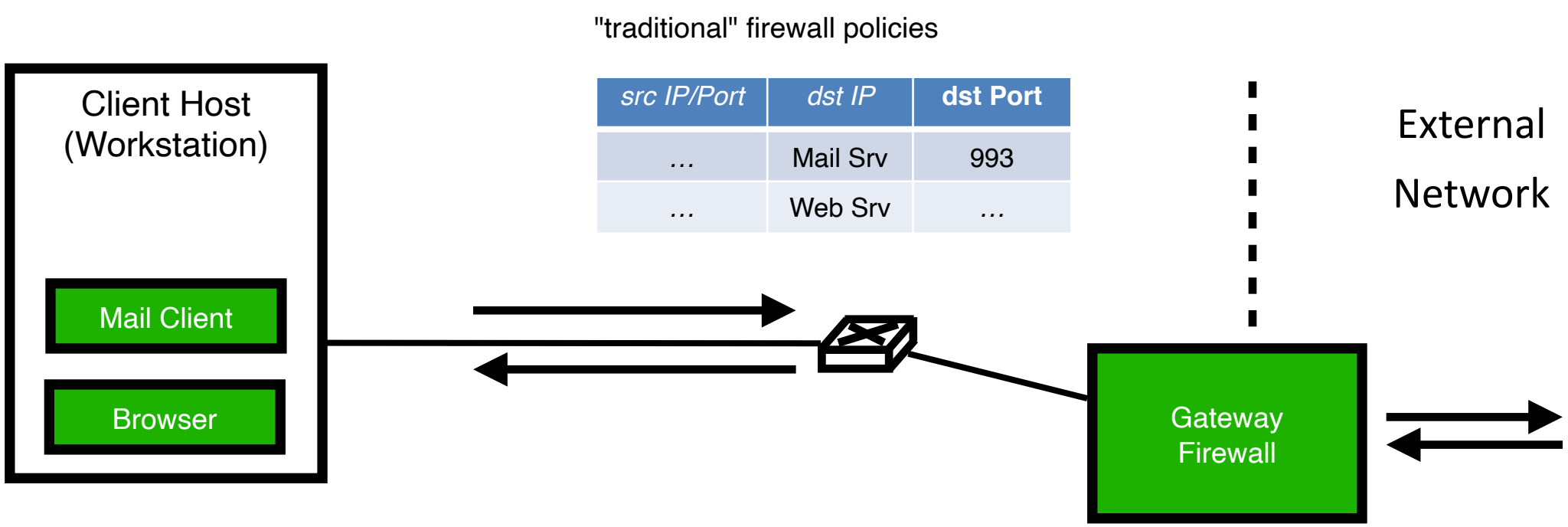
SENG, the SGX-Enforcing Network Gateway: Authorizing Communication from Shielded Clients

Fabian Schwarz and Christian Rossow

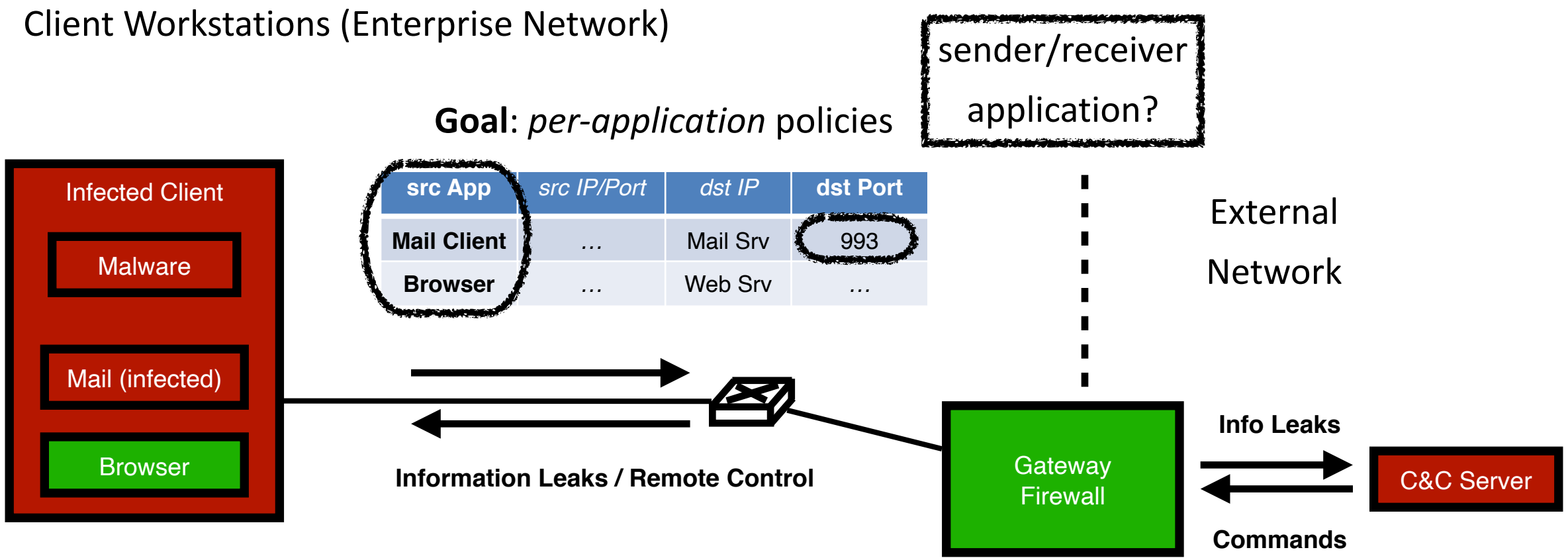
(CISPA Helmholtz Center for Information Security)

