Virtual Network Isolation: Are We There Yet?

Kashyap Thimmaraju, Gábor Rétvári and Stefan Schmid

Multi-tenant Virtual Networks

Amazon Virtual Private Cloud

Data Centre • Virtualization
VMware preps NSX network virtualization for smaller customers
Q1 2019 beats expectations, full-year guidance raised
By Simon Shanwood 1 Jun 2018 at 02:54

Virtual Network
Your private network in the cloud

Benefits
Secure
VMware vSphere provides advanced security features, such as security groups and network access control lists, to create isolated and highly scalable virtual networks.

Simple
You can create a VPC quickly and easily using the AWS Management Console. You can also use at the customer network. The Virtual Private Cloud (VPC) includes virtual private subnets, security groups, and managed route tables.

All the scalability and reliability of AWS
VMware vSphere provides all the scalability and reliability benefits as the rest of the AWS services. Customers can easily scale their virtual networks.

Microsoft Azure

https://aws.amazon.com/vpc/
https://www.theregister.co.uk/2018/06/01/vmware_q1_2019/
Virtual Networks in the Cloud
Virtual Switches

VFP: A Virtual Switch Platform for Host SDN in the Public Cloud

**Author:** Daniel Firestone, Microsoft

**Abstract:**
Many modern scalable cloud networking architectures rely on host networking for implementing SDN features – e.g., forwarding for virtual machines. NFP hosts virtual bridges, switches, and routers. We present the Virtual Forwarding Platform (VFP) – a programmable virtual switch that allows Microsoft Azure, a large public cloud, to provide this facility. We define a network architecture and protocol stack for a programmable virtual switch based on our operational experiences, including insights from multiple generations of VFP. We present the technical details of our design and implementation, including algorithms for performance, and efficient design of interfaces to programmable NFPs. We demonstrate how VFP operates.

**Open Access Media**
Demos are a central component of Open Access to the research presented at our events. Presenters are encouraged to create demos that can be published online to increase exposure. Support NSDI and our commitment to open access.

Presentation Video


https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/firestone
## Virtual Switches (Non-Exhaustive List)

<table>
<thead>
<tr>
<th>Name</th>
<th>Ref.</th>
<th>Year</th>
<th>Emphasis</th>
<th>Co-Location</th>
<th>Kernel</th>
<th>User</th>
<th>Ext. Parsing</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OvS</td>
<td>[22]</td>
<td>2009</td>
<td>Flexibility</td>
<td>✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Baseline</td>
</tr>
<tr>
<td>Cisco NexusV</td>
<td>[36]</td>
<td>2009</td>
<td>Flexibility</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Commercial</td>
</tr>
<tr>
<td>VMware vSwitch</td>
<td>[37]</td>
<td>2009</td>
<td>Centralized control</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Commercial</td>
</tr>
<tr>
<td>Vale</td>
<td>[25]</td>
<td>2012</td>
<td>Performance</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Using HPFP to increase performance</td>
</tr>
<tr>
<td>Hyper-Switch</td>
<td>[24]</td>
<td>2013</td>
<td>Performance</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Fast path in the Xen hypervisor</td>
</tr>
<tr>
<td>MS HyperV-Switch</td>
<td>[18]</td>
<td>2013</td>
<td>Centralized control</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Commercial</td>
</tr>
<tr>
<td>fD.io</td>
<td>[32]</td>
<td>2015</td>
<td>Performance</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Uses Vector Packet Processing, e.g., see Choi et al. [6].</td>
</tr>
<tr>
<td>mSwitch</td>
<td>[12]</td>
<td>2015</td>
<td>Performance</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Using HPFP to increase performance</td>
</tr>
<tr>
<td>BESS</td>
<td>[4]</td>
<td>2015</td>
<td>Programmability, NFV</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Similar to the Click modular router [15].</td>
</tr>
<tr>
<td>PISCES</td>
<td>[29]</td>
<td>2016</td>
<td>Programmability</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Uses a domain specific language to customize parsing.</td>
</tr>
<tr>
<td>Ovs with DPKD</td>
<td>[26]</td>
<td>2016</td>
<td>Performance</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Using HPFP for performance; sw. countermeasures, e.g., canaries and ASLR may not be used.</td>
</tr>
<tr>
<td>ESwitch</td>
<td>[19]</td>
<td>2016</td>
<td>Performance</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Proprietary</td>
</tr>
<tr>
<td>MS VFP</td>
<td>[8]</td>
<td>2017</td>
<td>Performance, flexibility</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Commercial</td>
</tr>
<tr>
<td>Mellanox BlueField</td>
<td>[17]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Runs full fledged OvS on CPU in NIC. Server leased, but provider controls the network.</td>
</tr>
<tr>
<td>Liquid IO</td>
<td>[21]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Runs full fledged OvS on CPU in NIC.</td>
</tr>
<tr>
<td>Stingray</td>
<td>[10]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Runs full fledged OvS on CPU in NIC.</td>
</tr>
<tr>
<td>MS AccelNet</td>
<td>[9]</td>
<td>2018</td>
<td>Performance, flexibility</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔ ✔</td>
<td>Packet processing and flow rules offloaded to an FPGA-based NIC.</td>
</tr>
</tbody>
</table>
Taking Control of SDN-based Cloud Systems via the Data Plane

