Accountability in Hosted Virtual Networks

Eric Keller
Princeton University,
Princeton, NJ, USA
ekeller@princeton.edu

Ruby B. Lee
Princeton University,
Princeton, NJ, USA
rblee@princeton.edu

Jennifer Rexford
Princeton University,
Princeton, NJ, USA
jrex@cs.princeton.edu

ABSTRACT

Virtualization enables multiple networks, each customized for a particular purpose, to run concurrently over a shared substrate. One such model for managing these virtual networks is to create a hosting platform where companies can deploy services by leasing a portion of several physical routers. While lowering the barrier for innovation in the network, this model introduces new security concerns. In this paper we examine the issue of accountability in this setting of hosted virtual networks. That is, how a *service provider* can know its software is running without modification and that the *infrastructure provider's* physical router is forwarding packets as instructed with the quality of service promised. Rather than presenting a single specification of what every router on the Internet must look like, in this paper we examine two possible approaches: one that detects violations by monitoring the service and one that prevents violations from occurring in the first place. For each, we provide a description of an architecture that can be achieved with technology available today, the limitations of that architecture, and then propose an extension which overcomes the limitations.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design; C.2.6 [**Computer Communication Networks**]: Internetworking

General Terms

Design, Security, Measurement, Performance

Keywords

Virtualization, router architecture, security, accountability

1. INTRODUCTION

There have been many proposals addressing the many challenges facing the Internet, yet it has proven extremely difficult to deploy the solutions in the form of modified protocols or new services. Virtualization has the promise of breaking this stalemate by providing a means to concurrently run multiple virtual networks, each customized for a particular use, on a shared physical infrastructure. This promise is becoming a reality as major router vendors are starting to support both router virtualization (running multiple virtual routers in parallel on a single router) [17] and router programmability (running customized protocols) [16] [6].

We argue that the future will bring about hosted virtual networks [12] [26], a model analogous to hosted cloud computing services, such as Amazon's EC2 [1]. In this model, the *infrastructure provider* deploys and manages a set of physical routers and the links between them, and forms business relationships with other infrastructure providers to ensure global connectivity. The *service provider* then leases virtual routers from one or more infrastructure providers to create a virtual network running its own custom control processes on each virtual router. It is this decoupling between infrastructure and service provider that would enable service providers to deploy a customized network or new service without needing to build an expensive infrastructure.

However, this model of hosted virtual networks raises a new class of security concerns. In this paper, we are particularly concerned with accountability. We define *accountability* for hosted virtual networks as the provision of services by an infrastructure provider to service providers as promised in the Service Level Agreement (SLA), and the provision of mechanisms to detect and record violations or ensure compliance. Here, we must be able to distinguish a bug in the service provider's own software from a violation in the service level agreement (SLA) by the infrastructure provider. Any deviation from the SLA in any way is considered a violation by the infrastructure provider. This includes (i) not running the service provider's control software exactly as provided, (ii) not forwarding or filtering packets as instructed by the control software, and (iii) not forwarding packets with the appropriate quality of service.

Complicating the matter are two issues: that the platform is hosted and that the platform is shared. Since the platform is hosted, the service provider has limited visibility into the actual operation of the physical router, making it difficult to assign proper blame. Since the platform is shared, there is more opportunity for attackers to attack the system. An attacker (which may be a competing service provider) can lease its own virtual router and intentionally attack the system, either to hurt the competition or just to be malicious.

While one might look to hosted cloud computing for solutions, given its tremendous recent growth, the issue of accountability has received relatively little attention in that field. This may be due to (i) the fact that the leaders in this space are considered highly reputable companies with those

VISA'09, August 17, 2009, Barcelona, Spain.

running services on it implicitly trusting the hosting company, and (ii) critical IT infrastructure has yet to migrate to these cloud services so the issue has not been pushed to the forefront. However, we feel that the issue cannot be ignored for hosted virtual networks. A service provider will often need to use multiple infrastructure providers to achieve global reach, leading them to place trust in a lot of different (competing) parties. In fact, in some regions, there may be little competition between infrastructure providers, forcing the service provider to use one that is less than reputable.

