Virtual Network Isolation: Are We There Yet?

Kashyap Thimmaraju
Security in Telecommunications
TU Berlin
Berlin, Germany
kash@sect.tu-berlin.de

Gábor Rétvári
MTA–BME Information Systems
Research Group
Budapest, Hungary
retvari@tmit.bmu.hu

Stefan Schmid*†
Faculty of Computer Science
University of Vienna
Vienna, Austria
schmiste@gmail.com

ABSTRACT

While multi-tenant cloud computing provides great benefits in terms of resource sharing, it introduces a new security landscape and requires strong network isolation guarantees between the tenants. Such network isolation is typically implemented using *network virtualization*: *Virtual switches* residing in the virtualization layer enforce isolation, e.g., via tunnel protocols and per-tenant flow rules. The design of such switches is a very active topic: Since 2009 alone, at least 22 different designs have been introduced. Our systematic analysis of 22 virtual switches uncovers 4 security weaknesses: Co-location, single point of failure, privileged packet processing and manual packet parsing. An attacker can easily undermine network isolation by exploiting those weaknesses. Hence, we introduce 3 *secure design principles* to build a *resilient* virtual switch, thereby offering strong virtual network isolation.

CCS CONCEPTS

• **Security and privacy** → **Network security**; *Distributed systems security*; • **Networks** → **Network components**; *Network architectures*; • **Hardware** → *Networking hardware*;

KEYWORDS

Network Isolation; Network Virtualization; Data Plane Security; Packet Parsing; Virtual Switches; Open vSwitch; Cloud Security; SDN; NFV; Smart NIC; SR-IOV; Co-location; Disaggregation

1 INTRODUCTION

Multi-tenant Infrastructure-as-a-Service (IaaS) cloud providers typically rely on virtualization techniques to isolate tenants from one another. Virtual switches are popularly used by IaaS cloud providers to isolate tenant networks. This is critical, as network traffic from one tenant's network must not interfere with another tenant's network.

Broadly speaking, virtual switch (vSwitch) functionality can either be present in the server or the first-hop switch (e.g., ToR switch), the former is commonly known as a *virtual switch*, and is the subject of this paper. In this setup, the vSwitch is present in the virtualization layer of the (edge) server. It interconnects the virtual machines (VMs) provisioned at the server and provides isolation between tenants' virtual networks. Connectivity across servers is achieved by establishing an "overlay" network using a tunneling protocol, e.g., VXLAN, on top of the underlying network.

Due to the relevance and wide deployment of virtualization, the design of vSwitches is a very active topic, both in academia and industry. Interestingly, however, we find that current research revolves mainly around performance and programmability. The security requirements and implications of vSwitches are less understood. This is worrisome as it has recently been shown that vSwitches potentially open a large attack surface, which can be exploited by low-resource adversaries to launch large-scale attacks [33, 34]. By *co-locating* the vSwitch with the virtualization layer, and exploiting *packet parsing vulnerabilities*, an attacker can easily propagate a *worm* from a VM to compromise an entire cloud setup such as OpenStack. Thereby violating network isolation among tenants.

Such attacks and the poorly charted security landscape in the literature raise the concern on the *security* of existing vSwitches and their *designs*. Clearly, while performance and flexibility are also important dimensions, we in this paper, take the *position* that it is time to make security a first class citizen in the design of vSwitches.

1.1 State-of-the-Art and Challenges

Let us quickly revisit the state-of-the-art designs and their shortcomings. A high-level illustration of an existing vSwitch design (Open vSwitch) is shown on the left in Figure 1. The salient aspects here are *co-location*, *single vSwitch*, *privileged packet processing* and *complex protocol processing*. The vSwitch is usually co-located with the virtualization layer. We argue that this increases the trusted computing base (TCB) of the server, as vSwitches can be tens of thousands of lines of code. Next, only one vSwitch for all the tenants is a single point of failure. The problem is further exacerbated as the virtual switch functionality is spread across the Host OS and hardware, e.g., user-space and kernel-space, which rely on *multiple* manually implemented parsers for an increasing number of network protocols [34].

The ongoing deployments of so-called *smart NICs* [10, 17, 21]–NICs that run multi-core ARM processors with a Linux OS and Open vSwitch–address the problem of co-location. Consolidating the vSwitch functionality into the NIC improves the security of the server as it reduces the trusted computing base (TCB) of the Host OS. Yet, the motivation for such a NIC, is one of performance, by running the vSwitch on the NIC's CPUs, the server's CPU cores spend fewer cycles on network virtualization. However, the vSwitch

*Also with, Internet Network Architectures, TU Berlin.
†Also with, Dept. of Computer Science, Aalborg University.

