SafeBricks: Shielding Network Functions in the Cloud

Rishabh Poddar, Chang Lan, Raluca Ada Popa, Sylvia Ratnasamy

UC Berkeley
Network Functions (NFs) in the cloud
Network Functions (NFs) in the cloud

Clients → Enterprise → Cloud Server → Destination
Network Functions (NFs) in the cloud
Problem: Security
Problem: Security

1. Need to protect traffic from the cloud provider

Hackers / curious employees

Clients

Enterprise

Email

Destination

NF providers

Clients
Problem: Security

2. Need to protect traffic from the NF providers

Exfiltration

Email

Destination

NF providers

Clients

Enterprise
Problem: Security

Need to protect **NF code and rulesets** from client enterprise and cloud
Cryptographic solutions do not suffice
Cryptographic solutions do not suffice

1. **Standard encryption**: e.g. end-to-end TLS
Cryptographic solutions do not suffice

1 **Standard encryption**: e.g. end-to-end TLS
   - **Functionality**: Doesn’t allow any computation on encrypted payload
Cryptographic solutions do not suffice

1 **Standard encryption**: e.g. end-to-end TLS

- **Functionality**: Doesn’t allow any computation on encrypted payload
- **Security**: Unencrypted fields (e.g. IP headers) still leak information

Diagram:
- NF providers
- Email
  - Clients
  - Enterprise
  - Destination
Cryptographic solutions do not suffice

2 Specialized encryption: e.g. BlindBox, Embark

[Sherry et al. (SIGCOMM'15)] [Lan et al. (NSDI'16)]
Cryptographic solutions do not suffice

2 Specialized encryption: e.g. BlindBox, Embark
   - Too limited in functionality!

- [✓] Header-based comparisons
- [✓] Keyword matching
- [✗] Regular expressions
- [✗] Cross-flow analysis
- [✗] Statistical computations
How to achieve **full functionality** and our security goals simultaneously?
SafeBricks

1. Protects traffic from the cloud provider
2. Protects traffic from the NF providers
3. Protects NF source code and rulesets from client enterprise and cloud
SafeBricks

1. Protects traffic from the cloud provider
2. Protects traffic from the NF providers
3. Protects NF source code and rulesets from client enterprise and cloud

Hardware enclaves + language-based isolation
Background: Hardware enclaves (e.g. Intel SGX)

- Secure region of memory (**enclaves**) protected by hardware
Background: Hardware enclaves (e.g. Intel SGX)

- Secure region of memory (enclaves) protected by hardware
Background: Hardware enclaves (e.g. Intel SGX)

- Secure region of memory (enclaves) protected by hardware

- Remote attestation by clients

Client

Operating System (untrusted)

Application (untrusted)

Enclave (trusted)

Secret data

Trusted code
Background: Hardware enclaves (e.g. Intel SGX)

- Secure region of memory (enclaves) protected by hardware

- **Remote attestation** by clients
  - Remotely verify enclave contents
Background: Hardware enclaves (e.g. Intel SGX)

- Secure region of memory (enclaves) protected by hardware
- Remote attestation by clients
  - Remotely verify enclave contents
  - Establish a secure channel with enclave
Background: **NetBricks**

[Panda et al. (OSDI’16)]

- Framework for developing *arbitrary NFs*
Background: **NetBricks**

[Panda et al. (OSDI’16)]

- Framework for developing **arbitrary NFs**
  - MapReduce like programming abstractions (operators) for packet processing
Background: **NetBricks**

[Panda et al. (OSDI’16)]

- Framework for developing **arbitrary NFs**
  - MapReduce like programming abstractions (operators) for packet processing
- NFs represented as a **directed graph** with operators as nodes
Background: NetBricks

- Written in **Rust**
  - Fast, safe, zero-copy semantics
  - **Isolates NFs** deployed in a chain while running them in the **same address space**
SafeBricks

1. Protects traffic from the cloud provider
2. Protects traffic from the NF providers
3. Protects NF source code and rulesets from client enterprise and cloud
SafeBricks

1. Protects traffic from the cloud provider

2. Protects traffic from the NF providers

3. Protects NF source code and rulesets from client enterprise and cloud
Outsourcing NFs using hardware enclaves
Outsourcing NFs using hardware enclaves
Outsourcing NFs using hardware enclaves
Outsourcing NFs using hardware enclaves

Cloud (untrusted)

Gateway

Enclave

NF

OS (untrusted)