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

Client Workstations (Enterprise Network)

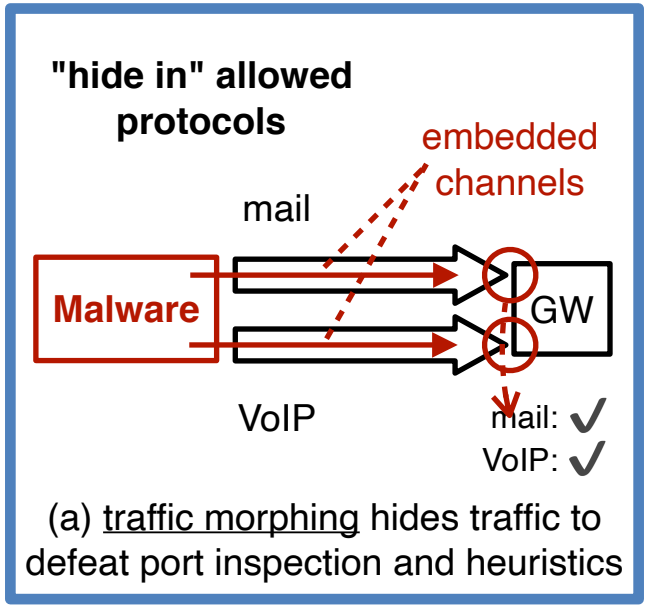


Network Firewalls: What are the origins of corporate network communication?

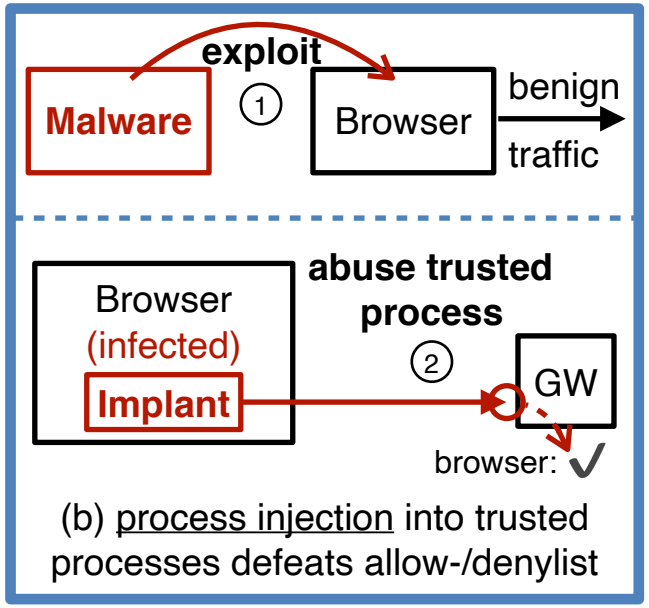


The Problem: Secure Traffic-to-Application Attribution is Challenging!

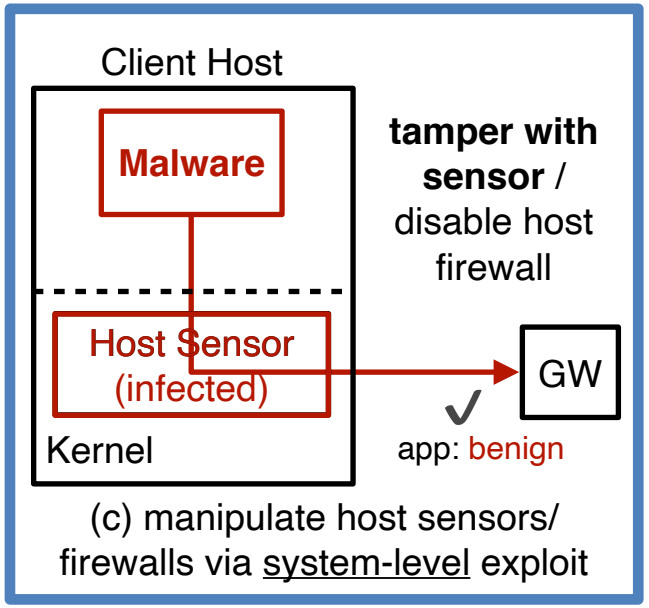
Malware evades traffic-to-application attribution:



(a) traffic morphing hides traffic to defeat port inspection and heuristics



(b) process injection into trusted processes defeats allow-/denylist



(c) manipulate host sensors/ firewalls via system-level exploit

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

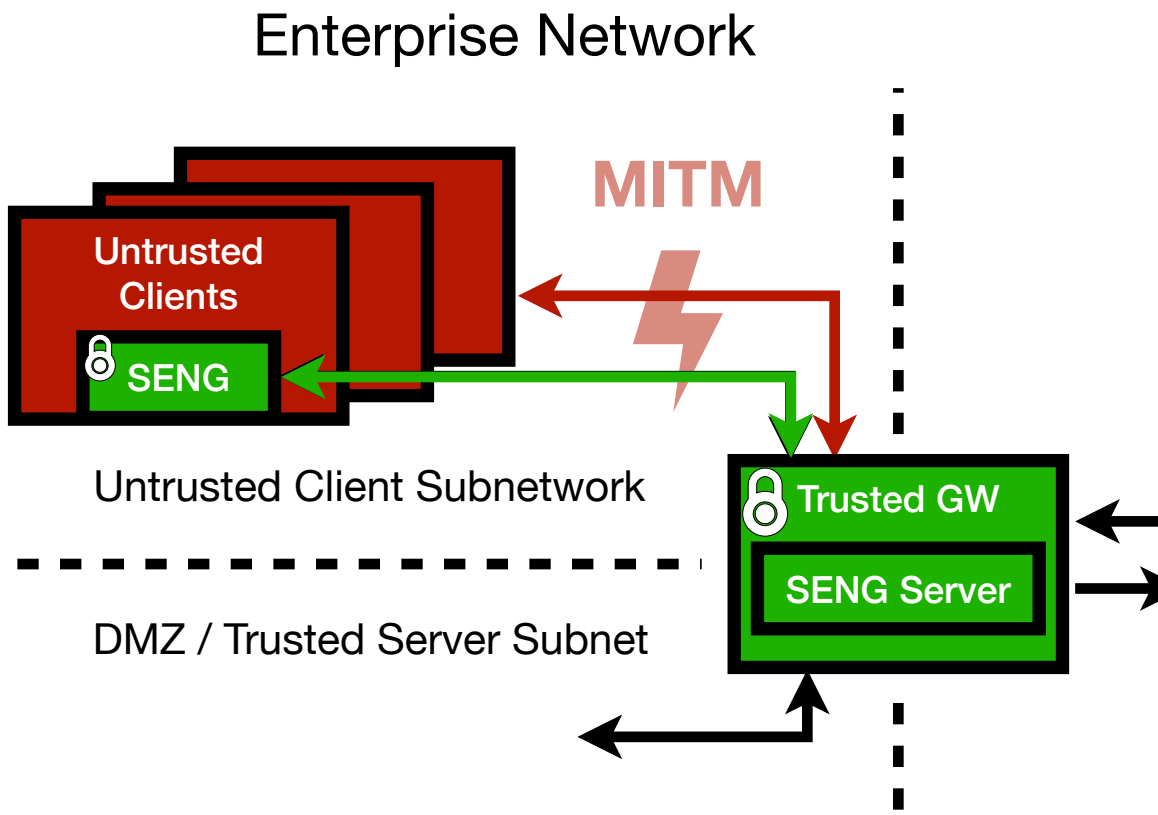
- MITM network attackers
- fully compromised client system, only trust hardware (Intel® SGX) and user
- trusted central gateway ("bastion host")

Our Idea(s):

