HoneyV: A Virtualized Honeynet System Based on Network Softwarization

Bahman Rashidi*, Carol Fung*, Kevin W. Hamlen§, Andrzej Kamisiński**
*Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA
{rashidib, cfung}@vcu.edu
§Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA
hamlen@utdallas.edu
**Department of Telecommunications, AGH University of Science and Technology, Kraków, Poland
kamisinski@kt.agh.edu.pl

***Abstract*—Intrusion detection in modern enterprise networks faces challenges due to the increasing large volume of data and insufficient training data for anomaly detections. In this work, we propose a novel network topology for improved intrusion detection through multi-phase data monitoring system. Rather than the all-or-nothing approach to terminate all sessions identified as suspicious, the topology route traffic to different servers replicas with different monitoring intensity level based on their likelihood of attacks. This topology leverages recent advances in software-defined networking (SDN) to dynamically route such sessions into risk-appropriate computing environments. These environments offer enhanced training opportunities intrusion detection systems (IDSes) by exposing data streams that would not have been observable had the session merely been terminated at the first sign of maliciousness. They also afford defenders finer-grained risk management by supporting a continuum of endpoint environments, ranging from *fully trusted*, to *semi-trusted*, to *fully untrusted*, for example.**

I. Introduction

Anomalous activity at the host level (e.g., unusual process control-flows or file access sequences) can be typically identified as potentially symptomatic of attacks. However, retaining and analyzing all features of all incoming traffic is prohibitively expensive in terms of both storage and computational resources. It might not be an option in practice due to limited computation resource. A means of rapidly collecting realistic data and prioritizing it by its potential analysis value is therefore important for achieving acceptable accuracy and false alarm rates in practice.

However, a critical missing piece of this strategy is a means to rapidly and transparently choose what level of monitoring and confinement to apply to each malicious or suspicious session in a large, distributed network in real time; and to revise such choices as the session evolves. Prior work has mainly considered an *all or nothing* approach that assumes that each session is either monitored or unmonitored; little work has considered differentiated levels of monitoring. Likewise, prior work endeavors to store all or most details of all malicious streams; a means to scale the degree of monitoring based on the *suspicious level* of the stream. potentially much more analysis value than sessions from sources that has never been detected to launch attacks in the past. When resources are finite, prioritizing the storage and analysis of the former over the latter has the potential to greatly improve IDS effectiveness.

978-1-5386-3416-5/18/$31.00

In this paper, we propose a novel approach for *fine-grained session monitoring* based on an SDN network, called HoneyV, which leverages automated network softwarization technology for a highly efficient and automated intrusion detection. More specifically, the system equips each VM with different level of monitoring intensity based on the *level of trust* of the sessions that this VM handles. Namely, these VMs are called trusted, semi-trusted, and untrusted. The incoming flows are automatically assigned different levels of trust and directed to servers with different monitoring intensities. For example, an incoming flow with high likelihood of being attacking traffic will be directed to a VM which is equipped with heavy monitoring sensors to handle untrusted sessions, so that malicious activities are more likely to be detected. Traffic from trusted sources are lightly monitored to save computation resource.

The contribution of the paper can be summarized as follows: (1) we propose a novel architecture design to enhance network security using SDN and NFV technologies; (2) we propose a novel approach that provides fine-grained monitoring control for flows with different trust levels; and (3) we use a simulation approach to provide a proof of concept evaluation on the advantage of multi-level monitoring and the performance of the proposed architecture.

II. Related Work

There are some recent works that utilize network softwarization technologies for intrusion detection and honeypotting. SDNIPS [6] is an SDN-based IPS solution that is a full lifecycle solution including detection and prevention in the cloud networks. HonetMix [4] leverages the programmability of SDN to circumvent attackers' detection mechanisms and enables fine-grained data control for Honeynet. VGuard [3] utilizes the flexibility of the NFV framework to prioritize flows to defend against DDoS attacks.

