Verifiable Cloud Outsourcing for Network Functions
(+ Verifiable Resource Accounting for Cloud Services)

Vyas Sekar

Carnegie Mellon

vNFO joint with
Seyed Fayazbakhsh, Mike Reiter

VRA joint with
Chen Chen, Petros Maniatis, Adrian Perrig, Amit Vasudevan
"Middleboxes" are valuable, but have many pain points!

Based on survey responses + discussions

<table>
<thead>
<tr>
<th>Type of appliance</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewalls</td>
<td>166</td>
</tr>
<tr>
<td>NIDS</td>
<td>127</td>
</tr>
<tr>
<td>Media gateways</td>
<td>110</td>
</tr>
<tr>
<td>Load balancers</td>
<td>67</td>
</tr>
<tr>
<td>Proxies</td>
<td>66</td>
</tr>
<tr>
<td>VPN gateways</td>
<td>45</td>
</tr>
<tr>
<td>WAN Optimizers</td>
<td>44</td>
</tr>
<tr>
<td>Voice gateways</td>
<td>11</td>
</tr>
</tbody>
</table>

- **High Capital Expenses Device Sprawl**
- **High Operating Expenses**
e.g., separate management teams need manual tuning
- **Inflexible, difficult to extend**
  → need for new boxes!

[COMB, NSDI ’12]
Case for Network Function Outsourcing (NFO)

Today:
High CapEx, OpEx,
Delay in innovation

+ Economies of scale, pay-per use
+ Simplifies configuration & deployment

[APLOMB, SIGCOMM ’12]
Concerns with ceding control

Cloud Provider

Correctness properties:
Behavior, Performance, Accounting

Outside scope: Isolation, privacy, ..

[vNFO, HotMiddlebox ’13]
What makes this challenging?

• Lack of visibility into the workload

• Dynamic, traffic-dependent, and proprietary actions of the network functions

• Stochastic effects introduced by the network
Outline

• Motivation for verifiable NFO

• Formalizing properties

• A roadmap for vNFO

• Discussion
Preliminaries:

Given the customer has contracted with the NFO provider to subject its pack-

tes to the environment effects that need to be captured. A bill that represents usage of various resources (e.g., CPU) includes a special symbol that could be a packet that matches a drop rule at a firewall function.

Figure 4: The system parameters necessary to specify the formal properties only require the input/output packets, whereas the snapshot primitive equivalence:

Let the guarantee provided by the black-box primitive be 

\[ \pi_i^{\text{in}}, \pi_i^{\text{out}}, \ldots \]

\[ f : (\Pi \times \Sigma) \rightarrow (\Pi \times \Sigma) \]

The guarantee provided by the black-box primitive equivalence:

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]

\[ f \]
Informally, the correctness properties of NFO are a weak form of correctness, as they just state that there is no loss of information (i.e., the input and output packets of each primitive function are the same). As a starting point, we list two properties that must be guaranteed by each middlebox.

Figure 4: The system parameters necessary to specify the formal correctness properties of NFO.

Black-box Behavioral Correctness

visible to customer

Is there some viable state?

Black-box Pipeline Equivalence:

Behavioral Correctness

Black-box Primitive Equivalence:
Informally, the sequence parameter is an input or an output of a function. We assume that a (i.e., Figure 4: The system parameters necessary to specify the formal was dropped.) Let return a sequence is, if \( i \) \( \in \mathcal{P} \) and \( i \) \( \notin \mathcal{P} \) in the next invocation of \( f \) (e.g., a packet that matches a drop rule at a firewall function).

Let \( f \) \( \in \mathcal{F} \), indicating that the corresponding packet in \( f \) \( \notin \mathcal{F} \) denote the packet and state outputs of \( f \) \( \notin \mathcal{F} \) and \( f \) \( \notin \mathcal{F} \) denote a reference implementation.

As a starting point, we list two properties that must be guaranteed.

1. **Snapshot primitive equivalence:**
   - Given \( f \) \( \in \mathcal{F} \), \( \hat{f} \) \( \in \mathcal{F} \), suppose the NFO customer and provider have contracts that \( f \) and \( \hat{f} \) have output the observed packet.

2. **Snapshot pipeline equivalence:**
   - Suppose the NFO customer and provider have contracts that both \( f \) and \( \hat{f} \) resulted in the observed input-output behavior.

As we will see in the next section, the different properties entail that the state of the middleboxes and the intermediate packets that could have been received was dropped. The guarantee provided by the black-box primitive equivalence requires that there is some instantiation of the internal states of respective functions at the time of processing.

