HogMap: Using SDNs to Incentivize Collaborative Security Monitoring

Xiang Pan
xiangpan2011@u.northwestern.edu

Vinod Yegneswaran
vinod@csl.sri.com

Yan Chen
ychen@northwestern.edu

Phillip Porras
porras@csl.sri.com

Seungwon Shin
claude@kaist.ac.kr

ABSTRACT

Cyber Threat Intelligence (CTI) sharing facilitates a comprehensive understanding of adversary activity and enables enterprise networks to prioritize their cyber defense technologies. To that end, we introduce HogMap, a novel software-defined infrastructure that simplifies and incentivizes collaborative measurement and monitoring of cyber-threat activity. HogMap proposes to transform the cyber-threat monitoring landscape by integrating several novel SDN-enabled capabilities: (i) intelligent in-place filtering of malicious traffic, (ii) dynamic migration of interesting and extraordinary traffic and (iii) a software-defined marketplace where various parties can opportunistically *subscribe to* and *publish* cyber-threat intelligence services in a flexible manner. We present the architectural vision and summarize our preliminary experience in developing and operating an SDN-based HoneyGrid, which spans three enterprises and implements several of the enabling capabilities (e.g., traffic filtering, traffic forwarding and connection migration). We find that SDN technologies greatly simplify the design and deployment of such globally distributed and elastic HoneyGrids.

1 Introduction

Effective and timely cyber threat intelligence (CTI) sharing is essential in the fight against sophisticated cyber-criminals. However, the state of the art in CTI sharing leaves much to be desired: it is a potpourri of community mailing lists or a set of ad-hoc automation solutions. We posit that the development of automated solutions to collaborative threat monitoring and exchange is principally constrained by two factors: technical complexity and lack of coercive incentives.

Novel and exciting capabilities enabled by SDNs include *physical separation* of the control plane from the data plane, *network programmability*, and *network function virtualization* which allows for dynamic network reconfiguration to support host mobility [1], in-network load balancing [2], and dynamic threat mitigation [3]. We make the case that such capabilities could mitigate the technical complexity of collaborative monitoring by simplifying the seamless integration of distributed threat monitoring sensors.

We architect our collaborative threat monitoring framework around dark address space monitors (network telescopes [4] and honeynets [5]) which are well-established security tools commonly used to gather CTI. While telescopes have limited fidelity due to the limitations of passive monitoring, effective honeynet deployments entail maintaining a constantly evolving repertoire of application responders and a menagerie of operating systems and applications which run at varying patch levels and becomes a constant operational challenge. In addition, most honeynets and telescopes suffer from a tunnel-vision syndrome that stems from monitoring one or a small number of contiguous network blocks. However, targeted scanning by sophisticated attackers implies that the choice of monitoring location is important. Prior works, such as highly predictive blacklists [6], have demonstrated that one gets better blacklists by prioritizing networks that are most similar to one's own. While the benefits of collaborative monitoring are proven, free and open data sharing approaches are straddled by the early adopter incentive problem: there is no benefit to being the first to publicly share data. In contrast, adversaries have become quite adept at incorporating technology advancements and incentivizing collaborative malware propagation through affiliate programs [7, 8] and anonymous currencies [9].

Therefore, an ideal CTI collection framework should achieve the following three design goals:

- *Modular Design*: Darknet provision, honeyfarm management and application development should be decoupled. The framework should support dynamic orchestration of globally distributed darknets and honeyfarms to avoid tunnel-vision syndrome and optimize resource utilization.
- *Incentivized Participation*: Darknet, honeyfarm and application providers should be incentivized to collaborate.
- *High Scalability*: The framework should enable large-scale measurements by intelligently and actively responding to a large volume of traffic with limited resources.

To overcome the existing gridlock in the development of threat intelligence collection services, we design and implement a scalable SDN-based HoneyGrid that decouples network telescopes and honeyfarms, to enable multiple HoneyGrids to dynamically collaborate with each other for higher resource utilization rates and global attack visibility. To incentivize more HoneyGrids and security application providers to collaborate, we present HogMap, a flexible, open, and collaborative marketplace.

2 Conceptual Overview

Hogmap is a marketplace to facilitate incentivized collaboration between network operators with reserved (unused) address space, honeynet service providers, and security application providers who would benefit from access to malicious traffic and threat intelligence across a widely-diverse range of addresses. In building our HoneyGrid marketplace, we embrace the paradigm of SDNs and programmable network switches to facilitate the seamless exchange of global attack activity across distributed participants. Specifically, for the first time, we use SDNs to realize concepts of attack flow orchestration and just-in-time honeypot migration that is informed through real-time observation of an adversary's behavior. Our objective is to enable dynamic threat intelligence collaborations and design analysis capabilities that produce a unified threat surveillance contact surface, scalable to millions of IP addresses.