Bohatei [2] is an SDN and NFV-based solution to defence networks against DDoS attacks. It copes with the expensiveness of hardware appliances of the existing solutions. Using

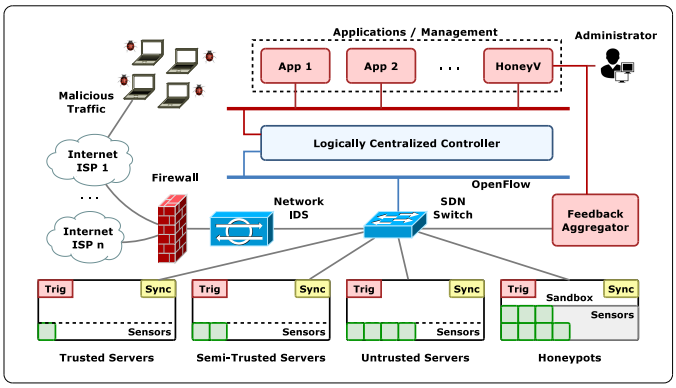


Fig. 1: HoneyV system architecture

its resource manager component, Bohatei assigns available network resources to the defense VMs once suspicious traffic is detected. Because Bohatei is using a limited amount of network resources (to launch VMs), it is ineffective in cases where the incoming attack traffic's strength is heavier than what it can handle. In contrast, in CoFence, in addition to utilizing NFV technology, allows attack victims to receive external help from their collaborators in terms of network resources.

In contrast to these related works, HoneyV focuses on detecting and mitigating misuse attacks rather than DDoS attacks. It does so by leveraging SDN and NFV to efficiently rereoute sessions and reassign workloads to an array of servers with varying levels of privilege, information access, sensors, and security controls.

III. HoneyV System Design

The HoneyV system leverages a typical SDN concept [5]. It assumes that the network control plane is decoupled from the data plane. In the proposed architecture shown in Figure 1, the control plane is represented by the Logically-Centralized Controller and the set of applications, while the data plane includes SDN-compatible network devices that participate in the forwarding of traffic flows, such as switches and security appliances.

A. Architecture overview

The HoneyV system is an architecture of achieveing fine-grained data flow monitoring based on their level of trust-worthiness. The system consists of three parts. The first part is *pre-filtering*, where data flows will have to pass firewall and network-based intrusion prevention system to remove sessions from known sources or with known attack pattern. The flows pass the pre-filtering will be dispatched by the forwarding devices to the servers for processing. The second part is *forwarding*, which consists of an SDN-compatible switch, an SDN controller and a HoneyV app running on the controller. The HoneyV app will determine where to forward each flow. The last but not least part is *servers*, which consists of multiple virtual servers and a virtual feedback aggregator. The servers virtual machines that are equipped with host-based intrusion detection system (sensors) with different levels of monitoring intensity. The servers will process the requests and also monitor the behaviors induced by the flows. The security alerts generated by sensors from all servers are sent to feedback aggregator, which will collect, aggregate and forward the results to the administrator for analysis.

B. Forwarding

The forwarding decision making is a key challenge for HoneyV. The routing decisions will be made by the HoneyV application and executed by the SDN switch. The internal algorithms of decision making rely on data fetched from the following two sources:

- **Logically-Centralized Controller** — information about traffic flows, the current routing scheme, statistics, and network state;
- **Feedback Aggregator** — specific information delivered by servers and related to the observed signs or evidence of malicious activity.

The Feedback Aggregator is a service that acts as a data gateway, collecting relevant information from all servers in the network and passing it to the HoneyV application.

Algorithm 1: Recieving/Dispatching Alg.

Input :
Incoming flow sequence f_i
Output :
$[i, s]$: assigned server to a flow
Notations :
ω_i : weight to flow i
t_i : trust level of flow i
R : the switch's link rate
R_i the data rate of flow i
Q : the queue of flows
//initialize parameters
$\omega_i, t_i \leftarrow 1$
//queuing incoming flows
repeat
  if there is a new flow **then**
    $t_i \leftarrow$*measureTrust*(f_i)
    $R_i \leftarrow$*measureDataRate*(f_i, ω_i)
    $Q_{t/st/u}.enqueue(f_i, t_i.R_i)$
  end if
until (true)
//dispatching queued flows
repeat
  if Q.inNotEmpty() **then**
    $Q.dequeue()$
  end if
until (true)