SecSoN'18, August 24, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5912-2/18/08…$15.00
https://doi.org/10.1145/3229616.3229618

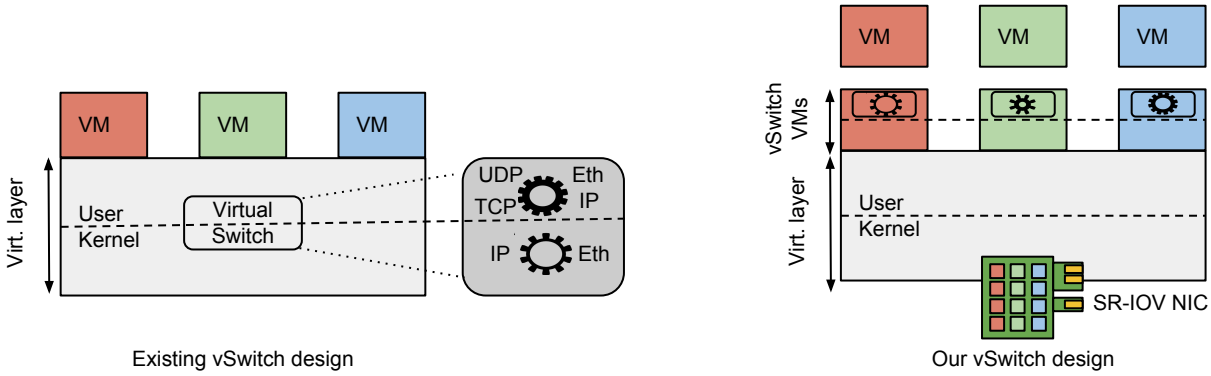

Figure 1: A high-level comparison of existing vSwitch designs (left) and our design (right). Our design leverages hardware virtualization such as SR-IOV, disaggregates the vSwitch from the virt. layer, isolates vSwitches for tenants, limits packet processing to user-space and implements minimum protocol parsing (shown as different shaped sprockets).

in the smart NIC remains co-located with its OS and has the same weaknesses described in the previous paragraph.

1.2 Vision and Position

Clearly, a key challenge in designing more secure vSwitches lies in balancing the tension between high performance and strong security. In particular, we in this paper advocate for designs that are based on sound security principles [27] *and* offer high performance. A high-level design that addresses the key weaknesses of existing vSwitches (which we highlighted previously) is shown on the right half of Figure 1. The design is based on the following *secure design principles* and enables secure network virtualization:

- **Disaggregation:** By separating the vSwitch from the virtualization layer and placing it into a guest VM (vSwitch VM), we eliminate the issue of co-location and more importantly reduce the TCB. Furthermore, by creating vSwitch VMs on a per-tenant, network slice or class-of-service basis we can strengthen isolation and avoid a single point of failure (single vSwitch).
- **User-space packet processing:** By leveraging hardware features, e.g., SR-IOV to deliver packets to a guest VM, with user-space packet processing libraries, e.g., DPDK [26], to process packets *fast* in user-space only, we adopt the principle of least privilege and reduce the impact of a compromise.
- **Limited protocol parsing:** By including the minimum required protocol parsing code we can reduce parsing vulnerabilities, e.g., buffer-overflows, out-of-bounds reads, integer underflows, etc. from occurring. This reduces the TCB and the attack surface.

Such a design firstly reduces the impact of a compromised vSwitch on the server, thereby maintaining system isolation. Secondly, multiple vSwitch VMs maintain network isolation even when a particular tenant's vSwitch VM is compromised. Third, processing a minimum set of network protocols in user-space only, using user-space packet processing libraries and hardware virtualization, further reduces the attack surface on the vSwitch VM *and* offers high performance. Additionally, the *hardware and software requirements* for our proposed design are affordably satisfied. Several servers already support Single Root I/O Virtualization (SR-IOV), Input Output Memory Management Unit (IOMMU) and Alternative Routing ID Interpretation (ARI); OSes have drivers for SR-IOV NICs and user-space packet processing (DPDK libraries).

By adopting our system, cloud providers can reap *trust* benefits. Firstly, cloud providers can trust that their management vSwitch will be isolated even when a tenant's vSwitch is compromised. Second, the cloud provider can offer tenants increased levels of isolation for their networks at a premium rate or to comply with regulations, e.g., HIPPA. Hospitals or health care providers can use a cloud service where they can trust their networks to be isolated.

