EndBox: Scalable Middlebox Functions Using Client-Side Trusted Execution

David Goltzsche,1 Signe Rüsch,1 Manuel Nieke,1 Sébastien Vaucher,2 Nico Weichbrodt,1 Valerio Schiavoni,2 Pierre-Louis Aublin,3 Paolo Costa,4 Christof Fetzer,5 Pascal Felber,2 Peter Pietzuch3 and Rüdiger Kapitza1

1TU Braunschweig  goltzsche@ibr.cs.tu-bs.de  @d_goltzsche
2University of Neuchâtel  3Imperial College London  4Microsoft Research  5TU Dresden
What Are Middleboxes?

- **Middleboxes** are essential parts of large networks
  - Example: enterprise networks

- Functions related to **security** or **performance**

- **Current best practice:**
  - central deployment as physical boxes
    - High infrastructure and management costs
      (Sherry et al. SIGCOMM’12)
    - Scalability issues with growing client numbers
What Are Middleboxes?

- **Middleboxes** are essential parts of large networks
  - Example: enterprise networks

- Functions related to **security** or **performance**

- **Current best practice:**
  central deployment as physical boxes
  - High infrastructure and management costs
    (Sherry et al. SIGCOMM’12)
  - Scalability issues with growing client numbers

**Problem:** Middleboxes are necessary for large networks, but come at **high costs** and **do not scale** well with number of clients.
Placement of Middleboxes

(a) Centralised
(b) Cloud-based
(c) Server-side
(d) Client-side

Low infra. cost
Low latency
Good scalability
Trusted infrastructure
Easy administration
Placement of Middleboxes

(a) Centralised
(b) Cloud-based
(c) Server-side
(d) Client-side

<table>
<thead>
<tr>
<th>Feature</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low infra. cost</td>
<td>×</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low latency</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good scalability</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Trusted infrastructure</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Easy administration</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Placement of Middleboxes

<table>
<thead>
<tr>
<th>Placement</th>
<th>Centralised</th>
<th>Cloud-based</th>
<th>Server-side</th>
<th>Client-side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low infra. cost</td>
<td>❌</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Low latency</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Good scalability</td>
<td>❌</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>Trusted infrastructure</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
<td>❌</td>
</tr>
<tr>
<td>Easy administration</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Placement of Middleboxes

<table>
<thead>
<tr>
<th>(a) Centralised</th>
<th>(b) Cloud-based</th>
<th>(c) Server-side</th>
<th>(d) Client-side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low infra. cost</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Low latency</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Good scalability</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Trusted infrastructure</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Easy administration</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>

D. Goltzsche, TU Braunschweig, Germany
Placement of Middleboxes

<table>
<thead>
<tr>
<th>Placement Type</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralised</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cloud-based</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Server-side</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Client-side</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- **Low infra. cost**
- **Low latency**
- **Good scalability**
- **Trusted infrastructure**
- **Easy administration**
**Placement of Middleboxes**

<table>
<thead>
<tr>
<th>Placement Type</th>
<th>Diagram</th>
<th>(a) Centralised</th>
<th>(b) Cloud-based</th>
<th>(c) Server-side</th>
<th>(d) Client-side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td><img src="image1" alt="Centralised Diagram" /></td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Gateway</td>
<td><img src="image2" alt="Centralised Diagram" /></td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Client</td>
<td><img src="image3" alt="Centralised Diagram" /></td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

**With EndBox**

- Low infra. cost: ✗ ✔ ✔ ✔
- Low latency: ✔ ✗ ✔ ✔
- Good scalability: ✗ ✔ ✗ ✔
- Trusted infrastructure: ✔ ✗ ✔ ✔
- Easy administration: ✔ ✔ ✔ ✗

**EndBox** targets enterprise networks and places middleboxes on **untrusted clients**.
Outline

- Introduction to Middleboxes
- Design of ENDBox
- Evaluation of ENDBox
- Related Work
- Conclusion
Approach of EndBox