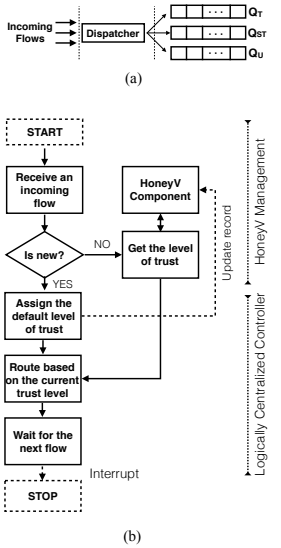


Fig. 2: HoneyV dispatching overview

1) Dispatching algorithm: In order to dispatch incoming data flows to the servers, we have designed a *weighted fair queueing (WFQ)* algorithm. A weighted fair algorithm is a data flow scheduling/queueing algorithm used in different network flow schedulers [1]. Unlike *fair queueing* that tries to give the flows equal shares, at least within the limits of actual demand, in WFQ, the flows receive resources based on an assigned weight (ω_i). The higher weight assigns to a flow, the more resource the flow receives. The key factor that has the highest impact on the amount of resources received by flow is the data rate (R_i). R_i is computed using Equation 1 in which N is the number of incoming flows and R is the data link rate. The data rate is computed upon receiving new incoming flows to the system. Algorithm 1 shows the process of dispatching flows to VMs and allocating resources. There are a certain number of queues depending on the number of servers' types.

In our model, because we have three types of servers, so there are three queues. Depending on flow's trust level the flow is sent to its corresponding queue. Figure 2(a) shows the HoneyV dispatcher and predefined queues and Figure 2(b) shows an overview flowchart of distributing incoming flows to different servers.

It is worth noting that the weights can be defined manually or automatically depending on the model's strategy. As an example of configuring the model automatically, the weights can be defined using *proportionally fairness* principle. Using this principle, we can set the weights to $\omega_i = \frac{1}{c_i}$, where c_i is the cost per data packets of data flow i.

$$R_i = \frac{\omega_i}{\sum_{j=1}^{N} \omega_j} R \tag{1}$$

C. Servers

Traffic flows forwarded through the network are handled by different sets of servers. The servers are equipped with different number of sensors to detect malicious activities. In our proposal, four different types of servers are defined:

- **Trusted Servers** — all servers having access to potentially-sensitive information, to which only highest-confidence, non-malicious flows should be directed;
- **Semi-Trusted Servers** — servers isolated from the most sensitive information, but open to all non-malicious flows and equipped with better monitoring mechanisms;
- **Untrusted Servers** — servers open to all flows and having more sophisticated monitoring and protection mechanisms;
- **Honeypots** — servers with advanced threat detection and analysis components, used mainly to deceive attackers and gather detailed information about their activity.

Figure 3 shows the servers' migration actions spaces. When a anomaly detection alarm is triggered, the server decides to take an action from its action space. As we see in the trusted server's action space (Figure 3(a)), if the signature-based sensors detect an attack, the server redirects the attack flow to the honeypot. Otherwise, if the anomaly-based sensors detect suspicious activities, it will migrate the session to the untrusted server for further diagnosis with full set of sensors. If no detection sensor is triggered on an attack flow, the damage cost induced is $d1$(see the middle rows of the Figure). Similarly, the semi-trusted server also performs migration for suspicious flows which are detected by anomaly sensors but not signature-based sensor. Note that the damage induced by missed attack flows should follow $d_1 > d_2 > d_3$ since highly guarded servers will lead to less damage.

S	A	
x	-	H
-	-	d1
-	x	U

(a)

S	A	
x	-	H
-	-	d2
-	x	U

(b)

S	A	
x	-	H
-	-	d3
-	x	U, d3

(c)

Fig. 3: Servers and the corresponding actions in the case of an attack. The first column is signature-based (S) detection outcome; second volume is anomaly-based (A) detection outcome; and third column is actions: (a) Trusted Server; (b) Semi-Trusted Server; (c) Untrusted Server.

IV. Evaluation

To demonstrate the benefit of the fine-grained flow monitoring system design, we conducted a series of experiments to evaluate the performance of the system using simulation.

TABLE I: Summary of Notations.