2 SECURITY LANDSCAPE OF TODAY'S VSWITCHES

Research and development of vSwitches have advanced considerably since 2009. A non-exhaustive list of commercial, open-source and academic vSwitches is tabulated in Table 1. In total 22 vSwitches are shown along with their emphasis, e.g., flexibility, performance or control. The table also depicts whether the vSwitch is co-located with the virtualization layer, processes packets in kernel and/or user-space, and supports extended parsing via a unified packet parser [34]. For example, OvS is co-located with the Host OS, uses a kernel- and user-space packet processor and, supports extended parsing via a unified packet parser.

From the emphasis column, we observe that performance, flexibility, programmability and centralized control are the key drivers for vSwitch research and development. There exist only two vSwitches with an emphasis on security, namely the research prototype by Jin et al. [14] and sv3 [31] by Stecklina. The design by Jin et al. addresses the vulnerability of co-location by placing the vSwitch in a *vSwitch VM*. However, the design has two shortcomings: First, once the vSwitch VM is compromised, network isolation for all tenants is compromised; Second, the prototype uses OvS with kernel and user-space packet processing, as well as extended parsing, which

Table 1: Design characteristics of virtual switches.

	Name	Ref.	Year	Emphasis	Co-Location	Kernel	User	Ext. Parsing	Comments
Virtual Switches	OvS	[22]	2009	Flexibility	✓	✓	✓	✓	Baseline
	Cisco NexusV	[36]	2009	Flexibility	✓	✓	✗	?	Commercial
	VMware vSwitch	[37]	2009	Centralized control	✓	✓	✗	✗	Commercial
	Vale	[25]	2012	Performance	✓	✓	✗	✗	Using HPFP to increase perfomance
	Research prototype	[14]	2012	Isolation	✗	✓	✓	✓	Place OvS in a VM [14].
	Hyper-Switch	[24]	2013	Performance	✓	✓	✓	✓	Fast path in the Xen hypervisor
	MS HyperV-Switch	[18]	2013	Centralized control	✓	✓	✗	?	Commercial
	NetVM	[13]	2014	Performance, NFV	✓	✗	✓	?	Using HPFP to increase performance.
	sv3	[31]	2014	Security	✓	✗	✓	?	Can run multiple sv3 switches on the Host, isolation via processes.
	fd.io	[32]	2015	Performance	✓	✗	✓	✗	Uses Vector Packet Processing, e.g., see Choi et al. [6].
	mSwitch	[12]	2015	Performance	✓	✓	✗	✓	Using HPFP to increase performance.
	BESS	[4]	2015	Programmability, NFV	✓	✗	✓	✗	Similar to the Click modular router [15].
	PISCES	[29]	2016	Programmability	✓	✓	✓	✗	Uses a domain specific language to customize parsing.
	OvS with DPDK	[26]	2016	Performance	✓	✗	✓	✓	Using HPFP for performance; sw. countermeasures, e.g., canaries and ASLR may not be used.
	ESwitch	[19]	2016	Performance	✓	✗	✓	✗	Proprietary.
	MS VFP	[8]	2017	Performance, flexibility	✓	✓	✗	✓	Commercial.
	Mellanox BlueField	[17]	2017	CPU offload	✗	✓	✓	✓	Runs full fledged OvS on CPU in NIC. Server leased, but provider controls the network.
	Liquid IO	[21]	2017	CPU offload	✗	✓	✓	✓	Runs full fledged OvS on CPU in NIC.
	Stingray	[10]	2017	CPU offload	✗	✓	✓	✓	Runs full fledged OvS on CPU in NIC.
	GPU-based OvS	[35]	2017	Acceleration	✓	✓	✓	✓	Leverages the GPU for packet processing.
	MS AccelNet	[9]	2018	Performance, flexibility	✓	✓	✗	✓	Packet processing and flow rules offloaded to an FPGA-based NIC.
	Google Andromeda	[7]	2018	Flexibility and performance	✓	✗	✓	✓	OvS-based software switch with hardware offloads.

still exposes a considerable attack surface. sv3 by Stecklina, is co-located with the Host, however, it runs in user-space and enforces process level isolation across multiple sv3 instances.

Since 2014 several vSwitches [4, 12, 13, 19, 32] eliminated kernel packet processing, primarily for performance reasons. We note here that those vSwitches are not based on OvS. Looking more closely, we observe that the switches are still co-located with the virtualization layer, however, only a few switches [4, 19, 29, 32] address the issue of complex packet parsing by incorporating a modular design for parsing network protocols; e.g., PISCES [29] addressed the parsing issue by introducing support for P4 in OvS and ESwitch programmatically constructs minimal packet parsers dynamically using a trusted parser template library.

