SENG, the SGX-Enforcing Network Gateway: Authorizing Communication from Shielded Clients

Fabian Schwarz and Christian Rossow
(CISPA Helmholtz Center for Information Security)
Network Firewalls: What are the origins of corporate network communication?

Client Workstations (Enterprise Network)

"traditional" firewall policies

<table>
<thead>
<tr>
<th>src IP/Port</th>
<th>dst IP</th>
<th>dst Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Mail Srv</td>
<td>993</td>
</tr>
<tr>
<td>...</td>
<td>Web Srv</td>
<td>...</td>
</tr>
</tbody>
</table>

Gateway Firewall

External Network
Network Firewalls: What are the origins of corporate network communication?

Client Workstations (Enterprise Network)

Goal: *per-application* policies

<table>
<thead>
<tr>
<th>src App</th>
<th>src IP/Port</th>
<th>dst IP</th>
<th>dst Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mail Client</td>
<td>...</td>
<td>Mail Srv</td>
<td>993</td>
</tr>
<tr>
<td>Browser</td>
<td>...</td>
<td>Web Srv</td>
<td>...</td>
</tr>
</tbody>
</table>

Infected Client

Malware

Mail (infected)

Browser

External Network

Gateway Firewall

Info Leaks

Commands

C&C Server

Information Leaks / Remote Control

sender/receiver application?
The Problem: Secure Traffic-to-Application Attribution is Challenging!

Malware evades traffic-to-application attribution:

Reliable and secure attribution requires:

(I) protection of applications and their traffic from system/MITM attackers

(II) precise, unforgeable application identifiers (exposed to firewall)
Threat Model

- MITM network attackers
- fully compromised client system, only trust hardware (Intel® SGX) and user
- trusted central gateway ("bastion host")

Our Idea(s):
Run applications in TEE and shield network traffic until the perimeter firewall.

++
Root application identities in HW trust anchor and expose them to the gateway services.
Ultimate Goal:
Enable *precise and secure per-application* policies at the perimeter firewall to prevent info leaks / remote control

Easy Deployment
- no client application modifications
- compatible with existing firewalls and gateway services
Infected Client

Malware

Application (binary, libs)
How to shield client applications and their connections?
Application (binary, libs)
SENG Runtime: Shielding Client Application Traffic from System-level Attacker

- shields app in SGX Enclave via library OS (Graphene-SGX)
- dynamic loading, threading, syscalls, and file system shield
- BUT: relies on host network stack
- SENG Runtime shields app connections
SENG Runtime: Shielding Client Application Traffic from System-level Attackers

- shields app in SGX Enclave via library OS (Graphene-SGX)
- dynamic loading, threading, syscalls, and file system shield
- BUT: relies on host network stack

- SENG Runtime shields app connections
- trusted TCP/IP network stack (lwIP)
- provides trusted Socket + DNS API

fabian.schwarz@cispa.saarland — SENG, the SGX-Enforcing Network Gateway
SENG Runtime: Shielding Client Application Traffic from System-level Attackers

- SENG Runtime shields app connections
- trusted TCP/IP network stack (lwIP)
- provides trusted Socket + DNS API
- DTLS protected IP-level tunnel

![Diagram showing SENG Runtime components and interactions]

- App data
- IP packets
- DTLS records
- SGX boundary
- System calls (incl. `send/recv`)

- SGX Enclave
- Application (binary, libs)
- lwIP
- tunnel module
- Graphene-SGX
- Linux Kernel
- Kernel Space
- User Space
- Untrusted IP Packet
- DTLS Record
- Trusted IP Packet
- Application Data

fabian.schwarz@cispa.saarland — SENG, the SGX-Enforcing Network Gateway
Our Idea(s):

Run applications in TEE and shield network traffic until the perimeter firewall.

Root application identities in HW trust anchor and expose them to the gateway services.
SENG Runtime: Shielding Client Application Traffic from System-level Attackers

How to centrally attribute and authorize shielded traffic?

- SENG Runtime shields app connections
- Trusted TCP/IP network stack (lwIP)
- Provides trusted Socket + DNS API
- DTLS protected IP-level tunnel

---

fabian.schwarz@cispa.saarland — SENG, the SGX-Enforcing Network Gateway
SENG Server: Shielded Traffic Attribution and Authorization

Enterprise Network

Infected Client
- Malware
- SGX Enclave
  - Shielded Application

External Network (Internet)

Gateway
- SENG Server
- Firewall
- *keys + attestation*
- *allowlist*
- *IP assignment*

DMZ / Trusted Server Subnet

Untrusted Client Subnetwork

External Servers

Untrusted IP Packet (Host IP)
- DTLS Record
- Trusted IP Packet (Enclave IP)
- Application Data

Shielded App Network Packet

- :trusted
- :untrusted

- [enclave IP]

- attested tunnel

Malware

Untrusted Client

SGX Enclave

Infected Client

Shielded Application

Firewall

Untrusted IP Packet
SENG Server: Shielded Traffic Attribution and Authorization

Enterprise Network

Infected Client

Malware

SGX Enclave

Shielded Application

[enclave IP]

malicious traffic

Gateway

SENG Server

attested tunnel

Firewall

Untrusted Client Subnetwork

DMZ / Trusted Server Subnet

External Network (Internet)