- **Untrusted clients** can manipulate or circumvent traffic analysis
  - Client traffic routed through **trusted execution environments** (TEEs)
- Inside TEE, packets are **processed, signed and encrypted**
- **Unsigned outgoing traffic dropped** by firewall/gateway (FW/GW)
- **Encrypted incoming traffic** cannot be encrypted outside of TEE
Approach of EndBox

- **Untrusted clients** can manipulate or circumvent traffic analysis
  - Client traffic routed through **trusted execution environments** (TEEs)
- Inside TEE, packets are **processed, signed and encrypted**
- **Unsigned outgoing traffic dropped** by firewall/gateway (FW/GW)
- **Encrypted incoming traffic** cannot be encrypted outside of TEE
Approach of EndBox

- **Untrusted clients** can manipulate or circumvent traffic analysis
  - Client traffic routed through **trusted execution environments** (TEEs)
- Inside TEE, packets are **processed, signed and encrypted**
- **Unsigned outgoing traffic dropped** by firewall/gateway (FW/GW)
- **Encrypted incoming traffic** cannot be encrypted outside of TEE

**EndBox** enforces the routing of application traffic through TEEs deployed on untrusted client machines.
**TEE: Intel SGX in a Nutshell**

- **x86 instruction set extension** introduced with Skylake architecture
- **Creation of trusted execution environments** (TEEs) → **enclaves**
- Execution and data inside enclaves protected from privileged software
- **Hardware-based memory integrity protection and encryption**
- **Remote attestation** of enclaves
- Only CPU is **trusted**

---

**Diagram:**

- Application
  - TEE / Enclave
- Operating System
- Hardware
  - CPU
TEE: Intel SGX in a Nutshell

- x86 instruction set extension introduced with Skylake architecture
- Creation of **trusted execution environments** (TEEs) → **enclaves**
- Execution and data inside enclaves protected from privileged software
- Hardware-based **memory integrity protection and encryption**
- Remote attestation of enclaves
- Only CPU is **trusted**

**Intel SGX** allows the creation of **enclaves**, trusted execution environments (TEEs) protected by hardware.
Implementation of **EndBox** Prototype

![](image)

---

**Client Machine**

**EndBox Client**

1. **Cryptography**
2. **Soft router**
3. **IDPS, Firewall, …**
4. **Fragmentation, Encapsulation**

---

**Packet copied into enclave**

**Execute middlebox function(s)**

**Packet accepted/discarded**

**Packet signed, encrypted and copied out of enclave**

---

Integration of enclaves into OpenVPN client

Utilise Click modular router (Kohler et al. TOCS'/zero.lf) for arbitrary middlebox functions

TaLoS library (Aublin et al. technical report '/one.lf/seven.lf) for in-enclave TLS termination

**E/n.sc/d.scB/o.sc/x.sc** executes middlebox functions inside trusted SGX enclaves embedded into a VPN client and uses the Click modular router.
Implementation of EndBox Prototype

1. Packet copied into enclave

Client Machine

EndBox Client

TEE
Soft router
Cryptography
IDPS, Firewall, ...
Fragmentation, Encapsulation
Implementation of ENDBox Prototype

1. Packet copied into enclave
2. Execute middlebox function(s)
Implementation of ENDBox Prototype

- **Packet copied into enclave**
- **Execute middlebox function(s)**
- **Packet accepted/discarded**

Diagram:

1. Packet copied into enclave
2. Execute middlebox function(s)
3. Packet accepted/discarded
Implementation of \textbf{EndBox} Prototype

1. Packet copied into enclave
2. Execute middlebox function(s)
3. Packet accepted/discarded
4. Packet signed, encrypted and copied out of enclave
Implementation of EndBox Prototype

- Integration of enclaves into **OpenVPN client**
- Utilise **Click modular router** (Kohler et al. TOCS’00) for arbitrary middlebox functions
- TaLoS library (Aublin et al. technical report ’17) for in-enclave **TLS termination**

1. Packet copied into enclave
2. Execute middlebox function(s)
3. Packet accepted/discarded
4. Packet signed, encrypted and copied out of enclave
Implementation of EndBox Prototype

EndBox Client

Integration of enclaves into OpenVPN client
Utilise Click modular router (Kohler et al. TOCS’00) for arbitrary middlebox functions
TaLoS library (Aublin et al. technical report ’17) for in-enclave TLS termination

EndBox executes middlebox functions inside trusted SGX enclaves embedded into a VPN client and uses the Click modular router.