In the last year (2017), Smart NICs running a full Linux-based OS and OvS have emerged. Indeed, these NICs eliminate the issue of co-location and reduce the TCB of the Host OS by consolidating OvS into the NIC OS. Furthermore, this approach has the advantage of eliminating CPU cycles used by the vSwitch on the Host. However, Smart NICs inherit the security issues that OvS has as these NICs port OvS to the NIC OS and run them in kernel and user-space with extended parsing.

In 2018, Google and Microsoft resp. presented their network virtualization systems, Andromeda [7] and Azure Accelerated Networking (AccelNet) [9]. Andromeda uses OvS in the Host OS's user-space and so-called *Co-processors* for CPU intensive packet processing in VMs. Overall, packet processing exists in multiple components across the Host OS and Co-processors. AccelNet extends VFP [8] to support VFP functionality in the FPGA-based NIC. As we can see in the last two rows in Table 1, both systems co-locate the vSwitch and have complex packet processing and parsing.

We can discern the following from Table 1. Roughly 80% of vSwitches co-locate with the virtualization layer, exceptions are the 3 Smart NICs and the research prototype. Next, approximately

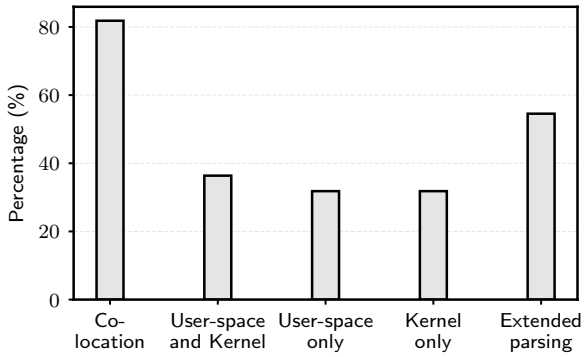

Figure 2: Distribution of vSwitch design characteristics.

35% of vSwitches process packets in kernel and user-space and 30% process packets only in the kernel. Therefore, in total we have nearly 65% of virtual switches processing packets in the kernel. Finally, at least 50% of the vSwitches support extended parsing.

Key Weaknesses. Our security analyses sheds light on the following weaknesses:

- **Co-location:** Network isolation and system isolation can easily be compromised by exploiting a simple packet processing vulnerability, e.g., buffer-overflow, while the vSwitch is co-located with the virtualization layer.
- **Single vSwitch:** A single vSwitch system cannot uphold network isolation even when placed in a separate VM.
- **Kernel packet processing:** One of the key tenets in computer security is the principle of least privilege. Handling attacker controlled data with high privileges when it can be

done with lower privileges, e.g., in user-space, is not a secure design.

- **Complex protocol parsing:** Implementing an increasing number of network protocol parsers manually is known to be error prone [28] and reduces performance [29]. Only a handful of virtual switches limit packet parsing today.

Attacker Model. We consider the attacker model from the *vAMP attack* [33, 34], which assumes that an attacker with limited resources can affordably and easily launch an attack in the cloud to compromise *network isolation* by targeting the vSwitch and its packet processing elements. However, using SR-IOV NICs requires us to assume that the NIC is trusted hardware. We assume it is more difficult to compromise the NIC as it has limited packet parsing logic, e.g., layer 2 (Ethernet and VLAN) processing only.

3 TOWARDS A SECURE VSWITCH

Our findings on the security shortcomings of existing vSwitches motivate us to reconsider vSwitches with security as a first class citizen. In particular, we identify four requirements for a vSwitch system to withstand a weak attacker in the cloud such as the vAMP attack:

(1) Maintain network isolation even when compromised.
(2) Process packets with least privilege.
(3) Reduce the trusted computing base (TCB).
(4) Limited protocol parsing.

We can meet these requirements by following three *secure design principles*: Disaggregation, User-space packet processing and Modular protocol parsing. As shown in the right half of Figure 1, by disaggregating the vSwitch from the virtualization layer into dedicated VMs, e.g., on a per-tenant basis, network isolation for other tenants can be upheld even when one tenant's vSwitch is compromised. By processing a limited number of network protocols in user-space only, we reduce the privilege and TCB of the vSwitch, thereby limiting the impact and surface of an attack. In the following we elaborate on the three *design principles*.