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

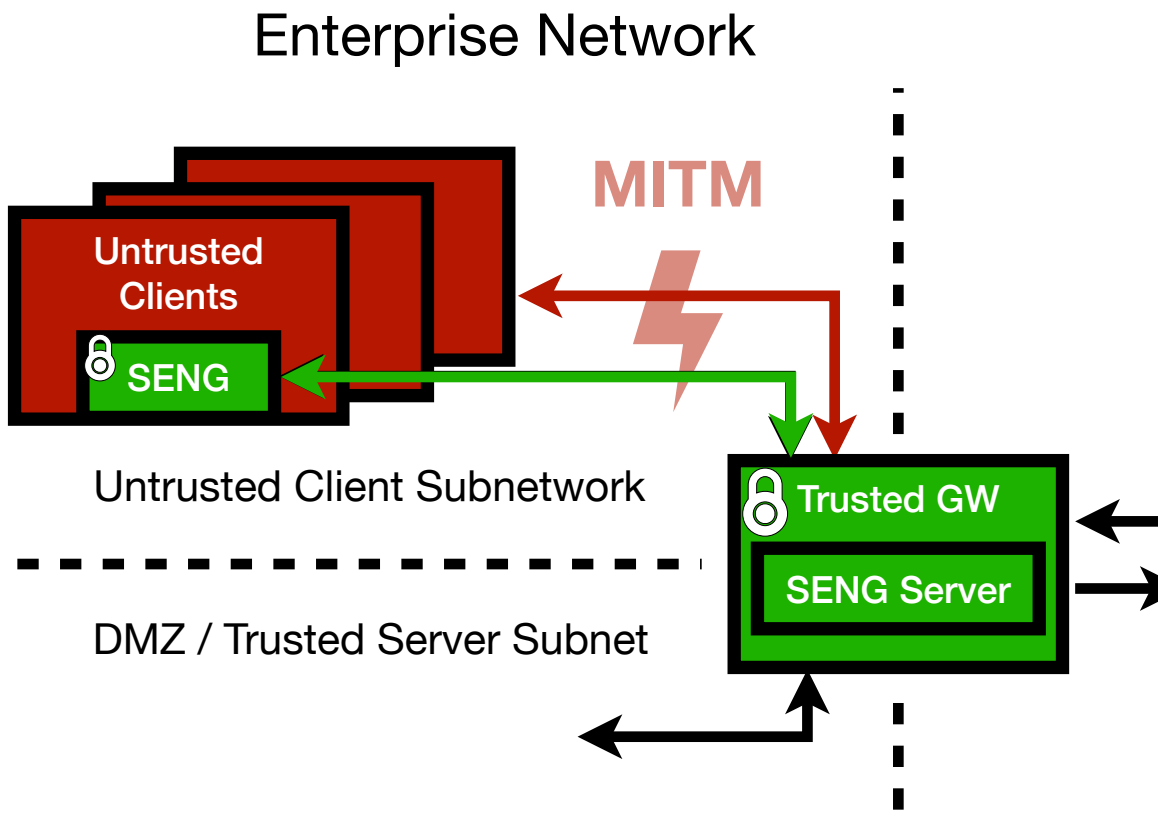
++

Root application identities in HW trust anchor and expose them to the gateway services.



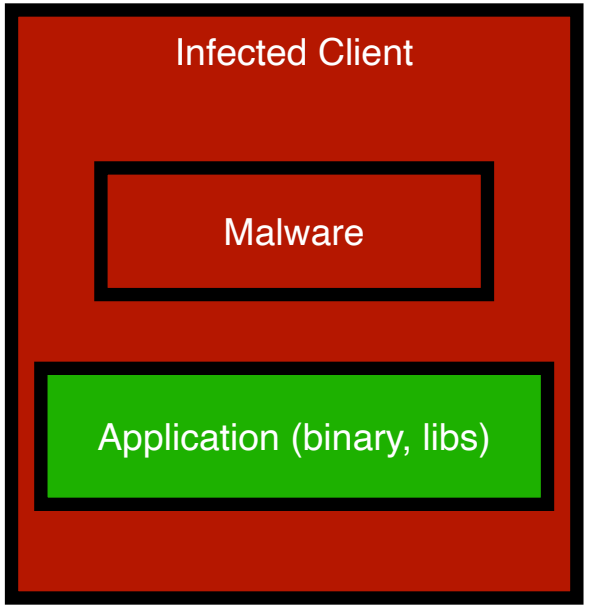
Ultimate Goal:

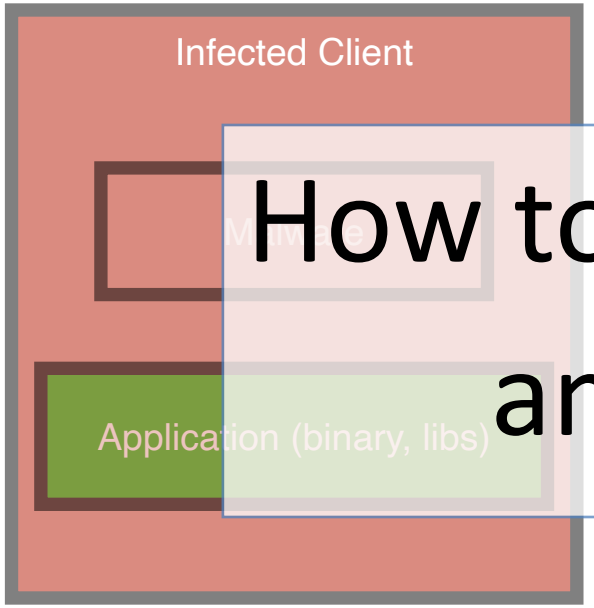
Enable *precise and secure per-application* policies at the perimeter firewall to prevent info leaks / remote control