SDN-NFVSec'16, March 11 2016, New Orleans, LA, USA
© 2016 ACM. ISBN 978-1-4503-4078-6/16/03…$15.00
DOI: http://dx.doi.org/10.1145/2876019.2876023

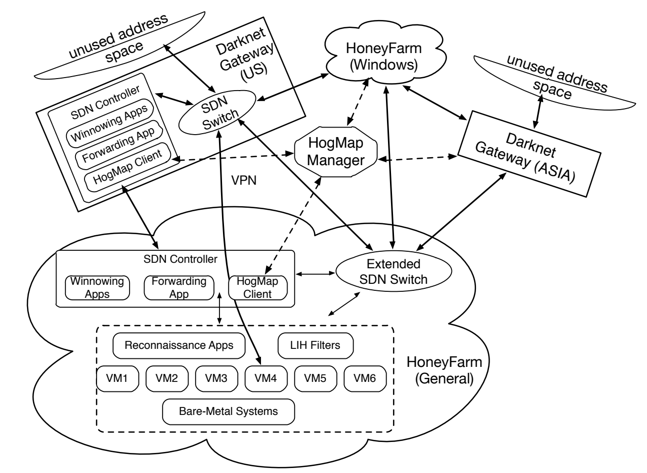


Figure 1: Architectural Overview of HogMap

Service Name / Provider	Description
Attack(er) as a Service (AaaS) / ASP	Real-time information about attacks or attackers targeting certain services or from specific geo- or netblock- locations are advertised.
BlackList as a Service (BLaS) / BLSP	Specialized blacklists services, such as highly predictive blacklists (HPBs) [6] are advertised.
Darknet as a Service (DaaS) / DSP	All or a subset of traffic that is sent to an unused address block are advertised.
Response as a Service (RaaS) / RSP	Mitigation responses (quarantine, migrate to honeynet, fishbowl, etc.), triggered by CTI gathered from HogMap and implemented as SDN applications, could also be advertised.
Honeyfarm as a Service (HaaS) / HSP	Honeyfarms that are comprised of VMs running various operating systems, with different patch levels and different containment policies or application emulators are advertised.
Monitoring as a Service (MaaS) / MSP	Extended monitoring services, e.g., long-term tracking of IRC channels [12], P2P botnet communications etc. are advertised.
Traffic Analysis as a Service (TanaaS) / TASP	Different levels of traffic analysis services, such as real-time intrusion detection, offline deep packet inspection, binary analysis services, etc. are advertised.
Traffic as a Service (TaaS) / TSP	Packet trace and forensic information collected at other HoneyGrids are advertised.

Table 1: Potential Set of HogMap Services and Providers

Figure 1 shows an example of HogMap infrastructure in which a HoneyGrid is composed of two darknet gateways and honeyfarms, owned by different darknet and honeyfarm providers. Through a bidding process coordinated by *HogMap Manager*, the two darknet gateways "outsource" each connection to the chosen honeyfarm for response.

The use of SDN simplifies the deployment of HogMap by providing a common forwarding substrate over which traffic redirection applications may be run. The software running at the gateways of the providers and subscribers may be directly updated by updating the marketplace apps. In addition, various traffic winnowing strategies, that we describe below, may be directly delivered as SDN applications. Realizing similar functionality in traditional networks would involve reimplementing such gateway or proxy forwarding, in custom software [10, 11], and that would be much less scalable.

2.1 SDN-Enabled HoneyGrid MarketPlace

The marketplace is a collection of *a la carte* services that the HoneyGrid participants can advertise and subscribe to. The centralized HogMap manager coordinates dynamic services to foster collaborative surveillance, data exchange and threat mitigation capabilities that would be difficult to enable independently.

HogMap adopts SDN technology to simplify marketplace coordination across different parties: to participate in HogMap, various providers with diverse network architectures only need to be equipped with an OF switch gateway and flexibility to deploy HogMap-certified SDN apps. These apps can then enable the provider to participate in various services and perform just-in-time actions to forward traffic without manual configuration. We discuss how HogMap marketplace works by presenting examples scenarios in Section 3.