3.1 Disaggregation

To preserve network isolation even when the vSwitch is compromised we must first address the issue of co-location. Second, we must isolate tenant vSwitches from each other. A *disaggregation* approach addresses the problem. We can disaggregate the vSwitch from the virtualization layer, e.g., as described by Jin et al. [14], by placing it in a dedicated *vSwitch VM*. Additionally, the vSwitch VM can be disaggregated further by creating a vSwitch VM for every tenant on the server. This way, compromising one tenant's vSwitch does not violate network isolation for other tenants on the Host.

Delivering packets to and from the vSwitch VM can be accomplished using hardware virtualization such as Single Root IO Virtualization (SR-IOV) as specified by PCI-SIG [16]. SR-IOV is supported by modern servers, NICs and OSes, as well as cloud providers such as Amazon EC2 [1] and Microsoft Azure [2].

We note that there exist further potential approaches to disaggregation beyond the SR-IOV-based scheme we describe in this paper; e.g., NetBricks [20] uses a novel compiler-enforced separation between tenant's datapaths, allowing virtual network functions to explicitly pass ownership of certain resources, like packet buffers, between themselves. We further note virtual machines are only one possible form of isolation; extending our design to different isolation schemes, like processes, containers, or unikernels, is left for further study.

3.2 User-space Packet Processing

Applying secure design principles to vSwitches, we propose to process packets with least privilege, i.e., in user-space only as opposed to (an additional code path in) the kernel. This limits the impact of a compromise: Compromising an unprivileged user-space application does not give an attacker privileged access to the system, e.g., the attacker cannot make changes to the NIC from the vSwitch VM. Second, by eliminating the kernel path, the TCB of the system can be reduced. For example, OvS processes packets across the system: Kernel and user-space, and the packet processing logic in the kernel is different from user-space. A smaller TCB increases the security of the system as there are fewer lines of code to manifest as a vulnerability.

Eliminating the kernel path for vSwitches has been proposed in the literature [11, 23], and is currently available for Open vSwitch, i.e., a user-space version of OvS using DPDK [26]. Hence, this principle can also be readily adopted.

3.3 Modular Protocol Parsing

Our principle of modular packet parsing is also based on the security principles of least privilege and having a small TCB. Similar to Section 3.2, we can reduce the TCB of the system by implementing the least number of protocols required in the parser. As much as possible, we must limit the parsing to user-space, as the packet parser directly processes attacker controlled data. By parsing in user-space with low privileges, we can limit the consequences of exploiting a protocol parsing vulnerability.

There are at least three approaches to reduce packet parsing code in the literature. There is the classic "Click" modular router approach [15] adopted by BESS [4], a domain specific language approach, e.g., P4 adapted to OvS in PISCES [29], and a template based approach adopted by ESwitch [19]. In all cases, the user has the flexibility over the network protocols parsed, i.e., she can choose to support only the necessary protocols.

4 DESIGN CHALLENGES

Having described secure design principles for a vSwitch in the previous section, we now discuss some of the challenges we expect in building such a system as illustrated in Figure 1 and described in Section 1.2. In particular, we believe that this work opens up an interesting line of inquiry, e.g., it forces us to revisit the *trade-offs between security and performance.*

4.1 Delivering Packets to Correct vSwitch VMs

As per the PCI-SIG specification for SR-IOV, the OS that has the so-called Physical Function (PF) driver can map the NIC's resources, e.g., Intel's Virtual Machine Device queues (VMDq) and Base Address Registers (BARs), as an individual PCI device to the resp. vSwitch VMs. Each vSwitch VM will then have an Ethernet interface configured with at least a unique MAC address and possibly an IP address. Hence, each vSwitch VM can be uniquely identified. The

administrator can then configure the NIC to filter packets based on the respective vSwitch VM's MAC address. Packets from the SR-IOV NIC can then be delivered to the respective vSwitch VM, bypassing the virtualization layer. Note that SR-IOV NICs typically defend against MAC address spoofing attacks, thereby preventing a compromised vSwitch VM from violating network isolation by intercepting another tenant's packets.