**Snapshot Behavioral Correctness**

Would I get the same output?

\[ \pi_1^{\text{in}} \rightarrow f_1 \rightarrow \sigma_1 \rightarrow \pi_1^{\text{out}} \]

\[ \hat{f}_1 \rightarrow \sigma_1 \rightarrow \pi_1^{\text{out}} \]

\[ \cdots \]

\[ \pi_1^{\text{in}} \rightarrow f_n \rightarrow \sigma_n \rightarrow \pi_1^{\text{out}} \]

\[ \hat{f}_n \rightarrow \sigma_n \rightarrow \pi_1^{\text{out}} \]

\[ \cdots \]

\( \pi_1^{\text{in}} \rightarrow \mathcal{F} \rightarrow \mathcal{M} \rightarrow \mathcal{F} \rightarrow \pi_1^{\text{out}} \)

\( \mathcal{F} \) is computed as a summary statistic (e.g., average delay) over the sequence \( \mathcal{F} \) and \( \mathcal{M} \) is the middleboxes.

visible to customer
Performance Correctness

\[ \pi_1^{in}, \pi_2^{in}, ... \]

\[ f_1 \quad \sigma_1 \quad \cdots \quad \cdots \quad \cdots \quad \sigma_n \quad f_n \]

\[ t_1^{out}, t_2^{out}, ... \]

\[ \pi_1^{out}, \pi_2^{out}, ... \]

Would it really take this long?

\[ \hat{f}_1 \quad \sigma_1 \quad \cdots \quad \cdots \quad \cdots \quad \hat{f}_n \quad \sigma_n \]

\[ t_1' \quad t_2' \quad ... \]

\[ \pi_1'^{out}, \pi_2'^{out}, ... \]

\[ \pi_1^{out}, \pi_2^{out}, ... \]

Observed provider performance ≈ Reference performance
“Did-I” Accounting Correctness

\[ \pi_1^{\text{in}}, \pi_2^{\text{in}}, \ldots \]

\[ f_1 \quad \sigma_1 \quad \ldots \quad f_n \quad \sigma_n \]

\[ \pi_1^{\text{out}}, \pi_2^{\text{out}}, \ldots \]

Did It actually consume?

Charged value of resource \( r \approx \)

Consumption of resource \( r \) by provider
Let \( f_1 \) be associated with invocation and completion timestamps \( \tau_1 \).

As in the earlier case, we are strengthening the correctness properties only require the input/output packets, whereas the snapshot pipeline equivalence.

The guarantee provided by the black-box primitive equivalence:

\[ \hat{f}_n(\pi_1, \pi_2, \ldots, \pi_n) = \pi_1, \pi_2, \ldots, \pi_n \]

Our first goal is to verify the semantic behavior of the outsourced pipeline of functions as follows:

\[ f_n(\pi_1, \pi_2, \ldots, \pi_n) = \pi_1, \pi_2, \ldots, \pi_n \]

Building on this, we extend the correctness properties to the full design of a verifiable NFO by highlighting the system and the middleboxes and the intermediate packets that could have resulted in the observed input-output behavior.

As we will see in the next section, the different properties en-

requirement by additionally binding the internal states of the respective functions at the time of processing.

This property states that if we can log the incoming/outgoing packets at a given middlebox, then there is some instantiation \( \pi \) of primitive functions. We describe this correctness as occurring at two levels:

\[ \pi(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sigma_1, \sigma_2, \ldots, \sigma_n \]

The guarantee provided by the black-box primitive equivalence:

\[ \hat{f}_n(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sigma_1, \sigma_2, \ldots, \sigma_n \]

Snapshot pipeline equivalence:

\[ \pi_1, \pi_2, \ldots, \pi_n \]

As in the earlier case, we are strengthening the correctness properties only require the input/output packets, whereas the snapshot pipeline equivalence.

The guarantee provided by the black-box primitive equivalence:

\[ \hat{f}_n(\pi_1, \pi_2, \ldots, \pi_n) = \pi_1, \pi_2, \ldots, \pi_n \]

Our first goal is to verify the semantic behavior of the outsourced pipeline of functions as follows:

\[ f_n(\pi_1, \pi_2, \ldots, \pi_n) = \pi_1, \pi_2, \ldots, \pi_n \]

Building on this, we extend the correctness properties to the full design of a verifiable NFO by highlighting the system and the middleboxes and the intermediate packets that could have resulted in the observed input-output behavior.

As we will see in the next section, the different properties en-

requirement by additionally binding the internal states of the respective functions at the time of processing.

This property states that if we can log the incoming/outgoing packets at a given middlebox, then there is some instantiation \( \pi \) of primitive functions. We describe this correctness as occurring at two levels:

\[ \pi(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sigma_1, \sigma_2, \ldots, \sigma_n \]

Snapshot pipeline equivalence:

\[ \pi_1, \pi_2, \ldots, \pi_n \]

As in the earlier case, we are strengthening the correctness properties only require the input/output packets, whereas the snapshot pipeline equivalence.

