TP-Link Archer C7

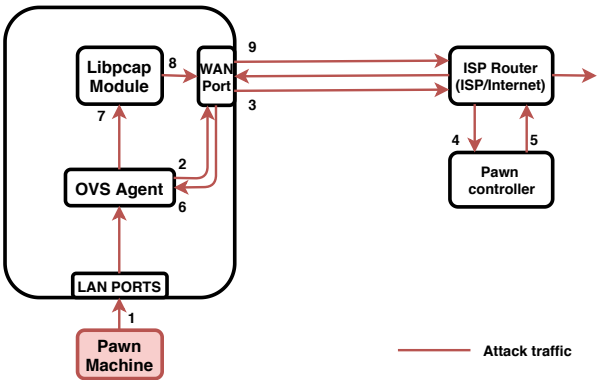

Fig. 3. This figure shows the interactions between pawn host and the OpenFlow router and controller. The process begins with the attack traffic from the pawn (line 1) which the pawn OVS router elevates to the pawn controller (lines 2, 3, 4). The pawn controller then generates a `SRC_COOKIE` and sends it back to the OVS router (lines 5 and 6). The OVS agent then transmits the packet to the victim (line 7), which is intercepted by a `libpcap` agent that stores the cookie value before sending it onward to the victim (lines 8 and 9).

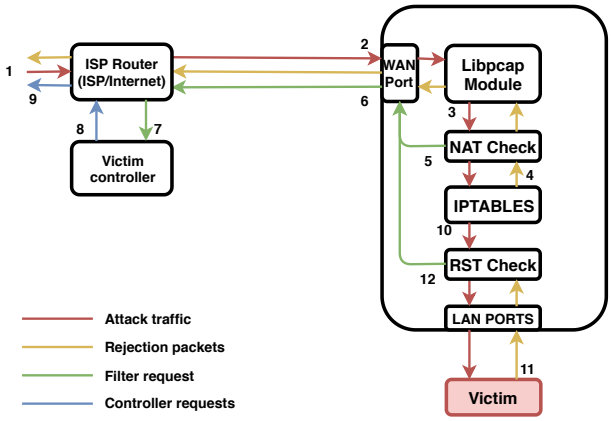

Fig. 4. The pawn's transmission is forwarded to the victim router (lines 1 and 2). On the victim router, a `libpcap` agent extracts and stores the `SRC_COOKIE` locally before they are processed by the router's NAT and firewall components. If the packet is sent to a port without a NAT mapping, the router issues a filter request to the victim controller (lines 4, 5, 6, 7). If a NAT entry exists, the router sends the packets to the victim (line 10). If the destination issues a large number of resets (line 11), the router likewise requests a filter request (lines 12, 6, 7). When processing a filter request, the controller inserts a `DST_COOKIE` and reports the attack to the pawn controller (lines 8 and 9).

v2.0 routers with the OpenWrt [33] firmware, which is a popular open source router operating system. We configure two of the three router to run the Open vSwitch (OVS) OpenFlow agent. We host the pawn controller and victim controller in VMs on a laptop running OS X. That laptop has four cores and 16GBytes RAM. Each controller runs POX [34], a Python-based OpenFlow controller and stores records using a MySQL database.

To model an upstream ISP router, we place the router that is not running OVS in the center of the network to link the other two via LAN ports. We further connect the laptop running the controller VMs to a LAN port on