A key driving force behind HogMap is the symbiotic nature of the threat monitoring ecosystem, i.e., service subscribers and providers benefit from HogMap. Subscribers benefit from access to heterogeneous services from distributed providers, e.g., collection of physically distributed honeyfarms, each specializing in a different area such as SCADA emulation, Mac Networks, Windows enterprise emulation, satellite telemetry systems and web server vulnerability emulation. These services enable subscribers to achieve custom response orchestration and global visibility; two goals traditional honeynets are unable to realize. Service providers gain access to both data and revenue, helping providers improve their CTI and refine mitigation systems. We identify a potential set of HogMap services [1] in Table 1.

For pricing, HogMap adopts a *laissez faire* approach that is inspired by MoB [13] and eBay [14] with no regulatory control. However, open market economics theory dictates that most traders will ultimately price their services based on prevailing competitive forces. Two critical components of such a marketplace include billing and reputation tracking, which can either be built into the system or relegated to third party providers. We leave this for future work.

2.2 SDN-Enabled Traffic Winnowing

We describe below four traffic winnowing strategies that have been implemented using SDNs to improve scalability.

1. Delayed Connection Migration: Significant honeynet VM cycles are wasted on attempts to service scanners, i.e., source IP addresses that send no more than a single SYN packet. Delayed Connection Migration (DCM) [15] is a strategy that tries to delay the allocation of a dedicated VM instance until the source has demonstrated that it is not just a scanner. We implement DCM by extending OF switches.

2. Dynamic Traffic Filter Scheme: Not all traffic merits a response. If a source exhibits certain activity in one connection, the same kind of activity will be seen on many other connections initiated by the same source. However, proposing an universal filter for all sources is difficult. For example, many sources possess the same degree of affinity to all monitored IP addresses, but there are still some active sources that carry different exploits for different services. To efficiently trade off traffic reduction and information loss, the traffic filter should be dynamically updated with more specific rules. We define a flexible traffic filter scheme that runs as an SDN controller app and supports multiple filtering strategies.

3. Intelligent HIH Management: When an HIH is allocated for a connection, a timeout specifies the period of time that this connection can occupy the HIH. A small timeout might terminate an attacker's ongoing activities; a large one might lead to low HIH utilization rate, since an HIH might still be reserved for a connection even if attacker has stopped exploiting activity. Another traffic winnowing strategy is to proactively remove sessions by preemptively recycling VM instances as soon as the HoneyGrid is able to identify the adversarial malware family or detect the termination of the attack.

In addition, we design a capability to dynamically modify priorities of HIH allocation based on history of sources: sources with a history of exploiting unknown vulnerabilities would have higher priorities than those repeatedly attempting well-known exploits.

4. LIH Filters and Session Migration: To further reduce the volume of traffic reflected to the honeyfarm, one might use low-interaction honeypots (LIHs) such as Dionaea [16], Honeyd [17] or iSink [18]

[1]This list is not intended to be complete, rather its evolutionary and meant to highlight the breadth of services that could be supported

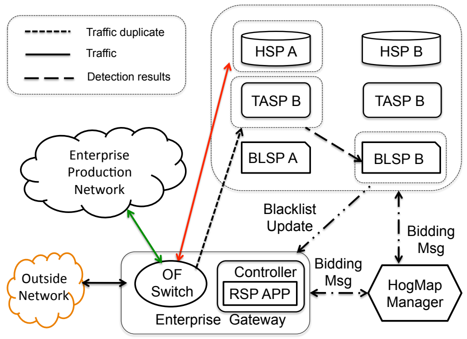


Figure 2: An enterprise subscribes to RaaS to better protect its production network

as a pre-filter before allocating high-interaction honeypots (HIHs) to traffic because our evaluation results show that LIHs are sufficient for a large portion of traffic. For example, in our operational HoneyGrid, we found that more than 90% of Telnet connections stopped at the Telnet authentication phase, which most of the Telnet LIHs can fully simulate; allocating an HIH to each of these connections is wasteful. Since we are not able to determine if a connection requires an HIH when one only sees the first packet, we allocate LIHs for these connections. If the LIH cannot handle the connection, we then migrate the session to an HIH instance in real time.

3 Case Study

We discuss two HogMap use case scenarios. The first one shows how bidding process works. The second one illustrates how multiple providers collaboratively service a subscriber using SDN techniques.