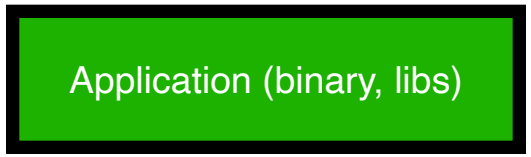
Easy Deployment

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

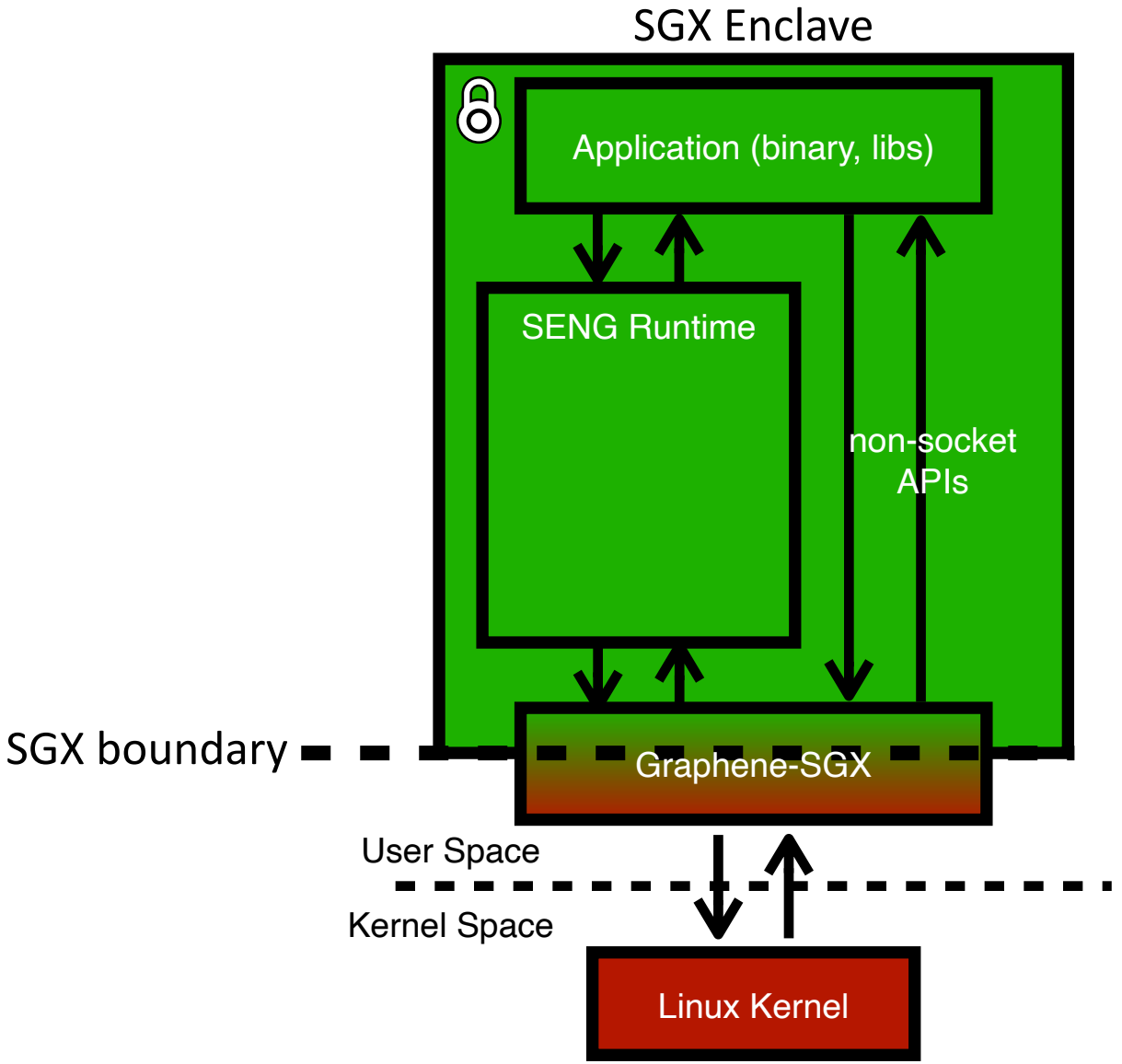

Infected Client
Malware
Application (binary, libs)



How to shield client applications and their connections?


Application (binary, libs)

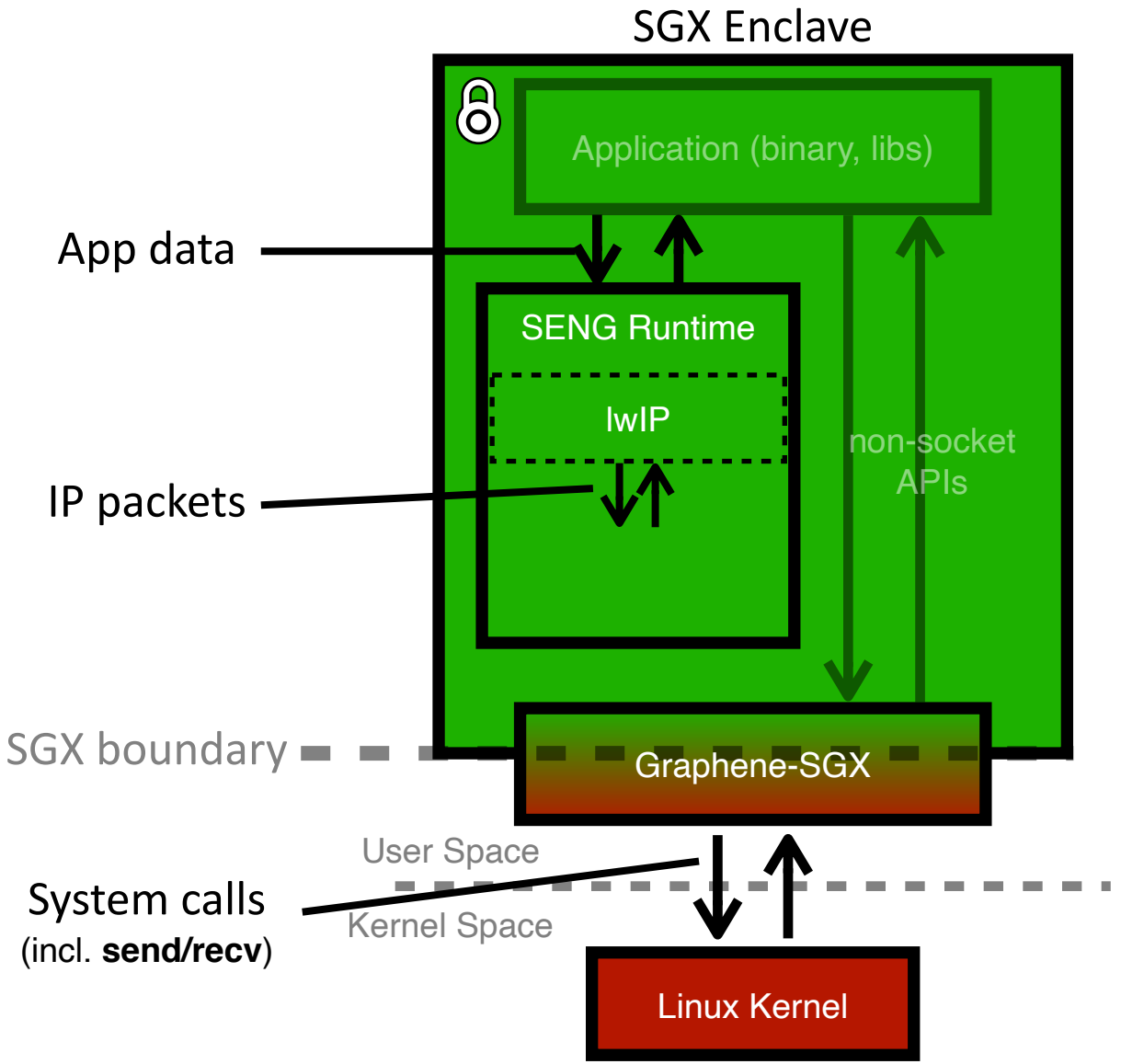
SENG Runtime: Shielding Client Application Traffic from System-level Attackers