Kashyap Thimmajarju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann and Stefan Schmid

SOSR’18

trusted components, are commonly not protected with an additional intrusion detection system.

Unified packet parser: Once a virtual switch receives a packet it parses its headers to determine if it already has a matching flow rule. If this is not the case it will forward the packet to an intermediate data path (slow path) that processes the packet further in order to request a new flow table entry. In this step, the virtual switch commonly extracts all header information from the packet, e.g., MPLS and application layer information, before requesting a flow table entry from the controller. Parsing is the switch’s re sponsibility as centralizing this task would not scale. The additional information from higher-level protocols is needed for advanced functionality like load balancing, deep packet inspection (DPI), and non-standard forwarding (see Section 5 for an overview of related technologies using these features in their implementation). However, with protocol parsing in the data plane the virtual switch is as susceptible to security vulnerabilities as any device for the parsed packet. Thus, the attack surface of the data plane increases with any new protocol that is included in parsing.

Untrusted input: Virtual switches are commonly deployed in data centers at the network edge. This implies that virtual switches receive network packets directly from the virtual machines, typically unfiltered, see Section 2. This can be abused by an attacker. She can—via a virtual machine—send arbitrary data to a virtual switch. Indeed, the virtual switch is typically the first data plane component to handle any packet from a VM. This enables attackers to take advantage of data plane vulnerabilities in virtual switches.

Summary: In combination, the above observations demonstrate why data plane attacks are a feasible threat and how they can spread throughout a cloud setup, see Fig. 2. By renting a VM and weaponizing a protocol parsing vulnerability an attacker can start her attack by taking over a single virtual switch (Step 1). Thus, she also takes control of the physical machine on which the virtual switch is running due to hypervisor co-location. Next (Step 2), she can take control of the Host OS where the VM running the network—and in most cases cloud—controller is hosted due to the direct communication channel. From the controller (Step 3), the attacker can leverage the logically centralized design to, e.g., manipulate flow rules to violate essential network security policies (Step 4). Alternatively, the attacker can change other cloud resources, e.g., modify the identity management service or change a boot image for VMs to contain a backdoor.

3.2 Attacker Models for Virtual Switches

With these vulnerabilities and attack surfaces in mind, we revisit existing threat models. We particularly focus on work starting from 2009 when virtual switches emerged into the virtualization market [63]. We find that virtual switches are not appropriately accounted for in existing threat models, which motivates us to subsequently introduce a new attacker model.