4.2 Processing Packets in User-Space

Packets from the NIC can be delivered to the user-space vSwitch using so-called *kernel-bypass* techniques. Such techniques are commonly supported by commodity NICs, e.g., most modern NICs support Intel's DPDK library. There are two ways to achieve kernel-bypass: using Interrupts, or Polling. In an Interrupt mechanism, the NIC sends an interrupt that traps the CPU. The CPU then services the interrupt and delivers the data in the buffer to the user-space application. In a Polling mechanism, the NIC and Host use shared memory to read and write packets. The user-space application constantly monitors the shared memory to check if packets have arrived for it or not. The NIC can write to the shared memory region using Direct Memory Access (DMA) and can be restricted to that memory using the IO Memory Management Unit (IOMMU). Indeed the two techniques have trade-offs, Polling requires dedicated CPU cores whereas Interrupts do not. However, Polling reduces latency whereas Interrupts do not. Depending on the use-case one or the other can be chosen. BESS for example is a user-space only vSwitch, and OvS has a user-space version available [26], both of which use the DPDK library. mSwitch [12] is an example of an Interrupt-based vSwitch that runs only in the kernel.

In some virtual switches, e.g., OvS, the kernel has a cache that increases packet processing performance (latency and throughput) considerably. One may be concerned that user-space packet processing precludes a flow cache, however, that is not the case with OvS. OvS with DPDK supports a user-space flow cache [5] that we could leverage in our prototype.

4.3 NIC Drivers

The OS that has the Physical Function (PF) driver of the NIC has full control over the NIC, whereas the OS that has the Virtual Function (VF) driver of the NIC has limited control over the NIC. For example, only the PF driver is allowed to configure the filter on the NIC that is necessary to correctly deliver packets to the respective vSwitch VMs. To use the drivers "out-of-the-box", it would appear reasonable to assign the VF driver to the vSwitch VM, however, doing so introduces limitations. For example, a vSwitch VM cannot dynamically connect a newly spun up tenant VM, the administrator must do so. On the other hand, the administrator must a priori allocate the PCIe device (queues and BARs) to vSwitch VMs and their respective tenant VMs, which may be inefficient use of resources or difficult to estimate. To avoid such limitations, one could potentially introduce a special vSwitch VM driver that is a subset of the PF driver.

4.4 Traffic between Tenants and their vSwitch

We discussed how packets arriving from the physical network (e.g., from the switch connected to the server) can be delivered to the vSwitch VM previously. Now, we discuss two ways the packets can be exchanged between tenant VMs that are co-located on the same server: Bouncing packets off the NIC or using shared memory between the vSwitch VM and its respective tenants. To bounce packets off the NIC, the tenant VMs must also be assigned NIC resources and have the VF driver installed. The advantage of this approach is reduced latency and bypassing the virtualization layer. However, this can introduce a lot of traffic along the PCI bus and NIC resources are limited, and could prevent live migration of tenant VMs (an SR-IOV implementation limitation). If polling via shared memory (as huge pages) is used between the vSwitch VMs and the respective tenants, then PCI bus contention is traded for CPU cores and cycles. Naturally, each method has its pros and cons, and hence, must be chosen depending on the use-case and available resources.

4.5 Resource Allocation

From Figure 1 it is clear that our disaggregation approach calls for dedicated vSwitch VMs, thereby reducing the number of available VMs for compute tasks. Of course, when high levels of isolation are necessary such an approach can be taken. An alternative minimalistic solution would be to have at least 2 vSwitch VMs: Management and Tenants. The cloud providers virtual network is then isolated from the untrusted Tenants. Hence, we expect at least 2 extra VMs per server to be the minimum resource overhead introduced. Additionally, the vSwitch VMs will require at least 2 dedicated CPU cores for packets to be delivered to the vSwitch VM. CPU cores for packet processing will increase when polling is used. The memory consumed will also increase depending on the approach taken. Allocating memory a priori reduces the dynamic memory management capabilities of modern hypervisor, however, it is difficult to know ahead of time *how much* memory to allocate for the vSwitch VMs when polling is used. Again, depending on the resources at hand and use-cases, specific design choices need to be made. It is impossible to have a "one size fits all" solution.

4.6 Management

The administrator/orchestrator can spin up/down vSwitches and assign VFs to the vSwitch VMs. Recall that each vSwitch VM is configured with a unique MAC and IP address. Hence, we expect address management to increase for every vSwitch VM introduced. Furthermore, the number of connections to the centralized controller will also increase. This can introduce (one-time) management/operational overhead, e.g., integrating APIs into the cloud managament system software, mapping vSwitch VMs to tenants. On a different note, introducing vSwitches on a per-tenant basis for example, could make it easier to manage the vSwitch flow configuration. Changes to the vSwitch configuration are limited to the specific tenant.

4.7 Reducing Protocol Parsing

Although there exist multiple approaches to reduce packet parsing code in the literature, there are some challenges to overcome. For example, PISCES appears to be inactive, ESwitch is proprietary, we encountered an issue when trying to use BESS on our server and OvS does not support modular packet parsing. At the time

of writing this paper BESS appears to be the most suited vSwitch, however, we may have to first use OvS and then migrate to BESS after fixing the problem.