IPSec

Destination

Enterprise

Clients
Outsourcing NFs using hardware enclaves

Diagram:
- Clients
- Enterprise
- Gateway
- Cloud (untrusted)
- Destination

- Interception proxy
- IPSec
- TLS
- ENCF
- OS (untrusted)
Outsourcing NFs using hardware enclaves

Enterprise

Clients

Gateway

Cloud (untrusted)

Enclave

OS (untrusted)

IPSec

NF

Destination
Outsourcing NFs using hardware enclaves
Outsourcing NFs using hardware enclaves

Cloud (untrusted)

Gateway

IPSec

Packet headers also encrypted

IPSec

OS (untrusted)

Enclave

NF

TLS

TLS

TLS

IPSec
Outsourcing NFs using hardware enclaves

Enterprise

Clients

Gateway

Cloud (untrusted)

Enclave

NF

OS (untrusted)

IPSec

TLS

IPSec

TLS

Destination
Outsourcing NFs using hardware enclaves

SafeBricks also supports “direct” delivery of traffic.
Outsourcing NFs using hardware enclaves

Can use general purpose frameworks, e.g. Haven, Scone
Challenges

1. **Small trusted computing base (TCB)** — enclave should contain minimal amount of code.
Challenges

1. **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

2. **High performance** — Transitioning into / out of enclaves is expensive!
Challenges

1. **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

2. **High performance** — Transitioning into / out of enclaves is expensive!

3. **Illegal enclave instructions** — SGX does not support system calls or instructions that may lead to a VMEXIT
Challenges

1. **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

2. **High performance** — Transitioning into / out of enclaves is expensive!

3. **Illegal enclave instructions** — SGX does not support system calls or instructions that lead to a VMEXIT
NetBricks

- Programming abstractions
- State abstractions

I/O interface

DPDK

Scheduler

Poll for I/O

NICs
Enclave

- **Maximal TCB**: NetBricks stack entirely within enclave
• **Minimal TCB**: Only security-critical components within enclave

• One enclave transition **per node per packet batch**
Intermediate TCB
One enclave **transition per packet batch**
- **Partitioned** NetBricks framework; glue code connects trusted and untrusted code
Two new operators for packet transfer to/from enclave: `toEnclave` and `toHost`.

- **Partitioned** NetBricks framework; glue code connects trusted and untrusted code.
Challenges

1. Small trusted computing base (TCB) — enclave should contain minimal amount of code

2. **High performance** — Transitioning into / out of enclaves is expensive!

3. Illegal enclave instructions — SGX does not support system calls or instructions that lead to a VMEXIT
- One enclave transition per packet batch
SafeBricks enclave

- **Shared queues** in non-enclave heap
- Separate enclave and host threads
- Access queues without exiting enclave — zero enclave transitions

Enclave I/O

Host I/O

NICs

toEnclave
toHost
recv send
Challenges

1. **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

2. **High performance** — Transitioning into / out of enclaves is expensive!

3. **Illegal enclave instructions** — SGX does not support system calls or instructions that lead to a VMEXIT
Observation: NFs in general do not require support for system calls / instructions that lead to VMEXITS.
Observation: NFs in general do not require support for system calls / instructions that lead to VMEXITs, except:

- Logging
- Timestamps (using `rdtsc`)
**Observation:** NFs in general do not require support for system calls / instructions that lead to VMEXITs, except:

- Logging
- Timestamps (using `rdtsc`)

SafeBricks designs **custom solutions** for these operations without enclave transitions.
SafeBricks

1. Protects traffic from the cloud provider

2. Protects traffic from the NF providers

3. Protects NF source code and rulesets from client enterprise and cloud
Problem: Malicious NFs within enclaves

Malicious NFs inside the enclave can exfiltrate or tamper with packets!
Problem: Malicious NFs within enclaves

Malicious NFs inside the enclave can exfiltrate or tamper with packets!

Observation: NFs typically need access only to specific packet fields
- E.g. Firewall needs read-only access to TCP/IP headers
- E.g. NAT needs both read-write access to headers but not to packet payload
Problem: Malicious NFs within enclaves

Malicious NFs inside the enclave can exfiltrate or tamper with packets!

Observation: NFs typically need access only to specific packet fields

- E.g. Firewall needs read-only access to TCP/IP headers
- E.g. NAT needs both read-write access to headers and payload

IP addresses; TCP ports; HTTP payload
Problem: Malicious NFs within enclaves

Malicious NFs inside the enclave can exfiltrate or tamper with packets!