In this paper, rather than proposing a single specification of a router that would "solve" this problem, we instead take a first step that explores the solution space by discussing two approaches that can be taken. For each approach we present an architecture that can be built by adapting technology that is currently available and then propose an extension that overcomes the limitations of that architecture in the setting of hosted virtual networks.

The first approach is to detect violations by observing the behavior of the virtual network and then checking against the expected behavior. Here, using traditional network measurement techniques, the service provider will monitor its service and gather enough evidence to convince the infrastructure provider of the violation. The second approach builds upon recent advances in trusted computing and processor architecture to enable the infrastructure provider to build in mechanisms that prevent any violations from occurring in the first place by ensuring that the software is resistant to tampering. This effectively preempts the need to prove innocence.

The paper is organized as follows. In Section 2 we discuss some of the unique characteristics of hosted virtual networks and the challenges and opportunities that come with them. In Section 3, we present our threat model and discuss the assumptions made. We then discuss the proposed solutions by addressing the two possible approaches: using network observation to detect faults (Section 4) and providing mechanisms that prevent the software from being tampered with (Section 5). We then wrap up with a discussion (Section 6) and conclusions (Section 7).

2. CHALLENGES AND OPPORTUNITIES

There are many unique properties of routing in hosted virtual network when compared to general-purpose computing or to more traditional routing. In this section we examine the unique challenges and opportunities associated with these properties.

2.1 Challenges

The nature of routing and the specific setting of hosted virtual networks makes it difficult to apply general purpose trusted computing techniques directly. Here we look at a few of these challenges.

Forwarding at high data rates: Perhaps the greatest challenge is that forwarding in routers requires high data rates. Network interfaces of 10Gbps are becoming common and 40Gbps are present in high-end routers. Routers will have many of these interfaces. This, coupled with the desire for low latency, means any mechanism to ensure correct packet handling must not add much overhead.

Shared packet handling: For software-based routers, studies have shown that performing forwarding inside of a virtual machine [10] or using a mechanism to "sandbox" the code in the kernel [25] incurs a significant performance penalty. Therefore, achieving high performance requires that packet handling must be executed in a shared kernel. However, allowing a service provider to run custom software (*e.g.,* to perform monitoring) in privileged kernel mode poses many risks. For hardware-based routers, little work has been done to safely enable running custom code on a network processor or inserting a custom circuit on an FPGA. While this is an open area of research, we proceed under the assumption that the service provide will not be able to implement custom functionality in the shared data plane.

Highly configurable packet handling: In addition to setting the entries in the forwarding table, a router can be configured by the network operator to block traffic, provide different quality-of-service levels, and balance load across multiple links. There is not a single table or small block of code that, if guaranteed correct, would guarantee the correct operation of the router.

Limited infrastructure provider competition: A physical network requires physical resources (routers and fiber connecting them) that are spread out over a large area. This is labor and capital intensive and likely means there will be a limited number of infrastructure providers in a given region. This means service providers may have to use an infrastructure provider they may not trust.

Multiple infrastructure providers: As a physical network is spread over a large area, it is unlikely that a single physical infrastructure provider will be able to provide global end-to-end reach. This is unlike cloud computing where one can go with a single company that has a great reputation for the entire infrastructure. Ignoring accountability in hosted virtual networks would mean that a service provider would have to trust a lot of (competing) parties.

2.2 Opportunities

Just as there are challenges imposed by the shared and hosted network platform, there are also several unique opportunities as compared to general computing.

Multiple infrastructure providers: While listed as a challenge as well, having multiple infrastructure providers may also present an opportunity. As the infrastructure providers will likely have different platforms, a vulnerability in one may not be present in the other, reducing the likelihood of an entire network of virtual routers being compromised.

Control messages exchanged between routers: Since control messages are exchanged between routers, this opens up the possibility that one router can be used to check the correctness of the other, similar to [29]. In the absence of collusion between infrastructure providers, we can take advantage of the fact that a service provider will have virtual routers placed in separate infrastructure providers to enable the virtual routers to perform checking of each other.

Simple and well-defined interfaces: There is a clean separation of functionality between the control plane, which runs routing processes, and the data plane, which handles each packet. The configuration of the data plane from the control plane follows a simple and well-defined interface specifying such things as forwarding table entries. Because of this, the interaction between the virtualization layer and the virtual routers can be minimized to just this interface — making it more difficult for the system to be compromised by an attacker.