4.8 Security of SR-IOV NICs

Our design involves hardware features, in particular SR-IOV NICs. However, the specification for SR-IOV is currently closed, and security issues of using SR-IOV are yet to be well understood. For example, a handful of researchers have uncovered denial of service attacks [30, 38] and covert channels [3] when SR-IOV NICs are used in IaaS clouds.

5 CONCLUSION AND FUTURE WORK

In this paper we systematically uncovered 4 key security weaknesses of existing virtual switches: Co-location, single vSwitch, kernel packet processing and complex protocol parsing. Our findings led us to identify 4 requirements for a secure vSwitch system. We then outlined 3 secure design principles to meet the identified requirements namely, disaggregation, user-space packet processing and modular protocol parsing. Finally, we sketched a design based on our principles and described some of the important challenges we will face in building such a system.

We aim our analysis as a first step. As future work, we plan to implement our design and conduct an extensive evaluation to shed light on factors that influence the performance of our prototype. In developing such a design, network virtualization via vSwitches will be resilient to a compromise and have a reduced attack surface, thereby preserving a critical virtualization requirement, *isolation*.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable feedback and comments. We also thank Robert Buhren for the initial discussion on SR-IOV. The first author (K. T.) acknowledges the financial support by the Federal Ministry of Education and Research of Germany in the framework of the Software Campus 2.0 project nos. 10044972 and 10045058. The second author (G. R.) is with the Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics.