Notation	Meaning
T, S, U, H	Trusted, Semi-Trusted, Untrusted, and HoneyPot Servers
d_i	Damage of attack on server i
R_{pf}	Required resource to assess the risk of a flow
TP_s, FP_s	True and False Positive of the Signature-based sensor
TP_a, FP_a	True and False Positive of the Anomaly-based sensor
L, M, H	Incoming flows categories (Low, Medium and High risks)
S, A	The signature-based and anomaly-based sensors

Table I shows the notations we use throughout the evaluation section. Table II shows the configuration we defined for the experiments. Considering our assumptions that servers have various levels of monitoring intensity, we assigned different detection ratios to servers. Due to the risk analysis for the anomaly-based and the signature-based detection sensors process, a certain amount of resources are consumed for each incoming flow. In our experiments, we define the resource units to be an integer value. Depending on the number of sensors that each server has, the amount of resources used per flow is different. For example, a server with more sensors allocates more resources per flow to be analyzed. We define the amount of resources to be allocated per flow (Trusted=1, Semi-Trusted=5, and Untrusted=10). since signature-based detection is known to be accurate once it detects an attack (though they may miss many obfuscated attacks).

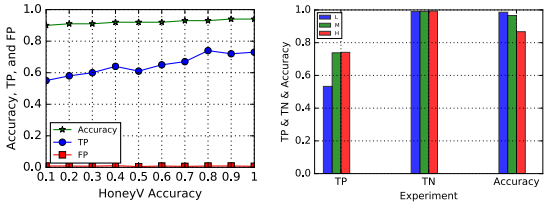
TABLE II: Servers' Configurations

Metric	R_{pf}	TP_s	FP_s	TP_a	FP_a	d
Server						
Trusted	1	.5	0	.1	.01	$d_1 = 10$
Semi-Trusted	5	.7	0	.15	.008	$d_2 = 5$
Untrusted	10	.9	0	.2	.006	$d_3 = 1$

A. Accuracy and Reliability

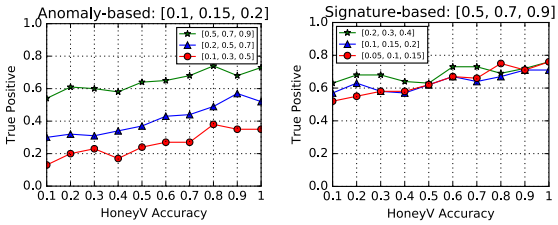
For the accuracy and reliability experiments, we generated a set of flows as our testbed. We decided to categorize them into flows with *Low* (L), *Medium* (M), and *High* (H) with risks 0.1, 0.5, and 0.9, respectively. As an assumption to the experiments, we generated 1000 flows per each risk category and 3000 flows in total. To have a combination of packets in terms of security risks in every group of flows, we generated the flow groups in a way that 1%, 10%, and 50% of the flows were attacks for the low, medium, and high risk flows, respectively. Since we only planned to evaluate the accuracy of the model, we set the servers' resources to be unlimited.

In the first experiment, we evaluated the accuracy of the system in terms of detecting the attacks (malicious flows). We ran the experiment for different levels of accuracy of the HoneyV application. Figure 4(a) shows the accuracy, true positive and false positive ratios of the system. The observed

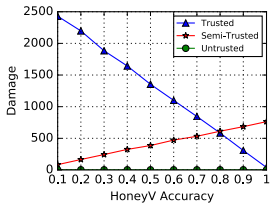


(a) (b)

Fig. 4: The performance of the model in terms of accuracy and true positive ratio: (1) the accuracy, true positive and false positive ratios of the HoneyV model; (b) the accuracy, true positive and true negative ratios of the model per group.



(a) (b)



(c)

Fig. 5: Impact of the risk assessment sensors on the accuracy and true positive ratio of the model: (a) the impact of the signature-based sensor on the true positive ratio; (b) the impact of the anomaly-based sensor on the true positive ratio of the model; (c) the amount of damage on the servers.

detection accuracy and true positive ratio both increase with increasing accuracy of the HoneyV application.

In the second experiment, we evaluated the detection accuracy, true positive, and true negative of the system per each group of incoming flows (Low, Medium, and High). In this experiment, we set the HoneyV application's accuracy to be 100% and because of this, low risk flows were directed to the trusted server, while medium and high risk flows were directed to the semi-trusted server. Figure 4(b) shows the results of this experiment. It can be observed that the true negative ratio is very high and almost the same for all the groups. The true positive ratio for the low risk flows is around 50% but it has a high accuracy. The reason is that only 1% of the low risk flows were attacks, so it does not have a high impact on the accuracy ratio. In contrast, because the number of attack flows in the medium and high risk flows is higher so more attack flows pass the sensors and that has an impact on the accuracy and it is lower than the accuracy for the low risk flows group.