Volume of control plane traffic is limited: Control messages are sent in fairly low volumes. As such, control messages can be logged as part of the accountability process (*e.g.,* gathering evidence), allowing approaches that perform checks after the fact in addition to the more strict approaches that prevent attacks. Additionally, routing algorithms commonly only base the computation on the set of routes currently advertised by neighbors. This can further trim the storage requirements in logging.

Independence between data packets: Just as control messages can be logged to support checking the correctness of operation, the data traffic may need to be logged as well — which is impractical given the high volume. However, since each packet is forwarded independently in routers, sampling and summary statistics can be used instead.

3. THREATS AND TRUST

The goal is to provide an architecture where a service provider can know its software is running as written and that the physical router is processing packets as instructed with the promised quality of service. Before discussing how such assurances can be achieved, in this section we discuss our assumptions and threat model.

Our assumed generic router architecture is shown in Figure 1. Each of the physical routers is partitioned into multiple virtual routers (one for each service provider) with the virtualization managed by a virtualization layer (controlled by the infrastructure provider). We simply use the generic term "virtualization layer" to cover the various technologies in use by routers today: full or para-virtualization with Vyatta's virtualized router [28], kernel-based virtual machines for providing software based redundancy in the Cisco ASR 1000 [5], or container based virtualization for providing a lightweight mechanism for integrating custom network services on Cisco's Integrated Service Routers [6]. Note that routers can be either hardware based or software based. In a hardware-based router, the packet handling is done in line cards (provided by router vendors) connected via a switching fabric. In contrast, in a software-based router, the packet handling is done in software. The interfaces are network interface cards (NICs) connected to main memory via a peripheral bus such as PCIe. In either the hardware case or the software case, the forwarding functionality is configured through an API that is exposed to the virtual router's control processes. This configuration goes through the virtualization layer managed by the infrastructure provider.

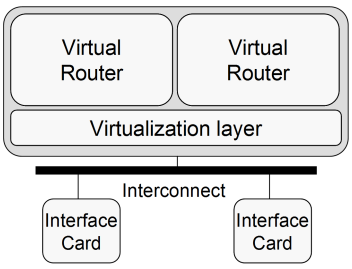

Figure 1: Generic router architecture showing two virtual routers and two network interfaces.

From this we can see each of the parties involved, giving better understanding of the possible security concerns. As different parts of the virtualization systems are controlled by different business entities, they each have different motivations, and therefore may not trust each other. In our threat model we assume there is mutual distrust between the infrastructure provider and service provider. We also assume that a 3rd party router or component (e.g., processor or network card) vendor is neutral and not colluding with either party. Furthermore, the system is complex, consisting of many different components, and thus these components can be attacked.

We make two assumptions about the capabilities of an attacker. First, we assume that any software on a general purpose processor can be compromised, including the virtualization layer. Attacks on the software can come from at least three possible sources. (i) A service provider: Because it is a shared platform, there is extra opportunity for a malicious party to attack the system. An attacker can lease resources on a router and run software to attempt to exploit any security vulnerabilities of the router's virtualization layer. (ii) A dishonest infrastructure provider: As we do not assume an honest infrastructure provider, we cannot trust that the underlying virtualization layer has not been modified by the infrastructure provider. While it would seem that the risk of a bad reputation would prevent a company from intentionally delivering a lower service level than promised, it is not unheard of [11]. (iii) A rouge employee: In addition to intentional modification for financial gains by the infrastructure provider, a rouge employee also has access to the router and can be paid to attack the system.

Second, we assume the functionality on the NIC or line card is not modifiable by the infrastructure provider. These cards contain custom ASICs, FPGAs, or network processors (often developed in house by the router vendor). FPGAs already have the ability to boot from a configuration that is stored in a non-volatile storage device (*e.g.,* CompactFlash) in encrypted form. The router vendor programs the decryption key, which is stored internal to the FPGA. It would be a simple extension to support this functionality in the ASICs or network processors for any firmware they need.