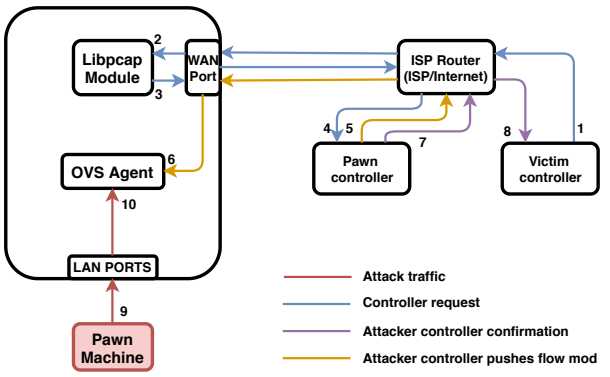

Fig. 5. The victim router's attack report (line 1) is sent to the pawn router's `libpcap` module (line 2). That module extracts and verifies the `SRC_COOKIE` in the request and compares it to the supplied flow information. If the request is valid, the router sends the request to the pawn controller (lines 3 and 4). The pawn controller again verifies the `SRC_COOKIE` and flow information. If valid, the controller records the report and pushes a `FlowMod` to the pawn router to filter the flow (lines 5 and 6). Meanwhile, it sends an acknowledgment to the victim controller (lines 7 and 8). Subsequent packets to the victim from the pawn are thus dropped due to the new rule (lines 9 and 10).

the ISP router. This upstream ISP router is configured so that each physical LAN port is on a separate VLAN and we use the `tc` command to constrain the downsteam bandwidth to 5Mbps to allow attack modeling without any testing hardware being a limiting factor. We manually configure the IP addresses, subnets, and routes on the routers and configure IP aliases on the pawn host to enable a single system to simulate multiple attacking hosts.

One of the two remaining TP-Link routers is designated as the pawn's router and the other is designated as the victim's router. The pawn's router elevates new flows to the pawn controller to obtain the `SRC_COOKIE` as described in Section III-B. Upon receiving a response, the pawn router stores the `SRC_COOKIE` locally in memory and then sends the packet on to the victim (Steps 2-6 in Figure 3).

The victim router is configured to examine incoming packets for IP options that could be associated with a `SRC_COOKIE`. Using the `libpcap` library, the router identifies cookies and associates them with flow data, including the source IP address, the transport layer protocol, and the transport layer source port. To test our system, we implemented an anomaly detection program that runs on the victim router. The program monitors packets to identify any indications the traffic is unwanted, such as ICMP errors or TCP reset packets, and filters the flow when a threshold is reached (e.g., 20 rejections/second in our experiments). Prior work analyzing TCP reset rates by Arlitt et al. [35] and Bilal et al. [36] shows some variation in reset rates based on client activity, but a high volume of resets when bandwidth is saturated is a potential attack indicator.

While the victim router's downstream bandwidth may be saturated during a DDoS attack, the victim router can control the upstream channel and prioritize attack reports over any other messages. Accordingly, the victim router can locally filter the unwanted flow and send a request to the victim controller to filter the traffic at the source, if possible. To do so, the victim router supplies the controller with the `SRC_COOKIE` that it previously logged, if any, along with details about the flow, including addresses, ports, transport protocol.

Upon receiving a filtering request, the victim controller crafts an UDP packet with the pawn's IP address as the destination and a globally-known filter service port as the destination port. In the UDP packet, the victim controller provides the pawn's IP address and port (which may be empty for `ICMP`), the victim's IP address, the transport layer protocol, and the `SRC_COOKIE`. The victim controller also generates and includes a random nonce value, the `DST_COOKIE`, that can be used to acknowledge the filter request. The victim router caches this information in a local database and then sends the UDP packet towards the pawn.

The pawn router monitors traffic on its globally-known filter request port. If it receives a request, it determines if the contained `SRC_COOKIE` is valid and matches the flow, and if so, elevates it to the controller, as shown in Step 4 in Figure 5. The pawn controller again verifies the request, and if authentic, sends an OpenFlow `FlowMod` message to the pawn router to deny traffic associated with the flow's IP addresses, ports, and protocol. The pawn controller then sends a message to the victim controller to acknowledge the request, as shown in Steps 7 and 8 in Figure 5. Once the pawn router applies the `FlowMod` rule, the attack traffic is filtered in the pawn's LAN.

VI. Evaluation

We evaluate our system from both a security effectiveness and a performance perspective. From a security standpoint, the approach must rapidly respond to attack reports in a manner that ensures the reports are authentic. From a performance perspective, the approach must minimize overheads and ensure it can scale in the event of an attack.