Observation: NFs typically need access only to specific packet fields

- E.g. Firewall needs read-only access to TCP/IP headers
- E.g. NAT needs both read-write access to headers but not to packet payload

SafeBricks enforces least privilege across NFs within the enclave
Least privilege enforcement

Run NFs within the **same** enclave
Least privilege enforcement

Run NFs within the **same** enclave
Least privilege enforcement

Run NFs within the same enclave

- Stitch NFs together interspersed with an operator (\(wList\)) that embeds a vector of permissions in packets — two bits per packet field
Least privilege enforcement

Enforce permissions by *mediating* access to packets using Rust’s *ownership model*
Least privilege enforcement

Enforce permissions by **mediating** access to packets using Rust’s **ownership model**

- Controller module holds ownership of packet buffers
Least privilege enforcement

Enforce permissions by **mediating** access to packets using Rust’s **ownership model**

- Controller module holds ownership of packet buffers
- NFs **borrow references** to packet fields from the Controller, which checks permissions vector in packet
Enforce permissions by **mediating** access to packets using Rust’s **ownership model**

- Controller module holds ownership of packet buffers
- NFs **borrow references** to packet fields from the Controller, which checks permissions vector in packet

Returns an **immutable** reference for read-only access, and a **mutable** reference for write access
Assumption: Trusted compilation of NFs

Least privilege guarantees only hold if NFs are built using a compiler that prohibits unsafe operations!
Assumption: Trusted compilation of NFs

Least privilege guarantees only hold if NFs are built using a **compiler that prohibits unsafe operations!**

E.g. Check array bounds, no pointer arithmetic, no unsafe type casts
Assumption: Trusted compilation of NFs

Least privilege guarantees only hold if NFs are built using a **compiler that prohibits unsafe operations**!

- Possible solution: Client obtains NF source codes from providers and assembles them locally
Assumption: Trusted compilation of NFs

Least privilege guarantees only hold if NFs are built using a **compiler that prohibits unsafe operations!**

- Possible solution: Client obtains NF source codes from providers and assembles them locally
- **Problem:** This violates the confidentiality of NF source code!
SafeBricks

1. Protects traffic from the cloud provider

2. Protects traffic from the NF providers

3. Protects NF source code and rulesets from client enterprise and cloud
Assembling NFs

• **Key idea**: Build NFs within a special “meta”-enclave in the cloud using an agreed upon compiler
Assembling NFs

- **Key idea**: Build NFs within a special “meta”-enclave in the cloud using an agreed upon compiler.

- Both client and NF providers can verify the agreed upon compiler using **remote attestation**.
Assembling NFs

Enterprise

Assembly enclave

Loader

Compiler
Assembling NFs

NF providers

Remote attestation

Loader

Compiler

Assembly enclave

Enterprise
Assembling NFs

NF providers

Assembly enclave

Loader

Compiler

Remote attestation

Enterprise
Assembling NFs

NF providers

Enterprise

Assembly enclave

Loader

Compiler

Remote attestation
Assembling NFs

NF providers

NF code + rulesets

Enterprise

Assembly enclave

Loader

Compiler
Assembling NFs

- NF providers
- NF code + rulesets
- Config
- Assembly enclave
  - Loader
  - Compiler
- Enterprise
Assembling NFs

NF providers

Assembly enclave

Loader

Compiler

NF code + rulesets

Config

Placement of NFs, least privilege policies per NF
Assembling NFs

NF providers

Assembly enclave

Loader

Compiler

NF code + rulesets

Config

Enterprise
Assembling NFs

NF providers

Enterprise

Assembly enclave
- Loader
- Compiler

Deployment enclave
SafeBricks

1. Protects **traffic** from the **cloud provider**
2. Protects **traffic** from the **NF providers**
3. Protects **NF source code and rulesets** from client enterprise and cloud
Performance
Throughput decline across NFs

~0–15% overhead across applications for different packet sizes
DPI performance with increasing no. of rules

Overhead spikes when NF working set **exceeds enclave memory**
DPI performance with increasing no. of rules

Throughput decline

Number of rules

0%  25%  50%  75%  100%
0  5000  10000  15000  20000  25000

Overhead spikes when NF working set exceeds enclave memory

Not a fundamental limitation
SafeBricks uses a combination of **hardware enclaves** and **language-based isolation** to:

- Protect client traffic from the cloud provider
- Enforce least privilege across NFs
- Protect the confidentiality of NF code and rulesets

**Modest overhead** across a range of applications