Scenario 1: A US research group intends to monitor and analyze web security trends in Asia, but they don't have Asian darknet address space. Therefore, the group sends a query to the HogMap manager to subscribe to TaaS with its requirements for DSP and HSP. Having received the request, the manager sends bid requests to all the qualified providers. Each provider returns a bid response including pricing and detailed capacities. The HogMap manager relays the bid responses to the subscriber, which then selects the bid winners (e.g., *DSP A* and *HSP B*) on the basis of price and reputations, and returns winners to the manager, which informs DSP A and HSP B of subscriber's selection. Then, HogMap client apps on both providers authenticate each other through digital certificates generated by HogMap manager during registration and create a temporary secure channel to negotiate traffic forwarding parameters such as the responding services and maximum outgoing requests. When the temporary channel is initiated, DSP A forwards a copy of the traffic to subscriber's address using GRE channel.

Scenario 2: Figure 2 illustrates a scenario in which an enterprise network subscribes to RaaS to better protect its production network. RaaS is collaboratively served by a BLSP, HSP, TASP. The bidding process is the same as in Scenario 1 and the subscriber selects one from each provider group. Based on the subscriber's request details and the chosen providers, the HogMap manager customizes an SDN mitigation response app and delivers it to the subscriber. The subscriber is required to run an SDN gateway switch upon which runs the mitigation response app that forwards incoming traffic. Traffic identified as malicious by the response app is then forwarded to selected HSP; the remaining non-suspicious traffic will be forwarded to the enterprise's production network.

In addition, the mitigation response app also instructs the gateway switch to send a copy of the honeyfarm traffic to a TASP for deep packet inspection. The TASP will generate situational awareness reports [5] for the enterprise network's administrators everyday. Moreover, the detection results will also be directly sent to BLSP to update its blacklist and to HSP for better allocation of honeypots (see Section 2.2). If a BLSP updates its database, it will directly send a request to the mitigation response app through the app's REST API interface. Because the participants are all SDN-enabled, the subscriber is able to dynamically replace any of the providers for lower price or better performance without stopping the service.

Both scenarios illustrate how participants are incentivized to join HogMap: subscribers can get services with smaller budgets and better performance; providers not only get paid, but also get access to real-world data that would be otherwise hard to obtain for product improvement.

4 System Implementation and Evaluation

We have implemented a proof-of-concept SDN-enabled HoneyGrid, including one darknet gateway (hosting a contiguous /17 unused address block); one local honeyfarm; two globally distributed honeyfarms; and one traffic analysis server which stores traffic and runs analysis scripts every six hours to analyze the data and generate situational awareness reports [5].

4.1 Honeypot Allocation

We implemented the four strategies discussed in Section 2.2 to reduce traffic volume and fully utilize HIH resources.

1. Delayed Connection Migration: We extended the OpenFlow reference software switch [19] with an additional action (DELAYED_CONTROLLER) to implement the delayed connection migration function [15]. The extended switch proxies TCP handshakes, and once the handshake is complete, forwards packets to the controller's honeypot allocation app. The honeypot allocation app then allocates honeypots and directs the switch to migrate the connection. Once the migration process completes, subsequent packets will be forwarded to the honeypot with *SEQ_MOD* and *ACK_MOD* actions, which are two actions added by our extended switch to modify a TCP packet's SEQ and ACK fields. The honeypot allocation app selects flows on which to enforce delayed connection migration by installing new flow rules with the DELAYED_CONTROLLER action; matching flows will be subject to delayed connection migration.

We measured the volume of scan traffic on a honeynet that responds to all TCP connections. The results are shown in the first row of Figure 5c; among all the responded connections, scanning traffic accounts for 34.5%. Delayed connection migration can prevent HoneyGrid from allocating honeypots for them.

2. Dynamic Traffic Filtering Scheme: We implemented a filter scheme on honeypot allocation apps that maintains a filter table such that multiple filters can coexist. Each row of the table represents a filter instance with its parameters. For each connection, the table first finds all filters with matching scope and then applies the one with the highest priority.

By default, we employed a *one-source-one-destination-per-honey-service (OSODPHS)* filter: for each honey service that we deployed, a source can only interact with the first destination that it contacts. The honeypot allocation app supports other authorized participants (e.g., a TASP) to dynamically modify filters through the REST API interface [20] provided by the app.

We evaluated the effectiveness of the OSODPHS filter in our operational HoneyGrid. The results are illustrated in Figure 5d; on average, the OSODPHS filter drops 23.1% of connections. Another 21.1% of connections are dropped because our HoneyGrid does not have suitable honey services for those protocols. Only 55.8% of total connections will enter the honeyfarm.