Existing threat models: Virtual switches interconnect with several areas of network security research. Data plane, network virtualization, software defined networking (SDN), and the cloud. Therefore, we conducted a qualitative analysis that includes research we identified as relevant to attacker models for virtual switches in the cloud. In the following we elaborate on that.

Qubes OS [78] in general assumes that the networking stack can be compromised. Similarly, Dhawan et al. [26] assumed that the Software Defined Network (SDN) data plane can be compromised. Jero et al. [36] base their assumption on a malicious data plane in an SDN on Pickett’s BlackHat briefing [66] on compromising an SDN hardware switch. A conservative attacker model was assumed by Paladi et al. [55] who employ the DoD-Yao model for network virtualization in a multi-tenant cloud. Crucesaer et al. [28] observed that virtual networking can be attacked in the cloud without a specific attacker model.

Jin et al. [37] accurately described two threats to virtual switches: Virtual switches are co-located with the hypervisor; and guest VMs need to interact with the hypervisor. However, they stopped short of providing a concrete threat model, and underestimated the impact of compromising virtual switches. Indeed at the time, cloud systems were burgeoning. However, only recently Alramish et al. [9] proposed an updated approach to cloud threat modelling wherein the virtual switch was identified as a component of cloud systems that needs to be protected. However, in both cases, the authors overlooked the severity, and multitude of threats that apply to virtual switches.

Motivated by a strong adversary, Camaleses et al. [22] and Karmakar et al. [40] accounted for virtual switches, and the data plane. Similarly Yu et al. [97], Thimmajarju et al. [90], and Feldmann et al. [24] assumed a strong adversarial model,
Compromising the Cloud
Compromising the Cloud

[Diagram showing cloud infrastructure with nodes labeled as 'User Kernel', 'VM', 'Ctrl VM', and 'CONTROLLER NODE']
Compromising the Cloud
Compromising the Cloud
Taking Control of SDN-based Cloud Systems via the Data Plane

Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann and Stefan Schmid

SOSR'18

there is no impact of the user-land protection mechanisms in the fast path, see Fig. 5b.

Throughput Evaluation: For the throughput evaluation we use a constant stream of packets replayed at a specific rate. We opted for small packets to focus on the packets per second (pps) throughput rather than the bytes per second throughput. Indeed, pps throughput indicates performance bottlenecks earlier [34] than bytes per second. As in the latency experiments, we opted to use packets that are 600 byte long. Each experimental run lasts for 1000 seconds and is a specific repeat. Then we reset the system and start

One overview of the results for the slow and fast path throughput measurements are depicted in Figures 5c and 5d resp. In the slow path, packet loss for the vanilla kernel first sets in just after 184 pps, while the experiments on the gresecurity enabled kernel already exhibit packet loss at 144 pps. In the fast path, greseccuit exhibits packet loss from 350k pps whereas the vanilla kernel starts to drop packets at 696k pps. Hence, we note that gresecurity kernel patch does have a measurable impact on the forwarding throughput in the slow and fast path of Orwell. With respect to the user-land security features, we observe an overhead only in the slow path of approximately 4-15%.

Summary: Our measurements demonstrate that user-land mitigations do not have a large impact on Orwell’s forwarding performance. However, gresecurity kernel patches do cause a performance overhead for latency as well as throughput. Given that cloud systems support a variety of workloads, e.g., low latency or high throughput, kernel-based mitigations may or may not be used. However, cloud systems such as

7 DESIGN COUNTERMEASURES

Specific attacks against virtual switches may be prevented by software countermeasures. However, the underlying problems of co-location and a worm-friendly system design remain. Hence, in this section, we present mitigation strategies that detect, isolate, and prevent the spread of attacks via the data plane and thereby the threat surface we identified. We do so not only for cloud based systems and IX but also within the more general context of SDN.