1. Packet copied into enclave
2. Execute middlebox function(s)
3. Packet accepted/discarded
4. Packet signed, encrypted and copied out of enclave

Client Machine

EndBox Client

TEE
Soft router
IDPS, Firewall, …
Cryptography
Fragmentation, Encapsulation
ENDBox Configuration Updates

- Configuration updates are **challenging** with distributed middleboxes

![Diagram of ENDBox Configuration Updates]

Admin uploads encrypted configuration and starts grace period timer.

New version number piggybacked on OpenVPN ping messages.

If necessary, client obtains new configuration file.

Configuration is decrypted and applied.

Ping server with piggybacked version number to prove application.

ENDBox configurations are centrally controlled and enforced.
**EndBox Configuration Updates**

- Configuration updates are **challenging** with distributed middleboxes

1. Admin uploads **encrypted configuration** and starts grace period timer

- Admin uploads encrypted configuration and starts grace period timer
- New version number piggybacked on OpenVPN ping messages
- If necessary, client obtains new configuration file
- Configuration is decrypted and applied
- Ping server with piggybacked version number to prove application configurations are centrally controlled and enforced.

---

D. Goltzsche, TU Braunschweig, Germany | DSN’18: EndBox | Page 8
**EndBox Configuration Updates**

- Configuration updates are **challenging** with distributed middleboxes

1. Admin uploads **encrypted configuration** and starts grace period timer
2. New version number piggybacked on **OpenVPN ping messages**
EndBox Configuration Updates

- Configuration updates are **challenging** with distributed middleboxes

1. Admin uploads **encrypted configuration** and starts grace period timer
2. New version number piggybacked on **OpenVPN ping messages**
3. If necessary, **client obtains new configuration file**
EndBox Configuration Updates

- Configuration updates are **challenging** with distributed middleboxes

1. Admin uploads **encrypted configuration** and starts grace period timer
2. New version number piggybacked on **OpenVPN ping messages**
3. If necessary, **client obtains new configuration file**
4. Configuration is **decrypted and applied**
EndBox Configuration Updates

- Configuration updates are **challenging** with distributed middleboxes

1. Admin uploads **encrypted configuration** and starts grace period timer
2. New version number piggybacked on **OpenVPN ping messages**
3. If necessary, **client obtains new configuration file**
4. Configuration is **decrypted and applied**
5. Ping server with piggybacked version number to **prove application**
**ENDBox Configuration Updates**

- Configuration updates are **challenging** with distributed middleboxes

1. Admin uploads **encrypted configuration** and starts grace period timer
2. New version number piggybacked on **OpenVPN ping messages**
3. If necessary, **client obtains new configuration file**
4. Configuration is **decrypted and applied**
5. Ping server with piggybacked version number to **prove application**

**ENDBox configurations are centrally controlled and enforced.**
Outline

- Introduction to Middleboxes
- Design of ENDBox
- Evaluation of ENDBox
- Related Work
- Conclusion
Evaluation of ENDBox

- **5 client machines** for executing many clients
  - SGX-capable 4-core Xeon v5 CPUs, 32GB RAM

- **2 server machines** as OpenVPN servers
  - non-SGX 4-core Xeon v2 CPUs, 16GB RAM

- **10 Gbps** interconnection (switched network)

**Research questions:**
- What is ENDBox’s impact on latency?
- What throughput can ENDBox achieve?
- Does ENDBox improve scalability?
Latency Depending on Middlebox Placement

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Latency Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>local redirection</td>
<td>4.6%</td>
</tr>
<tr>
<td><strong>EndBox SGX</strong></td>
<td><strong>6.5%</strong></td>
</tr>
<tr>
<td>AWS (Europe)</td>
<td>61%</td>
</tr>
</tbody>
</table>