3. Intelligent HIH Management: We first leverage the OF switch's idle and hard timeout feature: for each flow rule, we set a large *hard timeout*, which indicates the maximum amount of time the rule can exist; and a small *idle timeout*, which specifies the amount of time before the rule

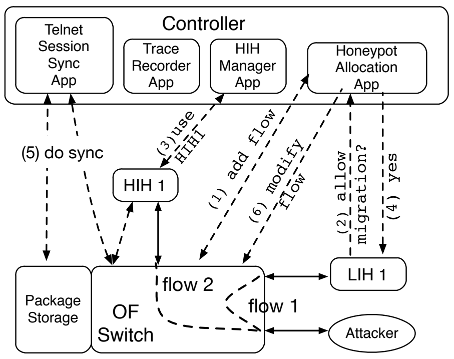


Figure 3: Telnet session migration

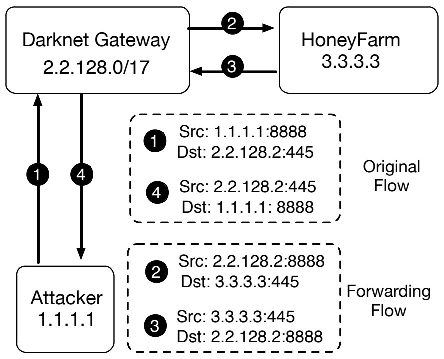

Darknet Gateway 2.2.128.0/17
HoneyFarm 3.3.3.3
Attacker 1.1.1.1
1 Src: 1.1.1.1:8888 Dst: 2.2.128.2:445
4 Src: 2.2.128.2:445 Dst: 1.1.1.1: 8888
Original Flow
2 Src: 2.2.128.2:8888 Dst: 3.3.3.3:445
3 Src: 3.3.3.3:445 Dst: 2.2.128.2:8888
Forwarding Flow

Figure 4: Darknet gateway and honeyfarm forward traffic by modifying IP header

is removed when no traffic is detected. When a timeout event fires, the corresponding rule will be automatically removed and a *flow removed* message will be sent to controller; upon receiving this message, the honeypot allocation app and the HIH manager app update connection status and recycle the HIH for the next connection. In addition, we run custom Bro policy scripts in each HIH to monitor traffic in real time. If the traffic can be attributed to a known malware family, the HIH will immediately send a message to the controller. Upon receipt of this message, the HIH will be recycled for new connections.

The honeypot allocation app supports source priority update by other authorized participants in the marketplace through a REST API interface during runtime. In the current implementation, a source's priority can be set as high, medium or low; medium is the default priority. The honeypot allocation app always tries to allocate HIH for sources with high priority. In the absence of an unallocated HIH, an HIH used by the source with lowest priority will be recycled and allocated to the high-priority source.

We evaluated the effectiveness of VM recycling on a small HIH honeyfarm with eight HIHs running. Figure 5b shows the CDF of 16,307 HIH instance lifetimes. From the figure, we see that for eighty percent of the times, an HIH will run for less than 51 seconds. For less than 10% of the time, an HIH's lifetime is longer than 120 seconds; the longest lifetime is 300 seconds. We are mostly interested in such rare-scenario traffic because these sources are more likely to initiate an unknown attack. Without preemptive VM recycling, every HIH has to run for 300 seconds in order to collect sufficient statistics, a period which would significantly waste HIH resources.

4. LIH Filters and Session Migration: We implemented a session migration mechanism for the Telnet protocol. Figure 3 illustrates how it works. (1): When an attacker initiates a connection to port 23 (Telnet), the honeypot allocation app first allocates a Telnet LIH (LIH1) for it. After attempting several username/password pairs, the attacker successfully logs in and sends the first command. (2): LIH1 sends a request to honeypot allocation app to ask if conducting session migration. (3, 4): Next, the honeypot allocation app chooses an unused Telnet HIH (HIH1) as the target honeypot, notifies LIH1 not to further respond to this session and then instructs Telnet session synchronization app to start synchronizing the session. (5): The packets that might be used for future session synchronizations are stored on the extended OF switch but managed by the trace recorder app. The Telnet session synchronization app fetches this session's previous packets through trace recorder app and synchronizes the session through OF switch. (6): After synchronization, the Telnet session synchronization app informs the honeypot allocation app, which then modifies flow rules to forward the session's subsequent traffic to the port connected to HIH1 with different ACK and SEQ fields through *SEQ_MOD* and *ACK_MOD* actions, which are provided by our extended OF switch. Thus, through this process, the attacker's session is first handled by LIH and then seamlessly handed off to the HIH.

We measured the number of Telnet connections that stopped at the authentication phase. The results are shown in the second row of Figure 5c; on average, there are 246 Telnet connections per hour and 96.3% of them stopped at the authentication phase (our Telnet LIH app will let the attacker pass the authentication phase if he/she either enters the most popular username password pairs, like root/root, or has attempted more than five times). Due to session migration, the HoneyGrid need not allocate HIHs to any of these connections.