4. DETECTING VIOLATIONS WITH NETWORK MEASUREMENT

In this section we apply traditional network measurement techniques to detecting violations in SLAs in hosted virtual networks. Recall that a SLA in hosted virtual networks consists of three main components that we are concerned with: control processes are run unmodified, the physical router is forwarding packets as instructed by the control software, and the physical router is providing the quality of service agreed upon. We first discuss techniques used today to monitor networks not under direct control (*e.g.,* monitoring the SLA compliance of an ISP). Then we propose extending the interface card to perform much of the functionality, but with greater accuracy and efficiency.

4.1 Monitoring SLA Compliance of Networks

The SLA between the service provider and the infrastructure provider has much in common with what ISPs offer customers today. Today, there is an expectation that paths are available with certain metrics (*e.g.,* loss, delay, jitter). This is monitored through measurements performed at the customer's sites [23]. As a simple example, the customer could periodically perform a ping between each site to determine the availability and latency of the path.

Making the assumption that infrastructure providers are not colluding with one another, we can extend this concept to the hosted virtual networks model by having each sub-network check each other sub-network, where a sub-network is all of the routers in a single infrastructure provider's network[1]. Reusing the simple ping example, rather than having one of the customer's sites pinging another one of their sites through the ISP being checked, a virtual router in one infrastructure provider would ping a virtual router in another infrastructure provider going through the network of the infrastructure provider being checked.

Of course, checking the data plane assumes knowledge of how the data plane was supposed to behave. To know this, we must determine how the data plane was configured by the control plane. For this we need to log control messages being sent across infrastructure provider boundaries and replay this against a simulation of the sub-network and the control software running on it. This can be used to check that the sub-network is propagating control messages as expected, as is done in PeerReview [15]. Any discrepancy between the messages that should have been received and the messages that were actually received is a potential indicator of a fault. This can also be used to determine the way each router's data plane was actually configured versus how it should have been.

4.2 Extending the Interface Card

Using the probing mechanism of the previous section has a number of limitations in terms of each component of the SLA. In particular (i) logging of control messages is only done at the edge of each infrastructure provider's network, leading to a computationally expensive task to simulate the internal operation of that sub-network, (ii) probing from virtual router to virtual router does not hide measurement traffic from the infrastructure provider being measured, enabling the potential to obfuscate the true quality of the path, and (iii) the accuracy of determining the path a packet takes is dependent upon the number of probe points (how many virtual routers send probe packets to how many other virtual routers), and therefore as the size of the sub-network being monitored increases, pinpointing a specific faulty router becomes more difficult (making it hard to distinguish between a fault the infrastructure provider is responsible for and a bug in the service provider's own software).

To deal with each of these issues, we propose extending the functionality of the interface card to perform similar functionality but with more accuracy and requiring less storage and computation to perform the checks. As this is a only a proposal at this point, further work is needed to accurately weigh the limitations against the effort of extending the interface card and the need to trust a third party.

While it's an understandable concern that assuming an extension to the network card may be impractical, we feel that it is indeed reasonable to expect NIC enhancements as (i) NIC vendors will want to differentiate their product, (ii) reputable infrastructure providers will want to provide the monitoring as a service (perhaps at extra cost) to give them a competitive advantage, and (iii) service providers already deal with routers from a multitude of vendors and dealing with multiple NIC logging formats is manageable.

[1]We assume an entire infrastructure provider can be compromised, either through a dishonest infrastructure provider or an attacker exploiting a vulnerability.

In the following, we discuss each of the extensions:

Log control messages at each interface: Checking an entire sub-network is difficult and computationally expensive as the software of the entire sub-network needs to be simulated to determine any inconsistencies in the control messages exchanged, and to determine how the data plane was configured. This, in turn, will lead to a compromise on the accuracy as checks will performed less often. We can take the monitoring approach to a finer granularity by doing it at each node. To do this, we propose extending the functionality of the interface card to perform logging of control messages (interface card is either the network interface card for software based routers or the line card for hardware based routers). This places trust in the vendor of the interface cards.