Virtualized/isolated data plane: One essential feature of the identified attack surface is the co-location of data plane and hypervisor (see Section 3). Addressing this problem in OpenStack is non-trivial due to the sheer number of interacting components and possible configurations. A virtualized/non-virtualized, integrated/distributed command/flow/forwarding controllers architecture is one way to design a system with stronger separation is to virtualize the data plane components, thereby decoupling it from the virtualization layer. For virtual switches one example of such a proposal is to shift the position of the virtual switch from the host to a dedicated guest as proposed by Jin et al. [37]. However, the IOAMU of the host must be used to restrict access of the network cards to the network interfaces. Otherwise the physical host and the operating system running there are left vulnerable to direct memory access (DMA) attacks [86]. Such a design reduces the host OS’s Trusted Computing Base (TCB) and, thereby, the attack surface of the virtual switch. We note that Arrakis [59] and IX [12] are promising proposals for HPDPs that would allow for designing such a system. Note, that while Arrakis utilizes the IOAMU, the authors of IX left this for further work.

Furthermore, to reduce the attack surface of hypervisors, Saefer et al. [87] suggest that the hypervisor should disenroll itself from guest VMs, and the VM should receive direct access to the hardware (e.g., NIC). In conjunction with our suggestion of transferring the virtual switch into a virtual machine, the approach of Saefer et al. results in a more secure data plane that can no longer attack the hypervisor.

Control plane communication firewalls: Another method to contain and prevent attacks like the worm is tight firewalling of the control plane. In contrast to "normal" Internet traffic, control plane traffic has characteristics that enable a tighter and more secure firewall design: (i) The control plane traffic volume should be significantly smaller than regular network traffic. (ii) Nodes should only communicate via the controller and not among each other. Hence, there is a central location for the firewall. (iii) On the control channel
Outline

- Motivation
- Security Landscape of Virtual Switches
- Secure Virtual Switch Design
- Conclusion
- Discussion
  - Performance Evaluation
Security Landscape of Virtual Switches
A security analysis of a non exhaustive list of virtual switches (screenshot from the paper).
<table>
<thead>
<tr>
<th>Name</th>
<th>Ref.</th>
<th>Year</th>
<th>Emphasis</th>
<th>Co-Location</th>
<th>Kernel User</th>
<th>Ext. Parsing</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OvS</td>
<td>[22]</td>
<td>2009</td>
<td>Flexibility</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Baseline</td>
</tr>
<tr>
<td>Cisco NexusV</td>
<td>[36]</td>
<td>2009</td>
<td>Flexibility</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>Commercial</td>
</tr>
<tr>
<td>VMware vSwitch</td>
<td>[37]</td>
<td>2009</td>
<td>Centralized control</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>Commercial</td>
</tr>
<tr>
<td>Vale</td>
<td>[25]</td>
<td>2012</td>
<td>Performance</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>Using HPFP to increase performance</td>
</tr>
<tr>
<td>Hyper-Switch</td>
<td>[24]</td>
<td>2013</td>
<td>Performance</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>Fast path in the Xen hypervisor</td>
</tr>
<tr>
<td>MS HyperV-Switch</td>
<td>[18]</td>
<td>2013</td>
<td>Centralized control</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>Commercial</td>
</tr>
<tr>
<td>fDio</td>
<td>[32]</td>
<td>2015</td>
<td>Performance</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>Uses Vector Packet Processing, e.g., see Choi et al. [6].</td>
</tr>
<tr>
<td>mSwitch</td>
<td>[12]</td>
<td>2015</td>
<td>Performance</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>Using HPFP to increase performance</td>
</tr>
<tr>
<td>BESS</td>
<td>[4]</td>
<td>2015</td>
<td>Programmability, NFV</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>Similar to the Click modular router [15].</td>
</tr>
<tr>
<td>PISCES</td>
<td>[29]</td>
<td>2016</td>
<td>Programmability</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>Uses a domain specific language to customize parsing.</td>
</tr>
<tr>
<td>OvS with DPDK</td>
<td>[26]</td>
<td>2016</td>
<td>Performance</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>Using HPFP for performance; sw. countermeasures, e.g., canaries and ASLR may not be used.</td>
</tr>
<tr>
<td>Mellanox BlueField</td>
<td>[17]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>Runs full fledged OvS on CPU in NIC. Server leased, but provider controls the network.</td>
</tr>
<tr>
<td>Liquid IO</td>
<td>[21]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>Runs full fledged OvS on CPU in NIC.</td>
</tr>
<tr>
<td>Stingray</td>
<td>[10]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>Runs full fledged OvS on CPU in NIC.</td>
</tr>
<tr>
<td>MS AccelNet</td>
<td>[9]</td>
<td>2018</td>
<td>Performance, flexibility</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>Packet processing and flow rules offloaded to an FPGA-based NIC.</td>
</tr>
</tbody>
</table>