A. Security Effectiveness

We begin by examining the size of reporting packets that are sent from the victim router to the victim controller. Each report is 100 bytes, of which 58 bytes is the payload of the filtering request. Accordingly, even a 5Mbps upload bandwidth capacity would allow a victim router to report roughly $6,250$ unique malicious flows per second. This would allow a victim to identify and report a large number of compromised machines while a DDoS attack is ongoing.

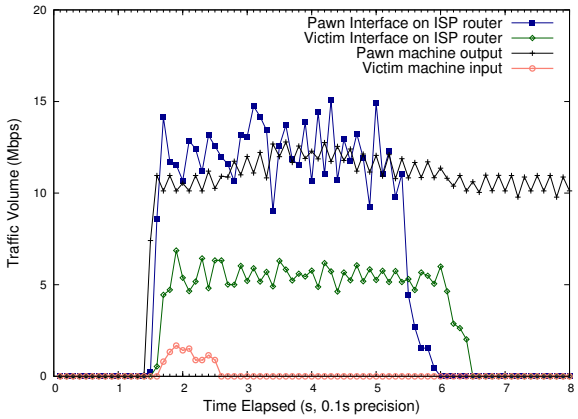

Fig. 6. When five pawn systems simultaneously DDoS a victim, the traffic spikes and saturates the link between the victim router and its ISP. When pawn routers and controllers respond to source filtering requests, the attack traffic quickly drops off, despite the continued transmissions from the pawn host.

We next empirically examine the potential attack traffic reduction possible if pawns perform source filtering. For our pawn host, we use a Thinkpad S3 machine with 4 cores and 8 GBytes RAM. We configure the pawn with five IP aliases to simulate multiple attackers that must be filtered individually. On the pawn, we use the `hping3` tool to launch a TCP SYN flood attack at the public IP address of the victim router. Since the victim router does not run public services, the router OS will send back a TCP RST packet in response. We use a separate `hping3` process for each IP alias on the pawn and we set each `hping3` process to send 2 Mbps of attack traffic in order to saturate the bottleneck link between the ISP router and the victim router.

We monitor the traffic volume at multiple points between the pawn and the victim router, as represented by the black dots in Figure 1. These measurement points include a sample on the pawn's network interface, the pawn-connected interface on the ISP router, the victim-connected interface on the ISP router, and the victim's interface.

In Figure 6, we show the results of our filtering experiments when all five pawns launch an attack simultaneously. The pawn traffic starts around the 1.5 second mark, averaging around 10Mbps (represented by the black line). A queue builds at the pawn router, leading to bursty results (represented by the blue line). The ISP router constrains the throughput of the traffic, which averages around 5 Mbps, as expected (represented by the green line). The victim router quickly notices the attack and applies filters locally, preventing the victim from ever experiencing the full bandwidth of the attack (represented by the red line). The victim router

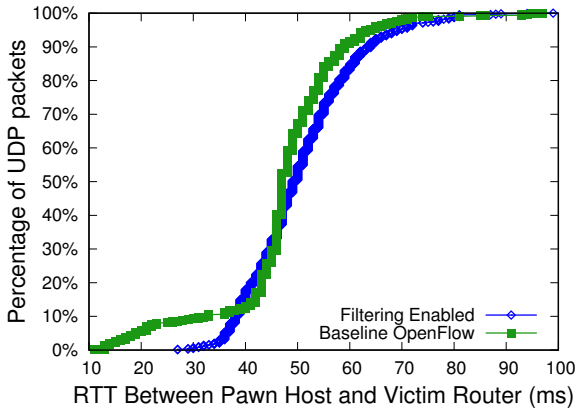

Fig. 7. The delay between a skeleton OpenFlow implementation and our distributed reporting averages only 3ms across our 1000 trials. This overhead is less than 10% of the overhead associated with basic flow elevation.

elevates the requests to the victim controller, which starts implementing the filters around 5.5 seconds into the experiment, resulting in a rapid drop of traffic volume at the ISP router on both the victim and attacker interfaces, despite steady out-bound traffic at the pawn host. The outbound queue causes some residual traffic to be sent to the victim router around the 6 second mark.