Securely performing measurements: In addition to reducing the complexity of performing the checks, extended capabilities in the data plane are necessary for securely performing the data plane measurements. Packets used for measurement to determine latency (probes or sampled packets) must be indistinguishable from packets that are not [3][14]. Otherwise, an infrastructure provider could treat the measurement packets favorably (*e.g.,* forward them correctly with low latency) while the other packets get treated unfavorably (*e.g.,* dropped). For this we propose the interface card include support for passive probing. With passive probing, actual data packets serve as implicit probe packets that are acknowledged by the end-point of the probe path. Each end-point must agree how to determine which packets are used, for example using the same key and applying a hash of the immutable fields of the packet as done in trajectory sampling [8]. However, this implies that the key must be known only to the interface cards and the service provider. This will require public key cryptography to enable the service provider to install a key on each interface card.

Tracing a packet's path: Support is needed to be able to trace a packet through the exact path that it takes. This allows us to check the packet was forwarded as instructed and be able to distinguish between a packet that was dropped because it was supposed to be dropped (*e.g.,* it was blocked by a filter) or for another reason. If it is dropped for another reason, then it should be included in the count against the SLA characterization of allowable packet loss. Unlike the control plane, logging each packet is not feasible given the high data rates. Fortunately, the requirements in this case are very similar to the IP traceback problem [22], for which we can use a similar technique.

With IP traceback, a bloom filter is maintained at each router using multiple hashes over the immutable packet content. Then, given a particular packet, the path that it took can be determined by querying the destination's adjacent routers to see if the bloom filter contains that packet (and subsequently querying the adjacent routers of those that do). In our situation, we are more interested in ensuring that packets are being handled correctly (which includes forwarded along the correct path as well as being correctly dropped at filter points). This means sampling is done at the ingress to the network rather than egress so as to include all packets, not just those that made it all the way through the network. Note that this does not require the service provider to provide a key to each device in the network, nor does it require each key used to be the same, avoiding the complexity of distributing private keys to each router. Bloom filters

track all packets, and each bloom filter can be different since it only needs to be able to respond to membership queries, not have the same filter values.

5. PREVENTING TAMPERING

In the previous section, we discussed more traditional techniques for monitoring compliance with SLAs in networking. However, these mechanisms are limited in that they are not performing real-time protection but rather detecting violations after the fact. Additionally, they each have significant storage and computation requirements to perform the off-line checks.

Rather than trying to detect if an infrastructure provider is in compliance, in this section we use recent advances in trusted computing and processor architecture to discuss how a router could be re-architected to have assurances built in. Here, the architecture has three components — control plane, data plane, and the communication channel between them (which can be the virtualization layer). If the control plane software and the data plane implementation are both trusted and cannot be tampered, and a secure channel between the two can be established, then the platform will behave as advertised and as instructed (by the service provider's control software). This is because knowing how the data plane is configured and that it has not been tampered with means that it is forwarding as instructed with a given quality of service (per the data sheet based on how it is configured). In this section we first discuss how this can be achieved using technology available today and then discuss an improvement on that architecture by proposing an extension to a general-purpose microprocessor.

5.1 Trusted Platform Module

A trusted platform module (TPM) [27] is a chip available today that can be used to measure the integrity of software at launch. Each layer in the platform performs a measurement of the layer above it before launching the next layer (*e.g.,* the BIOS measures the OS, the OS measures each application), forming a chain of trust. To verify the application has not been modified, the values stored in the TPM would contain the measurement which can be checked against the expected values.

There are two main limitations with this approach[2]. First, using a TPM does not protect against any dynamic attacks. It performs a measurement only when an executable is started, not while it is executing and therefore modifications to the control processes would not be prevented. Second, as it is built on a chain of trust, the virtualization layer (infrastructure provider) needs to be trusted by each of the virtual routers (service provider), an approach taken in Terra [13]. Both malicious attacks by other service providers or direct modifications by the infrastructure provider can go unnoticed. This would enable both the control plane and the data plane, which is run in the shared virtualization layer for highest performance, to be dynamically modified. While there have been attempts at using the TPM while protecting against dynamic attacks, the performance penalty was substantial (1 second switching time) and limited in functionality (cannot run the entire control process protected) [20].