- 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

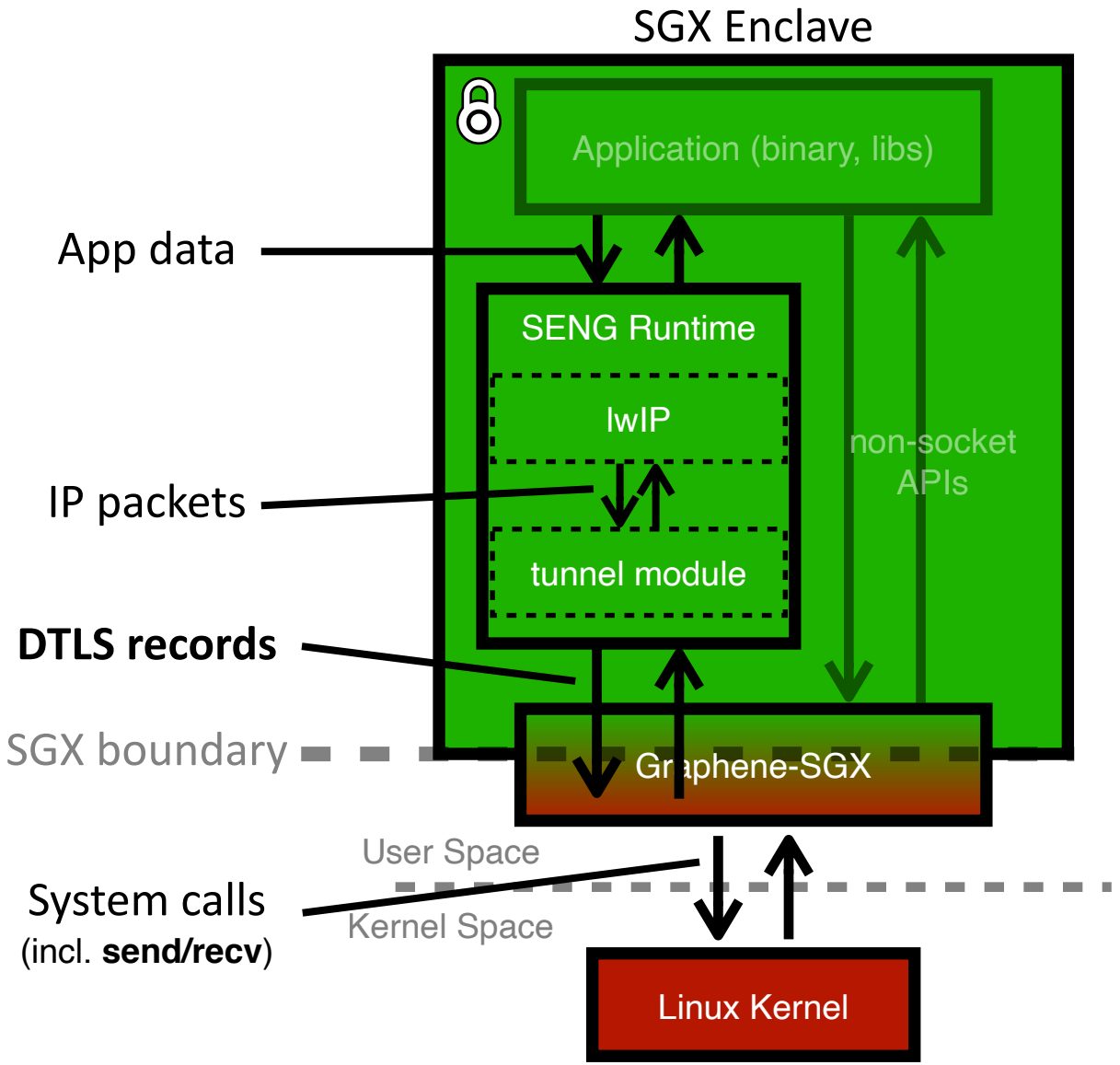
SENG Runtime: Shielding Client Application Traffic from System-level Attackers