**Experiment latency overhead**

- **no redirection**
- **local redirection**
- **EndBox SGX**
- **AWS (eu-central-1)**
Latency Depending on Middlebox Placement

Experiment | latency overhead
---|---
local redirection | 4.6%
**EndBox SGX** | **6.5%**
AWS (Europe) | 61%

*EndBox has low impact on latency compared to cloud-based solutions.*
Throughput for Different Middlebox Use cases

<table>
<thead>
<tr>
<th>Use case</th>
<th>throughput overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarding (NOP)</td>
<td>30.6%</td>
</tr>
<tr>
<td>Load balancing (LB)</td>
<td>34.8%</td>
</tr>
<tr>
<td>Firewalling (FW)</td>
<td>29.5%</td>
</tr>
<tr>
<td>Intrusion prev. (IDPS)</td>
<td>39.0%</td>
</tr>
<tr>
<td>DDoS mitigation (DDoS)</td>
<td>37.5%</td>
</tr>
</tbody>
</table>

packet size: 1500 bytes
Throughput for Different Middlebox Use cases

<table>
<thead>
<tr>
<th>Use case</th>
<th>throughput overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarding (NOP)</td>
<td>30.6%</td>
</tr>
<tr>
<td>Load balancing (LB)</td>
<td>34.8%</td>
</tr>
<tr>
<td>Firewalling (FW)</td>
<td>29.5%</td>
</tr>
<tr>
<td>Intrusion prev. (IDPS)</td>
<td>39.0%</td>
</tr>
<tr>
<td>DDoS mitigation (DDoS)</td>
<td>37.5%</td>
</tr>
</tbody>
</table>

**ENDBox has an average throughput overhead of 34.3% for multiple use cases.**
Scalability on Server-side

<table>
<thead>
<tr>
<th>Setup</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanilla OpenVPN</td>
<td>unmodified OpenVPN version</td>
</tr>
<tr>
<td>ENDBox</td>
<td>ENDBox with SGX</td>
</tr>
<tr>
<td>OpenVPN+Click</td>
<td>OpenVPN and server-side Click instance</td>
</tr>
</tbody>
</table>

Clients generate a workload of 200 Mbps each

- vanilla OpenVPN
- ENDBox
- OpenVPN+Click

![Graph showing throughput and CPU usage vs. number of clients]

*E/n.sc/d.scB/o.sc/x.sc scales linearly with the number of clients.*

*E/n.sc/d.scB/o.sc/x.sc has no server-side performance penalty.*

*E/n.sc/d.scB/o.sc/x.sc has a \( \times 8 \) higher throughput compared to a traditional deployment.*

*E/n.sc/d.scB/o.sc/x.sc saves resources on server-side.*
## Scalability on Server-side

<table>
<thead>
<tr>
<th>Setup</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanilla OpenVPN</td>
<td>unmodified OpenVPN version</td>
</tr>
<tr>
<td>ENDBox</td>
<td>ENDBox with SGX</td>
</tr>
<tr>
<td>OpenVPN+Click</td>
<td>OpenVPN and server-side Click instance</td>
</tr>
</tbody>
</table>

Clients generate a workload of 200 Mbps each

- vanilla OpenVPN
- ENDBox
- OpenVPN+Click

---

### EndBox scales linearly with the number of clients.

---

<table>
<thead>
<tr>
<th>Number of clients</th>
<th>Throughput [Gbps]</th>
<th>CPU usage [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>75</td>
</tr>
<tr>
<td>40</td>
<td>8</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

---

D. Goltzsche, TU Braunschweig, Germany | DSN’18: EndBox | Page 13
**Scalability on Server-side**

<table>
<thead>
<tr>
<th>Setup</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanilla OpenVPN</td>
<td>unmodified OpenVPN version</td>
</tr>
<tr>
<td><strong>EndBox</strong></td>
<td><strong>EndBox with SGX</strong></td>
</tr>
<tr>
<td>OpenVPN+Click</td>
<td>OpenVPN and server-side Click instance</td>
</tr>
</tbody>
</table>