However, despite the limitations, given the nature of routing, we can take advantage of the separation between the control plane and the data plane in routers. Researchers have proposed running a centralized control plane which controls the data plane of one or more routers from a remote server [4] [21]. While proposed for different reasons, that approach can be used here to allow the control processes to be run on trusted platforms. In the extreme, the remote control plane could be run on machines under the control of the service provider in its own facilities. In this case, only the data plane is run in a hosted virtual network platform. Alternatively, if the main concern is that the platform is shared and enables other service providers to attack the virtualization layer, the infrastructure provider could provide the service provider with dedicated servers for running control processes, using the TPM for performing integrity checks. Some hosted cloud computing providers have taken a similar approach to ensure a more secure platform, dedicating a server to a single customer [2].

Just as the control plane of one virtual network can be separated from the shared router, the data plane can also be separated from the software running on the physical router (virualization layer and other control processes). This separation is already present in the commonly used hardware based routers which perform the forwarding functionality on the line cards using ASICs, FPGAs, and/or network processors. As our threat model assumes that the vendor of these devices is neutral, and therefore each party mutually trusts that the device acts accordingly, the actual execution of forwarding can be trusted.

Forwarding in the data plane can also be achieved in software based routers. Rather than having the control processes and forwarding functionality co-exist on the same processor, we can instead have a dedicated processor for forwarding, attested with the TPM. One of the main issues with using the TPM is that it performs the attestation at boot time but does not protect against dynamic attacks. However, in this case, that is less of a problem since (i) the data plane is not affected by the vulnerabilities of the virtualization layer (there is no virtualization layer, just the packet handling functionality), (ii) we can run with high performance (do not need extra protection mechanisms, as there is no switching between protection domains), and (iii) no software other than the forwarding code can run on the processor (there is less opportunity given to malicious service providers).

With physical separation, however, the interaction between each service provider's control processes and the data plane must go through the untrusted virtualization layer or an untrusted network. For this, one can create a secure channel between the control process and the line card where each configuration command sent to the line card is secured. This can be achieved either through the control process encrypting each configuration command using a public key of the line card, or more efficiently using the public key to establish a shared private key (*e.g.,* with Diffie-Hellman [7]). An alternative is to periodically verify the data plane's table values by requesting the current state and having the data plane sign the result.

With both integrity verified control and data planes, and a secure communication channel between them, the infras-

[2]A third limitation would be the possible vulnerability of the TPM itself. The Xbox 360 is rumored to include a TPM to prevent playing pirated games yet mod chips are available to circumvent that protection.

tructure provider has built in enough mechanism to assure each of the service providers that the platform is behaving as advertised and as instructed.

5.2 Security Enhanced Processor

The TPM solution relies on physical separation to provide an architecture where the service provider can receive assurances of the integrity of both its control processes as well as the packet handling functionality. However, this is not really the ideal platform as it does not take full advantage of the benefits of virtualization technology. Instead, we should be able to run multiple service providers' control processes and the packet handling all on the same processor. In this section we propose an extension to a general-purpose processor that enables it to be shared while still providing assurances of the integrity of the control and data plane. This follows the recent trend of the TPM existing first as a co-processor and more recently being integrated into the chip-set. The next logical step is for security functionality to be integrated into the processor. Though, simply brining the TPM on chip is insufficient.

The research community has proposed small extensions to general-purpose processors to protect the confidentiality and integrity of software. That is, support such that all protected software is integrity checked when loaded into cache from memory and optionally stored in memory in encrypted form (plain text only in the cache). Such examples include SP [18], AEGIS [24], and XOM [19]. We focus our discussion on SP as a concrete example.

With one variant of SP, authority-mode SP [9], an authority can load software onto a device, deploy it in the field, and ensure that the software cannot be modified or read even though the physical device may be in the possession of an adversary. Here, each processor has a non-volatile register internal to the processor which is used for storing a private key called the device root key (DRK). This value is a write-only register (*i.e.,* there is no instruction that allows reading it) that is programmed by the authority in a secure facility. The purpose of this key is that it is used to check the integrity of each cache line when it is loaded into the cache from main memory. For this, when the authority compiles its software it will tag each cache line with a keyed hash of that cache line (keyed with the secret key that it stored as the DRK). This protects the executable from being modified both before launch and after. Without knowing the DRK, any modifications to the executable will not pass the integrity checks when it is loaded into cache during execution. SP also protects against any modifications during execution as well. Whenever a cache line is evicted from cache a new keyed hash is calculated and written to memory along with the cache line. Additionally, to make it so the operating system does not need to be trusted, upon an interrupt, a keyed hash of the registers is performed and stored and the registers are encrypted using the DRK and placed back in the registers they originated from. Then the operating system can perform a context switch as usual and has no opportunity to examine or change the values.