In the third experiment, we evaluated the amount of damage on the servers. Figure 5(c) shows the results of this experiment. The amount of damage on the trusted and semi-trusted servers changed with increasing accuracy of the HoneyV application. We can see that the damage ratio on the trusted server decreases and it is because when the HoneyV has higher accuracy, more benign flows are redirected to the trusted server and the damage goes down. In contrast, the damage ratio on the semi-trusted server increases, because more attack flows (medium and high risk flows groups) are redirected to the server. The damage ratio for the untrusted server stays almost the same and low, because not many flows are transferred to it, and also its sensor's accuracy is high (0.9%) and attack flows can easily be detected.

TABLE III: HoneyV and Uniform Server Comparison

Metric	TP	FP	Accuracy	Resource Usage	Capacity
Server					
Trusted	.51	.01	.9	3000	20000
Semi-Trusted	.74	.01	.94	15000	4000
Untrusted	.92	.01	.98	30000	2000
HoneyV	.73	.01	.94	11240	5340

In the fourth experiment, the impact of the signature-based and anomaly-based sensors' true positive ratio was evaluated. We also ran the experiment under different settings of the HoneyV's accuracy. We fixed the configuration for one type of the sensors and generated three different configurations for the other sensor. This way, we can see the impact of the sensors on the true positive ratio. Figure 5(a) shows the results when the anomaly-based configuration is fixed and the signature-based sensor configuration varies. Figure 5(b) also shows the results when the signature-based sensor's true positive is fixed. It can be observed that the more accurate the sensors, the higher the true positive ratio. We also see that the impact of the signature-based sensor on the true positive ratio is higher than in the case of the anomaly-based one.

Finally, we decided to compare the performance of our model with three uniform servers. Servers' configurations are as described in Table II. We computed the capacity of the model when each model had only 20000 resource units. We define the capacity to be the number of flows that a server can handle at a time. We used the same flows test-bed for all of the four models. Table III shows the comparison results. There are a few important facts about the results. First, in terms of accuracy, our model is close to the semi-trusted uniform server, but our model is more cost-effective and reliable because of less resource usage and higher capacity. Second, compared to the untrusted uniform server, our model is also more cost-effective and handles much higher number of flows.

V. Conclusion

In this paper, we introduced HoneyV: a new honeynet architecture leveraging the flexibility of SDN and NFV to detect and deal with malicious traffic flows in the network. Unlike the existing proposals, HoneyV redirects traffic flows to different types of servers based on the level of trust assigned to each flow, as well as the threat monitoring capabilities of particular servers. The level of trust is adjusted dynamically based on the information about the flow's activity, collected from network servers equipped with various sets of threat sensors. The number, kind, effectiveness, and complexity of sensors installed on network servers may be different for each group of servers, depending on the specific requirements related to safety, privacy, sensitivity of stored information, and other factors.

REFERENCES

[1] Cisco. Qos congestion management (queueing). https://www.cisco.com/c/en/us/tech/quality-of-service-qos/qos-congestion-management-queueing/index.html. Last Visit September 2017.

[2] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey. Bohatei: Flexible and elastic DDoS defense. In *Proc. 24th USENIX Security Sym.*, pages 817–832, 2015.

[3] C. J. Fung and B. McCormick. VGuard: A distributed denial of service attack mitigation method using network function virtualization. In *Proc. 11th IEEE Int. Conf. Network and Service Management (CNSM)*, pages 64–70, 2015.

[4] W. Han, Z. Zhao, A. Doupé, and G.-J. Ahn. HoneyMix: Toward SDN-based intelligent honeynet. In *Proc. ACM Int. Work. Security in Software Defined Networks & Network Function Virtualization*, pages 1–6, 2016.

[5] Open Networking Foundation. Software-defined networking (SDN) definition. https://www.opennetworking.org/sdn-resources/sdn-definition. Retrieved 7/17.

[6] T. Xing, Z. Xiong, D. Huang, and D. Medhi. SDNIPS: Enabling software-defined networking based intrusion prevention system in clouds. In *Proc. 10th Int. Conf. Network and Service Management (CNSM)*, pages 308–311, 2014.