External Servers

Infected Client

attested tunnel

MITM

Malware

 fabian.schwarz@cispa.saarland — SENG, the SGX-Enforcing Network Gateway
Our Idea(s):

- Run applications in TEE and shield network traffic until the perimeter firewall.
- Root application identities in HW trust anchor and expose them to the gateway services.

fabian.schwarz@cispa.saarland — SENG, the SGX-Enforcing Network Gateway
How to define and enforce per-application firewall rules?
SENG’s Enclave Subnetworks: per-application firewall rules

"traditional" firewall rules

<table>
<thead>
<tr>
<th>No.</th>
<th>source</th>
<th>destination</th>
<th>dst Port</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$workstations</td>
<td>$_external</td>
<td>443</td>
<td>…</td>
</tr>
<tr>
<td>2</td>
<td>$workstations</td>
<td>$_SQL_DB</td>
<td>5432</td>
<td>…</td>
</tr>
<tr>
<td>3</td>
<td>$any</td>
<td>$_FTP_Srv</td>
<td>989, 990</td>
<td>…</td>
</tr>
</tbody>
</table>

SENG’s per-application rules (with enclave subnetworks)

<table>
<thead>
<tr>
<th>No.</th>
<th>source</th>
<th>destination</th>
<th>dst Port</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$ws_Firefox72</td>
<td>$_external</td>
<td>443</td>
<td>…</td>
</tr>
<tr>
<td>2</td>
<td>$ws_psql_tls</td>
<td>$_SQL_DB</td>
<td>5432</td>
<td>…</td>
</tr>
<tr>
<td>3</td>
<td>$any_filezilla</td>
<td>$_FTP_Srv</td>
<td>989, 990</td>
<td>…</td>
</tr>
</tbody>
</table>

Firewalls enforce SENG’s per-application policies on the application-specific subnetworks.
**SENG’s Enclave Subnetworks: per-application firewall rules**

<table>
<thead>
<tr>
<th>No.</th>
<th>Source</th>
<th>Destination</th>
<th>dst Port</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$workstations</td>
<td>$_external</td>
<td>443</td>
<td>…</td>
</tr>
<tr>
<td>2</td>
<td>$workstations</td>
<td>extPsqlDB</td>
<td>5432</td>
<td>…</td>
</tr>
<tr>
<td>3</td>
<td>$any</td>
<td>$_FTP_Srv</td>
<td>989, 990</td>
<td>…</td>
</tr>
</tbody>
</table>

### "traditional" firewall rules

<table>
<thead>
<tr>
<th>No.</th>
<th>Source</th>
<th>Destination</th>
<th>dst Port</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$ws_Firefox72</td>
<td>$_external</td>
<td>443</td>
<td>…</td>
</tr>
<tr>
<td>2</td>
<td>$ws_psql_tls</td>
<td>$_extPsqlDB</td>
<td>5432</td>
<td>…</td>
</tr>
<tr>
<td>3</td>
<td>$any_filezilla</td>
<td>$_FTP_Srv</td>
<td>989, 990</td>
<td>…</td>
</tr>
</tbody>
</table>

**Ultimate Goal:**

Enable **precise and secure per-application policies** at the perimeter firewall to prevent info leaks / remote control

**Easy Deployment**

- no client application modifications
- compatible with existing firewalls and gateway services

Firewalls enforce SENG’s per-application policies on the application-specific subnetworks.

---

fabian.schwarz@cispa.saarland — SENG, the SGX-Enforcing Network Gateway
How does SENG perform compared ... to Graphene-SGX? ... to Native?
SENG Runtime Performance: Client Applications

- "native": Linux native
- "pure": Graphene-SGX (LibOS)
- local setup, 1 Gbps LAN

TCP throughput (iPerf3):
- native == pure (avg. ~ 926 Mbps)
- SENG: ~ 93 - 97 % (avg. ~ 868 Mbps)

HTTP download (cURL):
- SENG: 8.8 - 14.1 % overhead (< 1sec)
- (files: 1 MB, 10 MB, ..., 1 GB)
NGINX Server Application: SENG Runtime performance

HTTP response latency (NGINX):
- app: NGINX, bench with wrk2
- native: ~40k req/sec
- SENG/pure: ~15k req/sec (~37.5% of native)

Problem:
Graphene-SGX (our version) only supports synchronous syscalls, no batch mode

==> Will faster primitives help?
Problem:
Our Graphene-SGX version only supports synchronous syscalls, no batch mode

==> Will faster primitives help?
Problem:
Our Graphene-SGX version only supports synchronous syscalls, no batch mode

==> Will faster primitives help?

SENG SDK ("SENG-sdk"):  
- runtime alternative based on Intel® SGX SDK (no LibOS)  
- ~36k req/sec (➕2.4x SENG, ~90% of native)
SENG SDK ("SENG-sdk"):
- runtime alternative based on Intel® SGX SDK (no LibOS)
- ~36k req/sec (~2.4x SENG, ~90% of native)

Fabian Schwarz (Twitter): https://github.com/sengsgx