There are two main differences in our use of SP to protect virtual machines that require an extension, which we call virtual-mode SP. First, in our scenario there are multiple authorities: each of the service providers as well as the infrastructure provider, which can protect its own software. A simple solution to this is to have multiple DRK registers in the processor, one for each virtual machine. This, of course, will place a limit on the number of virtual routers that can share the same physical router. As there are limited resources such as memory and CPU, there is a practical limit on the number of virtual machines that can be run anyway. This extension would require extra bits in each cache line to mark which DRK is being used. These bits can be assigned by the virutalization layer at the time the virtual machine is launched. The service provider does not need to know which slot it will run in.

The second difference between authority-mode SP and virtual-mode SP is that authority-mode SP assumes the authority is in physical possession of the device to be able to program the DRK in a secured environment during device initialization. In our case, it is not possible for the service providers to physically access the device. However, public key cryptography can be used to overcome this limitation and enable a service provider to install a DRK remotely. Each processor would include (i) an extra write-only register for holding a private key, (ii) an instruction that will take in an encrypted value to install into one of the DRK registers, and (iii) circuitry to decrypt the encrypted value in the install instruction and store it in decrypted form in one of the DRK slots. The private device key will be written by the router vendor before selling the router to the infrastructure provider. The service provider can then contact the router vendor to obtain the public key, encrypt their DRK, and send it to the infrastructure provider along with their software to execute.

Note that the above does not address availability, as each of the write-only registers can be overwritten, rendering all executables unrunnable. However, a virtual router that is not running is easy to detect and therefore less of a concern than protecting the integrity of the software and data. However, it is an area that does need to be addressed.

Virtual-mode SP can also be used to protect the data plane in a software-based router (a hardware-based router already has a protected data plane). To do this, the packet handling would be run with a separate authority (having its own DRK) and be run in the kernel of the virtualization layer so it can have high performance. This implies that the service provider and infrastructure provider both need to trust the forwarding code. This can come from: (i) an infrastructure provider that is reputable and where the service provider just wants protection against dynamic attacks from other service providers, (ii) an open source implementation, allowing the service provider to analyze the code and verify that it is exactly what the infrastructure provider is using, (iii) an independent, mutually trusted, third party (such as a router vendor) providing the implementation.

Through these extensions to the authority-mode SP architecture, the virtual-mode SP architecture is capable of supporting our idealized router architecture in a tamper-resistant manner. As with the original SP architecture, the virtual-mode SP architecture is minimalist in terms of added circuitry. The required extra circuitry consists of a small number of registers along with encryption/decryption logic. As only portions of the code need to be protected, the encryption/decryption logic will be used infrequently, leading to minimal impact on performance. The original SP architecture showed less than a 1% performance hit for the SPEC benchmark, suggesting a similar overhead for virtual-mode SP is a reasonable estimate [18].

6. DISCUSSION

Ideally, all routers would have the mechanisms presented here that prevent violations from occurring in the first place. This has the advantage that it provides continuous protection, has no storage requirement, and no extra computation overhead to perform the necessary checks. However, it is unrealistic to expect this to be the case. Therefore, service providers will need to monitor the performance and behavior of some of their virtual routers. In fact, the solutions are not mutually exclusive, even on the same router. A hardware-based router could have a virtual-mode SP for protecting the control planes and integrate the functionality of the extended interface card for enabling probing of other routers.

7. CONCLUSIONS

In this paper we described two mechanisms to approach the problem of accountability in a virtualized and hosted infrastructure. In the first approach we used traditional network measurement techniques that are used to monitor the service received by ISPs. We applied this to hosted virtual networks and found that they are limited in the accuracy and efficiency with which they can perform the checks. To overcome these limitations, we proposed an extension to the interface card to enable more efficient, accurate, and finer grained detection capabilities.