4.2 Traffic Forwarding

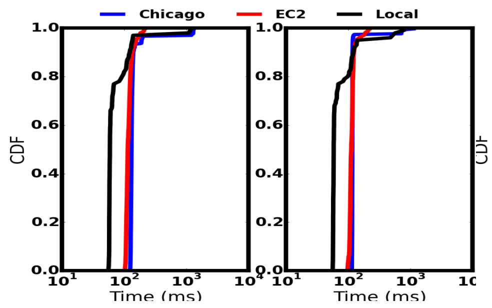
Traffic forwarding apps running on the darknet gateway and honeyfarm forward traffic between each other. We discuss how these work in this section.

1. Packet Modification: We adopt a header modification approach to forward traffic. This approach can be conveniently implemented using OF switches without incurring bandwidth overhead. The traffic forwarding app installs rules to modify an incoming packet's destination address as honeyfarm's public address and its source address as the target IP. Each packet will be mapped from flow *attackerIP*: *attackerPort* → *targetIP*:*targetPort* (the *original flow*), to flow *targetIP*: *attackerPort* → *honeyfarmIP* : *targetPort* (the *forwarding flow*). Figure 4 shows an example: an attacker with IP *1.1.1.1* attempts to exploit port 445 of a darknet address *2.2.128.2* (target IP). This connection is then "outsourced" to a honeyfarm with the public IP *3.3.3.3* for response.

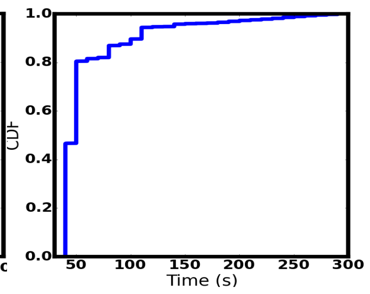
Though this scenario is rare, two original flows could be mapped to the same forwarding flow. Therefore, if a packet forwarding app detects such a conflict, it will add one more action to modify attacker's source port with a different port number to compose a new forwarding flow that has not been used by other original flows at the moment.

2. Attacker IP Notification: Darknet gateway switches modify the packets' IP headers to forward them to honeyfarm. To allow the honeyfarm to discern the attackers' true IP addresses, the traffic forwarding app creates a virtual tunnel for each connection before forwarding traffic (that is inspired by FlowTags [21]): for each new connection, traffic forwarding app on the darknet gateway side emits two setup packets that inform the honeyfarm of the attacker' true IP. Specifically, before installing rules for a new connection, the traffic forwarding app first builds two packets with their IPID fields set as first and second half of the source IP, respectively. We set the ECN fields in the IP headers of these packets as either 1 or 2 to indicate which half of the attacker IP is stored in the IPID field. The ECN fields of non-setup packets are unchanged.

We also propose a negotiation mechanism to prevent setup packets from getting lost: the honeyfarm will send a setup-request packet back to the darknet gateway if it detects a potential setup packet loss; the darknet gateway does not install backward forwarding rules unless it receives a confirmation. This mechanism guarantees that the traffic forwarding



(a) HoneyGrid Forwarding Delay



(b) HIH VM Lifetime

Type	Total Conns (per/hour)	Filtered Conns (per/hour)
General	284,346	97,994 (34.5%)
Telnet	246	237 (96.3%)

(c) Connection & Session Migration

HoneyFarm	Total Conns (per hour)	Percentage (%)
Total	399,972	100
OSODPHS	92,408	23.1
Unknown	84,425	21.1
Responded	223,139	55.8

(d) Total & Responded Connections

Figure 5: HoneyGrid Experimental Evaluation Results

app resends the lost setup packet in a timely manner and a connection will be continued only when the honeyfarm has received the attacker's IP address.

3. Flow Consistency: Inconsistencies may arise due to asynchronous deletion of session rules in the darknet gateway and the honeyfarm. In our implementation, when a rule in the darknet gateway gets deleted, the traffic forwarding app sends a message to the honeypot allocation app running on the honeyfarm, indicating the expired flow, which should be deleted. The honeypot allocation app deletes the specified flow rules and returns a confirmation to the traffic forwarding app. Only when the confirmation has been received, can the traffic forwarding app reuse the forwarding flow. For efficiency, this process is implemented to work asynchronously.