A security analysis of a non exhaustive list of virtual switches. Preliminary work on isolation.
A security analysis of a non-exhaustive list of virtual switches. New NICs offering isolation.
<table>
<thead>
<tr>
<th>Name</th>
<th>Ref.</th>
<th>Year</th>
<th>Emphasis</th>
<th>Co-Location</th>
<th>Kernel</th>
<th>User</th>
<th>Ext. Parsing</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OvS</td>
<td>[22]</td>
<td>2009</td>
<td>Flexibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Baseline</td>
</tr>
<tr>
<td>Cisco NexusV</td>
<td>[36]</td>
<td>2009</td>
<td>Flexibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Commercial</td>
</tr>
<tr>
<td>VMware vSwitch</td>
<td>[37]</td>
<td>2009</td>
<td>Centralized control</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>Commercial</td>
</tr>
<tr>
<td>Vale</td>
<td>[25]</td>
<td>2012</td>
<td>Performance</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>Using HPFP to increase performance</td>
</tr>
<tr>
<td>Hyper-Switch</td>
<td>[24]</td>
<td>2013</td>
<td>Performance</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Fast path in the Xen hypervisor</td>
</tr>
<tr>
<td>MS Hyper-V-Switch</td>
<td>[18]</td>
<td>2013</td>
<td>Centralized control</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>Commercial</td>
</tr>
<tr>
<td>tDio</td>
<td>[32]</td>
<td>2015</td>
<td>Performance</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>Uses Vector Packet Processing, e.g., see Choi et al. [6].</td>
</tr>
<tr>
<td>mSwitch</td>
<td>[12]</td>
<td>2015</td>
<td>Performance</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>Using HPFP to increase performance</td>
</tr>
<tr>
<td>BESS</td>
<td>[4]</td>
<td>2015</td>
<td>Programmability, NFV</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Similar to the Click modular router [15].</td>
</tr>
<tr>
<td>PISCES</td>
<td>[29]</td>
<td>2016</td>
<td>Programmability</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Uses a domain specific language to customize parsing.</td>
</tr>
<tr>
<td>OvS with DPDK</td>
<td>[26]</td>
<td>2016</td>
<td>Performance</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Using HPFP for performance; sw. countermeasures, e.g., canaries and ASLR may not be used.</td>
</tr>
<tr>
<td>MS VFP</td>
<td>[8]</td>
<td>2017</td>
<td>Performance, flexibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Commercial</td>
</tr>
<tr>
<td>Mellanox BlueField</td>
<td>[17]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Runs full fledged OvS on CPU in NIC. Server leased, but provider controls the network.</td>
</tr>
<tr>
<td>Liquid IO</td>
<td>[21]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Runs full fledged OvS on CPU in NIC.</td>
</tr>
<tr>
<td>Stingray</td>
<td>[10]</td>
<td>2017</td>
<td>CPU offload</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Runs full fledged OvS on CPU in NIC.</td>
</tr>
<tr>
<td>MS AccelNet</td>
<td>[9]</td>
<td>2018</td>
<td>Performance, flexibility</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>Packet processing and flow rules offloaded to an FPGA-based NIC.</td>
</tr>
</tbody>
</table>