REFERENCES

[1] AWS. 2018. Enhanced Networking on Linux. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html. (2018). Accessed: 24-01-2018.
[2] Microsoft Azure. 2018. Create a Linux virtual machine with Accelerated Networking. https://docs.microsoft.com/en-us/azure/virtual-network/create-vm-accelerated-networking-cli. (2018). Accessed: 24-01-2018.
[3] Adam Bates et al. 2014. On detecting co-resident cloud instances using network flow watermarking techniques. *Springer International Journal of Information Security* (2014).
[4] BESS Comitters. 2017. BESS (Berkeley Extensible Software Switch). https://github.com/NetSys/bess. (2017). Accessed: 09-05-2017.
[5] Bhanuprakash Bodireddy and Antonio Fischetti. 2016. OVS-DPDK Datapath Classifier. https://software.intel.com/en-us/articles/ovs-dpdk-datapath-classifier. (2016). Accessed: 05-06-2018.
[6] Sean Choi, Xiang Long, Muhammad Shahbaz, Skip Booth, Andy Keep, John Marshall, and Changhoon Kim. 2017. PVPP: A Programmable Vector Packet Processor. In *Proc. SOSR*. ACM, 197–198.
[7] Michael Dalton et al. 2018. Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization. In *Proc. NSDI*. 373–387.
[8] Daniel Firestone. 2017. VFP: A Virtual Switch Platform for Host SDN in the Public Cloud. In *Proc. NSDI*. 315–328.
[9] Daniel Firestone et al. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *Proc. NSDI*. 51–66.
[10] Andy Gospodarek. 2017. The Rise of SmartNICs – offloading dataplane traffic to...software. https://youtu.be/AGSy51VlKaM. (2017). Open vSwitch Conference.
[11] Sangjin Han, Keon Jang, Aurojit Panda, Shoumik Palkar, Dongsu Han, and Sylvia Ratnasamy. 2015. SoftNIC: A software NIC to augment hardware. (2015). Tech. Rep. UCB/EECS-2015-155.
[12] Michio Honda, Felipe Huici, Giuseppe Lettieri, and Luigi Rizzo. 2015. mSwitch: a highly-scalable, modular software switch. In *Proc. SOSR*. 1.
[13] Jinho Hwang, KK Ramakrishnan, and Timothy Wood. 2014. NetVM: high performance and flexible networking using virtualization on commodity platforms. In *Proc. NSDI*. 445–458.
[14] Xin Jin, Eric Keller, and Jennifer Rexford. 2012. Virtual Switching Without a Hypervisor for a More Secure Cloud. In *Proc. USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (HotICE)*. San Jose, CA.
[15] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M Frans Kaashoek. 2000. The Click modular router. *ACM Trans. Computer Systems* 18, 3 (2000), 263–297.
[16] Patrick Kutch. 2011. PCI-SIG SR-IOV primer: An introduction to SR-IOV technology. *Intel application note* (2011), 321211–002.
[17] Mellanox. 2017. Mellanox BlueField SmartNIC. https://bit.ly/2JaMitA. (2017). Accessed: 05-06-2018.
[18] Microsoft. 2013. Hyper-V Virtual Switch Overview. https://technet.microsoft.com/en-us/library/hh831823(v=ws.11).aspx. (2013). Accessed: 27-01-2017.
[19] László Molnár, Gergely Pongrácz, Gábor Enyedi, Zoltán Lajos Kis, Levente Csikor, Ferenc Juhász, Attila Kőrösi, and Gábor Rétvári. 2016. Dataplane Specialization for High-performance OpenFlow Software Switching. In *Proc. ACM SIGCOMM*. 539–552.
[20] Aurojit Panda et al. 2016. NetBricks: Taking the V out of NFV. In *Proc. OSDI*. 203–216.
[21] Manoj Panicker. 2017. Enabling Hardware Offload of OVS Control & Data plane using LiquidIO. https://youtu.be/qjXBRCFhbqU. (2017). Open vSwitch Conference.
[22] Ben Pfaff. 2013. Open vSwitch: Past, Present, and Future. http://openvswitch.org/slides/ppf.pdf. (2013). Accessed: 27-01-2017.
[23] Gergely Pongrácz, László Molnár, and Zoltán Lajos Kis. 2013. Removing roadblocks from SDN: OpenFlow software switch performance on Intel DPDK. In *European Workshop on Software Defined Networking*. IEEE, 62–67.
[24] Kaushik Kumar Ram, Alan L Cox, Mehul Chadha, Scott Rixner, and TW Barr. 2013. Hyper-Switch: A Scalable Software Virtual Switching Architecture. In *Usenix Annual Technical Conference (ATC)*. 13–24.
[25] Luigi Rizzo and Giuseppe Lettieri. 2012. VALE, a Switched Ethernet for Virtual Machines. In *Proc. ACM CoNEXT*. 61–72.
[26] Robin G. 2016. Open vSwitch with DPDK Overview. https://software.intel.com/en-us/articles/open-vswitch-with-dpdk-overview. (2016). Accessed: 27-01-2017.
[27] Jerome H. Saltzer and Michael D. Schroeder. 1975. The protection of information in computer systems. *Proc. IEEE* 63, 9 (1975), 1278–1308.
[28] Len Sassaman et al. 2013. Security Applications of Formal Language Theory. *IEEE Systems Journal* 7, 3 (Sept 2013), 489–500.
[29] Muhammad Shahbaz, Sean Choi, Ben Pfaff, Changhoon Kim, Nick Feamster, Nick McKeown, and Jennifer Rexford. 2016. Pisces: A programmable, protocol-independent software switch. In *Proc. ACM SIGCOMM*. 525–538.
[30] Igor Smolyar, Muli Ben-Yehuda, and Dan Tsafrir. 2015. Securing Self-Virtualizing Ethernet Devices. In *Proc. Usenix Security Symp.* 335–350.
[31] Julian Stecklina. 2014. Shrinking the Hypervisor One Subsystem at a Time: A Userspace Packet Switch for Virtual Machines. In *Proc. ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments (VEE)*.
[32] The Fast Data Project. 2017. What is the Fast Data Project (FD.io)? https://fd.io/about. (2017). Accessed: 05-06-2018.
[33] Kashyap Thimmaraju et al. 2017. The vAMP Attack: Compromising Cloud Systems via the Unified Packet Processor. In *Proc. ACM Workshop on Cloud Computing Security Workshop*.
[34] Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann, and Stefan Schmid. 2018. Taking Control of SDN-based Cloud Systems via the Data Plane. In *Proc. SOSR*.
[35] Janet Tseng et al. 2017. Accelerating Open vSwitch with Integrated GPU. In *Proc. ACM Workshop on Kernel-Bypass Networks*.
[36] Rick Vanover. 2008. Virtual switching to become enhanced with Cisco and VMware announcement. http://www.techrepublic.com/blog/data-center/virtual-switching-to-become-enhanced-with-cisco-and-vmware-announcement. (2008). Accessed: 27-01-2017.
[37] VMware. 2009. VMware ESX 4.0 Update 1 Release Notes. https://bit.ly/2sFTuTy. (2009). Accessed: 05-06-2018.
[38] Zhe Zhou, Zhou Li, and Kehuan Zhang. 2017. All Your VMs are Disconnected: Attacking Hardware Virtualized Network. In *Proc. ACM Conference on Data and Application Security and Privacy (CODASPY)*.