4. HoneyGrid Forwarding Delay: Our HoneyGrid is deployed with one local honeyfarm and two remote honeyfarms located in Chicago and Amazon's EC2 Cloud [22], respectively. We measured the HoneyGrid's forwarding delay by initiating 3000 HTTP requests to the darknet gateway, which then forwarded the traffic to the three honeyfarms for response. Specifically, we measured the initial RTT (i.e., the time it takes from SYN to the SYN/ACK) and non-initial RTT (i.e., the round trip delay after TCP handshake). The initial RTT's CDF is shown in Figure 5a (left), while the non-initial RTT is shown in the right. The 80th percentiles of non-initial RTTs are 117.91 and 120.34 milliseconds for the two remote honeyfarms and 99.34 milliseconds for local honeyfarm. The overhead results primarily from the transmission delay between the darknet gateway and remote honeyfarms. The remote honeyfarms' initial RTTs incur extra tunnel setup overhead; the 80th percentile of initial RTTs for the Chicago and EC2 honeyfarms are 136.89 and 126.37 ms, meaning that the tunnel setup overhead is between 6 and 20 ms. Compared with the overall end-to-end delay, this overhead is negligible and incurred only once for each connection.

5 Discussion

Our proof-of-concept implementation is just a starting point toward a full-fledged HogMap marketplace. Here, we discuss some attacks, limitations and future research directions.

As an SDN-based system, HogMap introduces new attack vectors. First, vulnerabilities in SDN applications and controller may be exploited by attackers. This issue can be addressed by timely update of controller and app software. Moreover, state-of-the-art SDN security techniques [23] can be deployed on HogMap to mitigate the threats. Second, the centralized control plane makes SDN-based systems more vulnerable to DoS attacks. In HogMap, these attacks may be addressed through delayed connection migration.

HogMap providers may come from various organizations with different security capabilities and trust levels. Therefore, in HogMap, different providers are isolated and may only communicate through temporary channels setup by the HogMap manager with limited privileges. For example, an untrusted honey-service provider may not be allowed to open new outbound connections from within its application. Each provider's compromise history affects its HogMap reputation score. However, the effect of a compromised provider will be limited to its network slice and not affect other participants. A future research direction is developing algorithms that automatically enable the HogMap manager detect compromised providers.

Our implemented approach to forward traffic, though efficient, does not encrypt traffic while forwarding, which might cause IDS/IPS deployed in the packet forwarding path to block traffic carrying exploit data. Therefore, we plan to support a complementary VPN tunnel to forward traffic. In addition, we plan to deliver our extended switch as a whitebox switch image, which can then be downloaded and installed on participants' standardized physical OF switch, such as PicOS [24]. Finally, we plan to implement an LIH filter framework that provides APIs for standardized operations, such as controller-app communications. This would enable developers to simply focus on protocol simulation without worrying about migration.

6 Related Work

Historically, network telescopes have been used to successfully measure and characterize denial-of-service attacks [25], Internet worm outbreaks [26], Internet background radiation (IBR) traffic [27], and Internet outages due to natural disasters and censorship events [28]. While these studies were able to opportunistically infer global attack phenomena, from a single large network slice, they are limited in cyber-event detection fidelity due to the passive nature of such measurement strategies. In contrast, active response honeynets suffer from challenges of scale due to the incessant and high-volume nature of IBR traffic. To our knowledge, we are the first and only attempt to integrate SDN capabilities into the operation of a global HoneyGrid. Our current effort is also inspired by prior work on building distributed network telescopes (e.g., Internet Motion Sensor [29]) and efforts to build scalable and intelligent honeyfarms (e.g., GQ [30] and Potemkin [31]), which are considered state of the art in honeynet monitoring. While we share their aspirations for building scalable honeynet monitors, we differ in our marketplace approach to globally distributed threat monitoring. Unlike prior research efforts, we are also less interested in understanding the behavior of self-propagating worms and botnets and more focused on building new and innovative techniques for tracking stealthy and human-driven activities in these ecosystems. Hence, we particularly emphasize the role of filters and low-interaction honeynets as a first-order filter prior to migrating connections to a centralized high-interaction honeyfarm.

7 Conclusion

We designed and implemented an SDN-based HoneyGrid that decouples darknet gateways and honeyfarms, to enable various participants to dynamically collaborate for higher resource utilization rate and global attack visibility. Moreover, to simplify and incentivize collaboration, we propose a software-defined marketplace where various participants can opportunistically subscribe to and publish, CTI services in a flexible manner. The evaluation results demonstrate the utility of SDNs in implementing a range of sophisticated traffic filtering and migration strategies. HogMap's traffic reduction strategies can effectively decrease the number of connections allocated to high-interaction honeypots (by over 90% for some protocols). In addition, connection forwarding delays across the HoneyGrid are quite reasonable and dominated by the propagation delay between the networks.