In the second approach, rather than detect violations, we discussed re-architecting a router to build assurances into the system such that violations are prevented from occurring in the first place. We first presented a solution based on the TPM chip, with the limitation that physical separation is required to actually achieve the desired assurances. We then proposed an extension to a general-purpose processor to overcome this limitation, finding that with minimal extra circuitry, these assurances can be met and the processor can be shared.

While this is only a first step, accountability in hosted virtual networks is a promising area for future research. With tamper-resistance built into the platform, or sufficient measurement capabilities, the trust between service provider and infrastructure provider does not need to be implicit in the business relationship. This will open the opportunity for more infrastructure providers to co-exist and achieve future networks based on hosted virtual networks.

8. REFERENCES

[1] Amazon elastic compute cloud. `http://aws.amazon.com/ec2/`.

[2] Appnexus security. `https://wiki.appnexus.com/display/documentation/Security`.

[3] I. Avramopoulos and J. Rexford. Stealth probing: efficient data-plane security for IP routing. In *USENIX Annual Technical Conference*, 2006.

[4] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *Proc. NSDI*, 2005.

[5] Cisco ASR 1000 series aggregation services router high availability: Delivering carrier-class services to midrange router. `http://www.cisco.com/en/US/prod/collateral/routers/ps9343/solution_overview_c22-450809_ps9343_Product_Solution_Overview.html`.

[6] Cisco Application eXtension Platform overview. `http://www.cisco.com/en/US/prod/collateral/routers/ps9701/white_paper_c11_459082.html`.

[7] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, IT-22, Nov 1976.

[8] N. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Transactions on Networking*, 9(3), June 2001.

[9] J. Dwoskin and R. B. Lee. Hardware-rooted trust for secure key management and transient trust. In *Proc. CCS*, Oct. 2007.

[10] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley. Evaluating Xen for virtual routers. In *Proc. PMECT*, 2007.

[11] FCC: Commission orders comcast to end discriminatory network management practices. `http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-08-183A1.pdf`.

[12] N. Feamster, L. Gao, and J. Rexford. How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.*, 37(1), 2007.

[13] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proc.SOSP*, October 2003.

[14] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path-quality monitoring in the presence of adversaries. In *Proc. SIGMETRICS*, 2008.

[15] A. Haeberlen, P. Kuznetsov, and P. Druschel. PeerReview: Practical accountability for distributed systems. In *Proc. SOSP*, Oct 2007.

[16] Juniper partner solution development platform. `http://www.juniper.net/us/en/products-services/nos/junos/psdp`.

[17] Juniper networks brings virtualization to the core with industry's most flexible multi-chassis routing system, February 2009. `http://www.juniper.net/company/presscenter/pr/2009/pr_2009_02_02-17_19.html`.

[18] R. B. Lee, P. C. S. Kwan, J. P. McGregor, J. Dwoskin, and Z. Wang. Architecture for protecting critical secrets in microprocessors. In *Proc. ISCA*, June 2005.

[19] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell, and M. Horowitz. Architectural support for copy and tamper resistant software. In *Proc. ASPLOS*, November 2000.

[20] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: An execution infrastructure for tcb minimization. In *Proc. EuroSys*, March 2008.

[21] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(3), 2008.

[22] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, 10(6), Dec 2002.

[23] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and efficient SLA compliance monitoring. In *Proc. SIGCOMM*, 2007.

[24] G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas. Design and implementation of the AEGIS single-chip secure processor using physical random functions. In *Proc. ISCA*, June 2005.

[25] M. M. Swift, S. Martin, H. M. Levy, and S. J. Eggers. Nooks: an architecture for reliable device drivers. In *EW10: Proceedings of the 10th workshop on ACM SIGOPS European workshop*, 2002.

[26] D. Taylor and J. Turner. Diversifying the internet. In *Proc. IEEE GLOBECOM*, Nov. 2005.

[27] Trusted platform module. `https://www.trustedcomputinggroup.org/specs/TPM/`.

[28] Vyatta - network virtualization. `http://www.vyatta.com/products/virtualized.php`.

[29] L. Wang, D. Massey, K. Patel, and L. Zhang. FRTR: A scalable mechanism for global routing table consistency. In *Proc. DSN*, 2004.