Clients generate a workload of 200 Mbps each

- vanilla OpenVPN
- **EndBox**
- OpenVPN+Click

**EndBox scales linearly with the number of clients.**

**EndBox has no server-side performance penalty.**

---

Throughput [Gbps]

CPU usage [%]

Number of clients
## Scalability on Server-side

<table>
<thead>
<tr>
<th>Setup</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanilla OpenVPN</td>
<td>unmodified OpenVPN version</td>
</tr>
<tr>
<td>ENDBox</td>
<td>ENDBox with SGX</td>
</tr>
<tr>
<td>OpenVPN+Click</td>
<td>OpenVPN and server-side Click instance</td>
</tr>
</tbody>
</table>

Clients generate a workload of 200 Mbps each

- vanilla OpenVPN
- ENDBox
- OpenVPN+Click

**ENDBox scales linearly with the number of clients.**

**ENDBox has no server-side performance penalty.**

**ENDBox has a 3.8× higher throughput compared to a traditional deployment.**
## Scalability on Server-side

<table>
<thead>
<tr>
<th>Setup</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanilla OpenVPN</td>
<td>unmodified OpenVPN version</td>
</tr>
<tr>
<td>EndBox</td>
<td>EndBox with SGX</td>
</tr>
<tr>
<td>OpenVPN+Click</td>
<td>OpenVPN and server-side Click instance</td>
</tr>
</tbody>
</table>

Clients generate a workload of 200 Mbps each

- **vanilla OpenVPN**
- **EndBox**
- **OpenVPN+Click**

---

**EndBox scales linearly with the number of clients.**

**EndBox has no server-side performance penalty.**

**EndBox has a 3.8× higher throughput compared to a traditional deployment.**

**EndBox saves resources on server-side.**
Outline

- Introduction to Middleboxes
- Design of \textsc{EndBox}
- Evaluation of \textsc{EndBox}
- Related Work
- Conclusion
Related Work

- Moving middlebox functions to clients has been proposed before

- **Trusted clients** assumed, exception: **ETTM** (Dixon et al. NSDI’11)
  - Based on **Trusted Platform Module** (TPM)
  - Large trusted computing base (TCB) includes hypervisor
  - **Paxos** applied for consensus $\rightarrow$ bad scalability

- Recent work uses SGX, but target **cloud-based trusted middleboxes**
  - **ShieldBox** (Trach et al. SOSR’18)
  - **SafeBricks** (Poddar et al. NSDI’18)
Related Work

- Moving middlebox functions to clients has been proposed before

- **Trusted clients** assumed, exception: **ETTM** (Dixon et al. NSDI’11)
  - Based on **Trusted Platform Module** (TPM)
  - Large trusted computing base (TCB) includes hypervisor
  - **Paxos** applied for consensus → bad scalability

- Recent work uses SGX, but target **cloud-based trusted middleboxes**
  - **ShieldBox** (Trach et al. SOSR’18)
  - **SafeBricks** (Poddar et al. NSDI’18)

---

**ENDBox** is the first approach exploring the deployment of **client-side middleboxes** with recent hardware trends like Intel SGX
Outline

- Introduction to Middleboxes
- Design of ENDBOX
- Evaluation of ENDBOX
- Related Work
- Conclusion
Conclusion

EndBox’s contributions:

- Secure deployment and execution of middlebox functions on untrusted client machines
- Scales linearly with number of clients
- Up to $3.8 \times$ higher throughput
- Centrally controlled and enforced configuration
- Secure analysis of encrypted traffic (see paper!)
- Additional scenario: ISP (see paper!)
Conclusion

**EndBox**’s contributions:

- **Secure deployment and execution** of middlebox functions on untrusted client machines
- **Scales linearly** with number of clients
- Up to *3.8* × higher throughput
- Centrally controlled and enforced configuration
- Secure analysis of encrypted traffic (see paper!)
- Additional scenario: ISP (see paper!)

Thank you for your time! Questions?

goltzsche@ibr.cs.tu-bs.de
@d_goltzsche
github.com/ibr-ds