8 Acknowledgements

This project was partially funded by the National Science Foundation (NSF) (Award Number CNS-1314956) and the Defense Advanced Research Projects Agency (DARPA) under Air Force Research Laboratory (AFRL) contract FA8750-11-C-0249. The views, opinions, and/or findings contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of NSF, DARPA or the Department of Defense. It is approved for public release, distribution unlimited.

9 References

[1] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford, "Live migration of an entire network (and its hosts)," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pp. 109–114, ACM, 2012.

[2] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 1–6, ACM, 2012.

[3] Radware, "Defenseflow? sdn based network ddos, application dos and apt protection." http://opennetsummit.org/archives/mar14/site/pdf/2014/sdn-idol/Radware-SDN-Idol-Proposal.pdf.

[4] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Network telescopes: Technical report," *UCSD Computer Science*.

[5] V. Yegneswaran, P. Barford, and V. Paxson, "Using honeynets for internet situational awareness," in *In Proceedings of the Fourth Workshop on Hot Topics in Networks (HotNets IV*, 2005.

[6] J. Zhang, P. A. Porras, and J. Ullrich, "Highly predictive blacklisting.," in *USENIX Security Symposium*, pp. 107–122, 2008.

[7] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, *et al.*, "Click trajectories: End-to-end analysis of the spam value chain," in *Security and Privacy (SP), 2011 IEEE Symposium on*, pp. 431–446, IEEE, 2011.

[8] C. Kanich, N. Weaver, D. McCoy, T. Halvorson, C. Kreibich, K. Levchenko, V. Paxson, G. M. Voelker, and S. Savage, "Show me the money: Characterizing spam-advertised revenue.," in *USENIX Security Symposium*, pp. 15–15, 2011.

[9] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*, pp. 127–140, ACM, 2013.

[10] P. Inacio, "Honeymole 2.0." https://www.honeynet.org/project/Honeymole.

[11] R. Berthier, J. Vehent, T. Coquelin, and T. K. Lengyel, "Honeybrid: Hybrid honeypot framework." http://honeybrid.sourceforge.net/.

[12] "Shadowserver." https://www.shadowserver.org/wiki/.

[13] R. Chakravorty, S. Agarwal, S. Banerjee, and I. Pratt, "Mob: A mobile bazaar for wide-area wireless services," in *Proceedings of the 11th annual international conference on Mobile computing and networking*, pp. 228–242, ACM, 2005.

[14] "ebay: Electronics, cars, fashion, collectibles, coupons and more online shopping." http://www.ebay.com/.

[15] 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*, pp. 413–424, ACM, 2013.

[16] Dionaea, "Dionaea - catches bugs." http://dionaea.carnivore.it/.

[17] N. Provos, "Developments of the honeyd virtual honeypot." http://www.honeyd.org/.

[18] D. Dagon, C. C. Zou, and W. Lee, "Modeling botnet propagation using time zones.," in *NDSS*, vol. 6, pp. 2–13, 2006.

[19] "Openflow switching reference system." http://archive.openflow.org/wp/downloads/.

[20] "Floodlight rest api." http://www.openflowhub.org/display/floodlightcontroller/Floodlight+REST+API.

[21] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *Proc. USENIX NSDI*, 2014.

[22] "Aws | amazon elastic compute cloud (ec2) - scalable cloud hosting." http://aws.amazon.com/ec2/.

[23] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, "Securing the software-defined network control layer," in *NDSS*, 2015.

[24] "Pica8: White box sdn." http://www.pica8.com/.

[25] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service activity," *ACM Transactions on Computer Systems*.

[26] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Stanford, and N. Weaver, "The spread of the sapphire/slammer worm." http://www.caida.org/publications/papers/2003/sapphire/sapphire.html.

[27] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 27–40, ACM, 2004.

[28] A. Dainotti, K. C. Claffy, M. Russo, C. Squarcella, M. Chiesa, A. PescapÃl', and E. Aben, "Analysis of country-wide internet outages caused by censorship," in *In ACM SIGCOMM conference on Internet measurement*, 2011.

[29] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, D. Watson, *et al.*, "The internet motion sensor-a distributed blackhole monitoring system.," in *NDSS*, 2005.

[30] C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson, "Gq: Practical containment for measuring modern malware systems," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 397–412, ACM, 2011.

[31] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage, "Scalability, fidelity, and containment in the potemkin virtual honeyfarm," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 5, pp. 148–162, 2005.