B. Performance evaluation

We measure the overhead introduced by our system from different viewpoints. We first measure how much overhead is introduced by inserting cookies into the first packet of each flow. On the pawn machine, we create a UDP client program that sends UDP packets with different source ports to ensure that each packet represents a new flow that is elevated by the router's OVS agent to the pawn's controller. We implement a UDP server at the victim router to echo back each requested packet. We then measure the round trip time (RTT) of 1,000 UDP packets when both the pawn router and pawn controller are processing packets. We configure the controller to create `PacketOut` messages immediately upon receiving a packet without any packet processing. Accordingly, the RTT includes the bidirectional propagation time plus two packet elevations to the attacker controller since both the outbound packet and its reply are elevated to the controller. In Figure 7, the RTT is represented by the green line. We see that 90% of packets have an RTT of less than 60 milliseconds. We compare this with the OVS router and OpenFlow running our distributed reporting approach, depicted by the blue line in Figure 7. These results show minor differences between the two OpenFlow implementations, with the version running our code taking approximately 3 milliseconds longer than a skeleton OpenFlow elevation. While OpenFlow naturally introduces delays with reactive flow elevation, the delays are only on the first packet in a flow and can be accommodated in production environments. The small differences in RTT introduced by our approach are dwarfed by the base OpenFlow delay and are unlikely to be apparent to an end-user.

VII. Conclusion

This work introduced a distributed reporting approach that allows third-party service providers to detect and filter DDoS traffic for residential users. This model incentivizes each stakeholder in the system: residential users save bandwidth when under attack and learn about compromised systems quickly, while security service providers respond rapidly to events and provide notification to residential customers. With incremental deployment and immediate incentives, collaborative filtering can motivate users to adopt a platform for security providers to host residential SDN security applications.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1651540.

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, M. Damian, C. Seaman, N. Sullivan, K. Tomas, and Y. Zhou, "Understanding the mirai botnet," in *USENIX Security Symposium (USENIX Security)*, 2017, pp. 1093–1110.

[2] K. Egevang and P. Francis, "The IP network address translator (NAT)," RFC 1631, May, Tech. Rep., 1994.

[3] F. lab, "The myth of network address translation as security," https://www.f5.com/services/resources/white-papers/the-myth-of-network-address-translation-as-security, 2016.

[4] S. Stamm, Z. Ramzan, and M. Jakobsson, "Drive-by pharming," in *International Conference on Information and Communications Security*. Springer, 2007, pp. 495–506.

[5] S. Zanero, "Wireless malware propagation: A reality check," *IEEE Security & Privacy*, vol. 7, no. 5, pp. 70–74, 2009.

[6] N. Feamster, "Outsourcing home network security," in *ACM SIGCOMM Workshop on Home Networks*. ACM, 2010, pp. 37–42.

[7] S. Grover, M. S. Park, S. Sundaresan, S. Burnett, H. Kim, B. Ravi, and N. Feamster, "Peeking behind the NAT: an empirical study of home networks," in *Internet Measurement Conference*. ACM, 2013, pp. 377–390.

[8] C. R. Taylor, T. Guo, C. A. Shue, and M. E. Najd, "On the feasibility of cloud-based SDN controllers for residential networks," in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2017, pp. 1–6.

[9] K. J. Argyraki and D. R. Cheriton, "Active Internet traffic filtering: Real-time response to Denial-of-Service attacks." in *USENIX Annual Technical Conference*, vol. 38, 2005.

[10] L. Qihoo 360 Technology Co., "Insight into global DDoS threat landscape," https://ddosmon.net/insight/, 2019.

[11] W. Chang, A. Mohaisen, A. Wang, and S. Chen, "Measuring botnets in the wild: Some new trends," in *ACM Symposium on Information, Computer and Communications Security*. ACM, 2015, pp. 645–650.