The guarantee provided by the black-box primitive equivalence:

\[ \hat{f}_n(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sigma_1, \sigma_2, \ldots, \sigma_n \]

Our first goal is to verify the semantic behavior of the outsourced pipeline of functions as follows:

\[ f_n(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sigma_1, \sigma_2, \ldots, \sigma_n \]
Outline

• Motivation for NFO + vNFO

• Formalizing vNFO properties

• A roadmap for vNFO

• Discussion
Verifiable NFO (vNFO) Overview

Management Interface

CPU, Mem

Net

CPU, Mem

Each function is implemented as a virtual appliance. NFO provider deploys a trusted shim for logging.
Behavioral + Performance Correctness

Management Interface

CPU, Mem

Net

CPU, Mem

B_CPU, B_Mem, B_Net

Shim logs: every packet, VM state, timestamps per packet
Challenges!

1. Middlebox actions make it difficult to correlate logs
2. Scalability and performance impact due to logging
Potential solutions to challenges

1. Lack of visibility into middlebox actions:
   – Packets may be modified by middleboxes.

   **FlowTags: NSDI ‘14**

2. Scalability
   – Infeasible to log all packets and processing stats.

   **Trajectory Sampling**
Outline

• Motivation for NFO + vNFO

• Formalizing vNFO properties

• A roadmap for vNFO
  – Verifiable accounting for Did-I correctness

• Discussion
“Did-I” Accounting Correctness

\[ \pi_1^{\text{in}}, \pi_2^{\text{in}}, \ldots \rightarrow f_1, \sigma_1 \rightarrow \ldots \rightarrow f_n, \sigma_n \rightarrow \pi_1^{\text{out}}, \pi_2^{\text{out}}, \ldots \]

\[ \text{Did It actually consume?} \]

Charged value of resource \( r \approx \) Consumption of resource \( r \) by provider
Desired Properties

• Image Integrity
  – What is running

• Execution Integrity
  – How it is running

• Accounting Integrity
  – Only chargeable events are accounted
ALIBI Design Overview

- Image Integrity via Attested Instance Launch
- Execution Integrity via Guest-Platform Isolation
- Accounting Integrity via Bracketing
ALIBI architecture

Enhance KVM nested virtualization with resource accounting and protection

- **Advantage**
  - Intercept critical events
  - No modification to L1 hypervisor

- **Current Implementation**
  - CPU accounting
  - Memory accounting

<table>
<thead>
<tr>
<th></th>
<th>L2 Guest</th>
<th>L2 Guest</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM-L1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KVM-L0</td>
<td><strong>Alibi</strong></td>
<td></td>
</tr>
<tr>
<td>HW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Guest-Platform Isolation (Execution Integrity)

• Memory Integrity
  – Isolate memory pages $M$ by instances
  – $M_i$ is writeable only when instance $i$ is running

• Control Flow Integrity
  – Protect program stack by memory protection
  – Monitor and validate guest-CPU state changes

• Storage Integrity
  – Integrity protected file system
Bracketing (Accounting integrity)

- Event Detection
  - Control transfer
  - Memory mapping and unmapping

- Event Attribution
  - Associate resource usage with CPU ownership

- Event Reporting
  - Collect event measurements
  - Store and protect event measurements
CPU Accounting Case Study

- Account CPU cycles directly used by L2 guest
- Protect Time Stamp Counter (TSC) register

<table>
<thead>
<tr>
<th>L2 Guest</th>
<th>L2 Guest</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM-L1</td>
<td></td>
</tr>
<tr>
<td>KVM-L0</td>
<td>Alibi</td>
</tr>
<tr>
<td>HW</td>
<td></td>
</tr>
</tbody>
</table>

- Get CPU cycles, e.g., RDTSC
  - Entry into L2 guest
  - Exit from L2 guest

- Virtualize TSC register

⚠️ Read Timestamp Counter
Overhead of ALIBI

- HW: Intel Xeon E3-1220 (3.10Ghz) with 8GB RAM
- L2/L1: Ubuntu 9.04 (kernel version 2.6.18-10)
  L0: Ubuntu 12.04 (kernel version 3.5.0) and ALIBI

- Single-level virt. vs. native (no virt.) : ~9.5% slowdown
- Nested virt. vs. Single-level virt. : ~6.3% slowdown
- ALIBI additional: ~0.5% slowdown
Outline

• Motivation for verifiable NFO

• Formalizing properties

• A roadmap for vNFO

• Discussion
Discussion

• Is the NFO provider willing to deploy a shim?
• What are the market implications for customers?
• What is the role of SLAs?
• Should-I accounting? I/O accounting?
• Interesting anecdotes of correctness or accounting problems?
• Minimal TCB? without nested?
• Crowdsourcing correctness?
• ...