A security analysis of a non exhaustive list of virtual switches. Least privilege packet processing.
Position

- Adapt the design principles by Saltzer and Schröder [1] to Virtual Switches
- Secure Design Principles for Virtual Switches
  - Isolate the host from the vSwitch
  - Isolate the tenant vSwitches from each other
  - Least privilege packet processing
  - Reduce the Trusted Computing Base (TCB)
The Secure Virtual Switch Vision
Vision

Existing vSwitch design
Vision

Existing vSwitch design

A Secure vSwitch design
Design Space
All Roads Lead to Rome

Figure credit: moovel Lab,
https://lab.moovel.com/projects/roads-to-rome
Design Space

- Isolate the virtual switch from the Host
  - VMs, Container, Process
- Isolate tenant virtual switches
  - VMs, Containers, Process
- Least privilege packet processing
  - User-space, VM, memory safety
- Reduce the TCB
  - Separation of virtual switch from the Host
  - Limited parsing

An SR-IOV-based design
SR-IOV

Single Root IO Virtualization
SR-IOV

- PCIe standard for IO Virtualization
- Allows a VM direct access to the NIC
- Dedicated registers for the NIC
- Physical Functions
- Virtual Functions

Single Root IO Virtualization
Challenges

- Reachability/Connectivity when the vSwitch is in the VM
  - L2
  - L3
  - ARP
- Drivers
- Resources
- Management
- Security of SR-IOV
First Experience

Servers: Supermicro (Intel Xeon(R) E5-2609 v4 Single Socket 8 Cores)
NICs: SolarFlare, Mellanox, Netronome
First Experience

+ It works!
+ Easy to configure
+ Good documentation
+ Each NIC is different (duh!)
+ Mellanox ftw!
+ Differences in PF and VF
+ Security features offered

- SolarFlare VFIO and DPDK in VM not yet supported
- PF driver not in VM
  - SolarFlare can do PF-IOV in 8 VMs only
- Layer 1 issues
  - Netronome only 10G, not 1G
  - Multi-mode vs Single-mode Splitter
- Should have got an Intel NIC as well
- IPv6 Multicast when VFs come up?
Conclusion

- 4 Security weaknesses with existing virtual switches
- 3 Secure design principles for virtual switches
- Introduced a first secure virtual switch design
Future Work

- Extend the evaluation
- Explore the design space
- Security of SR-IOV
Contact

Kashyap Thimmaraju

Email: kash@sect.tu-berlin.de

Web: www.fgsect.de/~hashkash

Fingerprint: 5FFC 5589 DC38 F6F5 CEF7 79D8 A10E 670F 9520 75CD
References

Backup
Vision

Existing vSwitch design

An SR-IOV-based design

VM

User

Kernel

Virtual

Switch

Virt. layer

UDP

TCP

IP

IP

Eth

Eth

VM

VM

VM

vSwitch VMs

User

Kernel

SR-IOV NIC
First Measurements
Measurement Setup

- Broadly 2 topologies
  - PHY-PHY
  - PHY-VM-PHY
- Uni-directional
Phy-Phy 1 Tenant Uni-directional Aggregate Throughput
Phy-Phy 1 Tenant Uni-directional Aggregate Latency
Phy-VM-VM-Phy 1 Tenant Uni-directional Aggregate Throughput
Phy-VM-VM-Phy 1 Tenant Uni-directional Aggregate Latency
Phy-Phy 2 Tenants Uni-directional Aggregate Throughput
Phy-Phy 2 Tenant Uni-directional Aggregate Latency
Phy-VM-VM-Phy 2 Tenants Uni-directional Aggregate Throughput
Phy-VM-VM-Phy 2 Tenant Uni-directional Aggregate Latency
Lessons Learned
What have we learned so far?

- Performance
- Resources
- Management