[12] S. Jin and D. S. Yeung, "A covariance analysis model for DDoS attack detection," *IEEE International Conference on Communications*, vol. 4, pp. 1882–1886, 2004.

[13] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *DARPA Information Survivability Conference and Exposition*, vol. 1. IEEE, 2003, pp. 303–314.

[14] S. M. T. Nezhad, M. Nazari, and E. A. Gharavol, "A novel DoS and DDoS attacks detection algorithm using ARIMA time series model and chaotic system in computer networks," *IEEE Communications Letters*, vol. 20, no. 4, pp. 700–703, 2016.

[15] S. Yu, W. Zhou, and R. Doss, "Information theory based detection against network behavior mimicking DDoS attacks," *IEEE Communications Letters*, vol. 12, no. 4, pp. 318–321, 2008.

[16] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," *arXiv preprint arXiv:1611.07400*, 2016.

[17] J. David and C. Thomas, "DDoS attack detection using fast entropy approach on flow-based network traffic," *Procedia Computer Science*, vol. 50, pp. 30–36, 2015.

[18] O. Haq, Z. Abaid, N. Bhatti, Z. Ahmed, and A. Syed, "SDN-inspired, real-time botnet detection and flow-blocking at ISP and enterprise-level," *IEEE International Conference on Communications (ICC)*, pp. 5278–5283, 2015.

[19] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "Delving into Internet DDoS attacks by botnets: characterization and analysis," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 6, pp. 2843–2855, 2018.

[20] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from distributed denial of service attacks using history-based IP filtering," in *IEEE International Conference on Communications, 2003.*, vol. 1, 2003, pp. 482–486.

[21] R. Pandrangi, "IP prioritization and scoring system for DDoS detection and mitigation," Jan. 13 2015, uS Patent 8,935,785.

[22] C. Buragohain and N. Medhi, "Flowtrapp: An SDN based architecture for DDoS attack detection and mitigation in data centers," in *International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE, 2016, pp. 519–524.

[23] F. Rebecchi, J. Boite, P.-A. Nardin, M. Bouet, and V. Conan, "Traffic monitoring and DDoS detection using stateful SDN," in *IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2017, pp. 1–2.

[24] K. Weniger, J. Bachmann, and R. Hakenbert, "Method for mitigating denial of service attacks against a home," Dec. 10 2009, uS Patent App. 12/514,999.

[25] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 3, pp. 62–73, 2002.

[26] Wikipedia, "Openflow," https://en.wikipedia.org/wiki/OpenFlow, 2019.

[27] C. Ryan, "Computer and Internet use in the United States: 2016," https://www.census.gov/content/dam/Census/library/publications/2018/acs/ACS-39.pdf, 2018.

[28] P. Associates, "76% of North American broadband households use Wi-Fi as their primary connection technology," https://www.parksassociates.com/blog/article/pr-01242018, 2018.

[29] L. F. C. Projects, "Open vSwitch," https://www.openvswitch.org, 2016.

[30] R. Merchant, "New study from dashlane reveals extremes people will go to for online protection, shortcomings of steps they take to secure themselves," 2016, https://blog.dashlane.com/study-reveals-extremes-people-go-online-protection/.

[31] O. Kulyk, S. Neumann, J. Budurushi, and M. Volkamer, "Nothing comes for free: How much usability can you sacrifice for security?" *IEEE Security & Privacy*, 2017.

[32] N. George, "Upload vs. download speed: what's the difference?" https://www.allconnect.com/blog/difference-between-download-upload-internet-speeds/, 2019.

[33] O. Project, "OpenWrt wireless freedom," https://openwrt.org, 2019.

[34] MurphyMc, "The POX network software platform," https://github.com/noxrepo/pox, 2017.

[35] M. Arlitt and C. Williamson, "An analysis of TCP reset behaviour on the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 1, pp. 37–44, 2005.

[36] A. A. Bilal and Umer, "Study of abnormal TCP/HTTP connection," *Digitala Vetenskapliga Arkivet*, 2011.