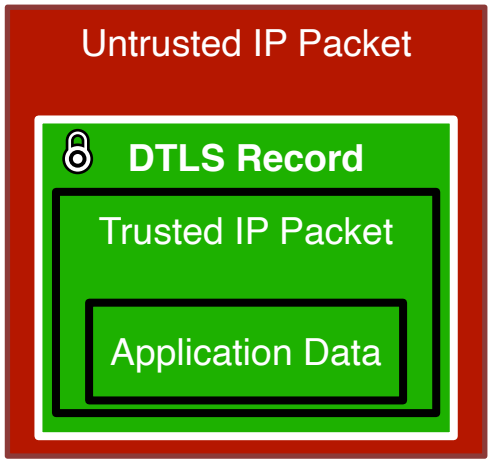


- shields app in SGX Enclave via library OS (Graphene-SGX)
- dynamic loading, threading, syscalls, and file system shield
- **BUT: relies on host network stack**

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

SENG Runtime: Shielding Client Application Traffic from System-level Attackers

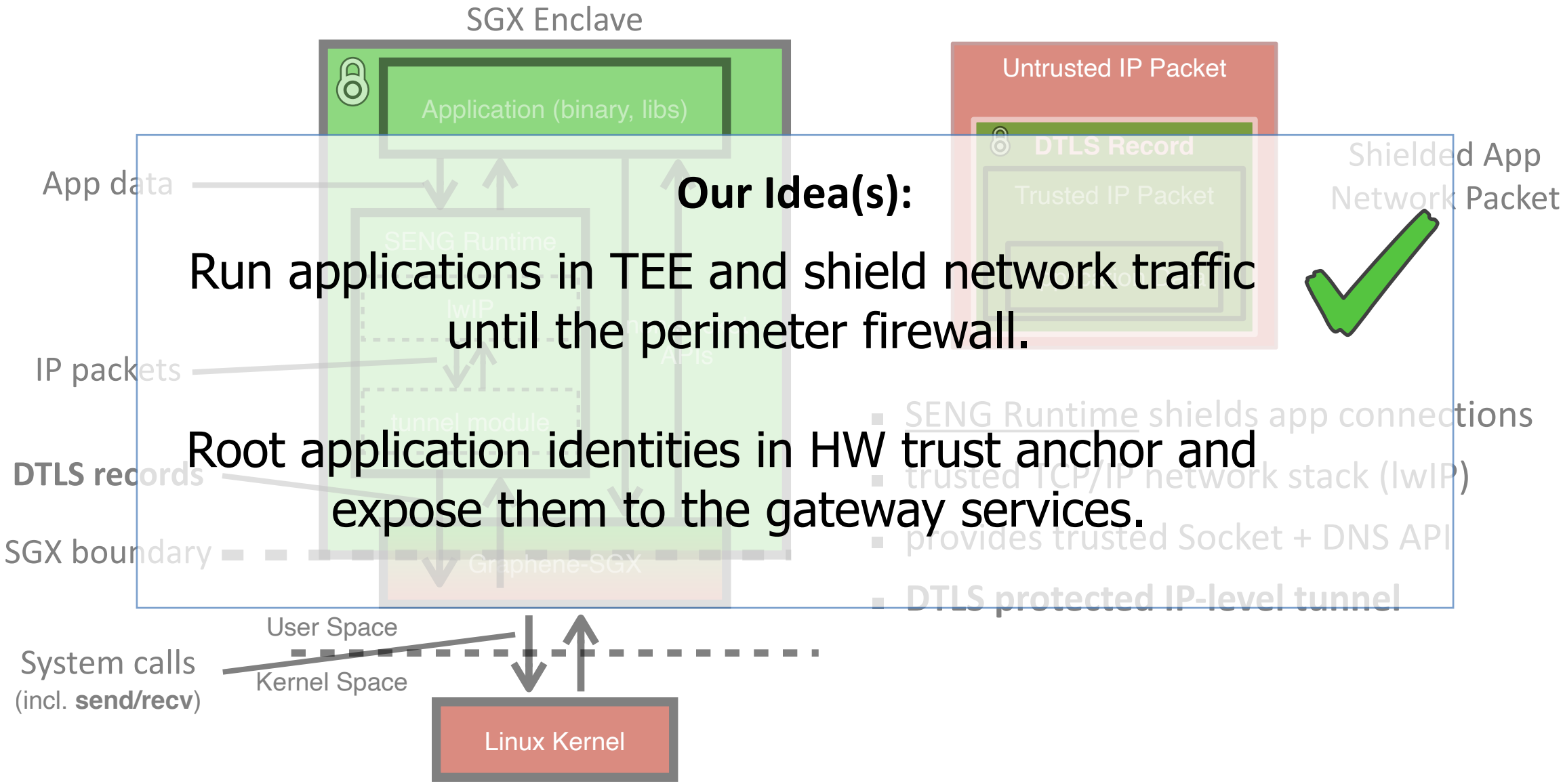




Shielded App Network Packet

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

SENG Runtime: Shielding Client Application Traffic from System-level Attackers

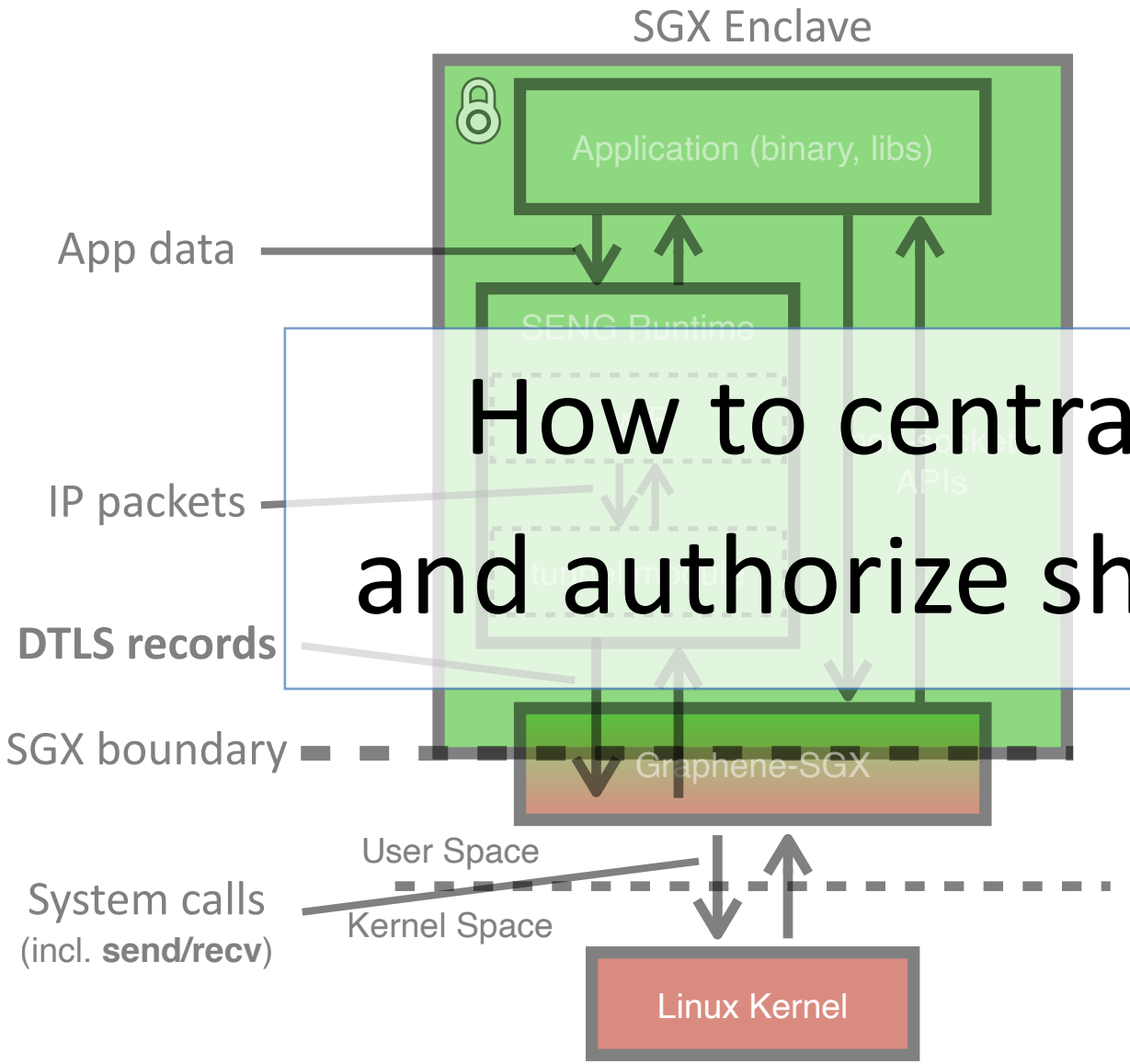


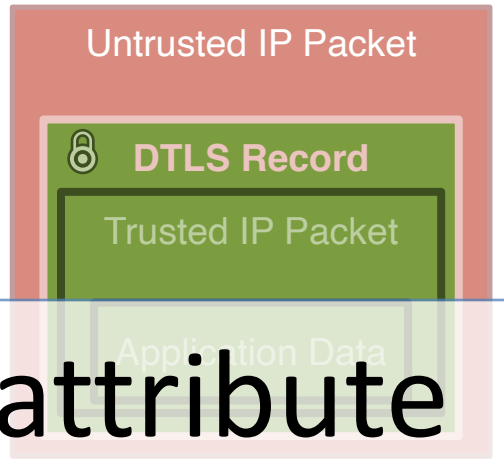
Our Idea(s):

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



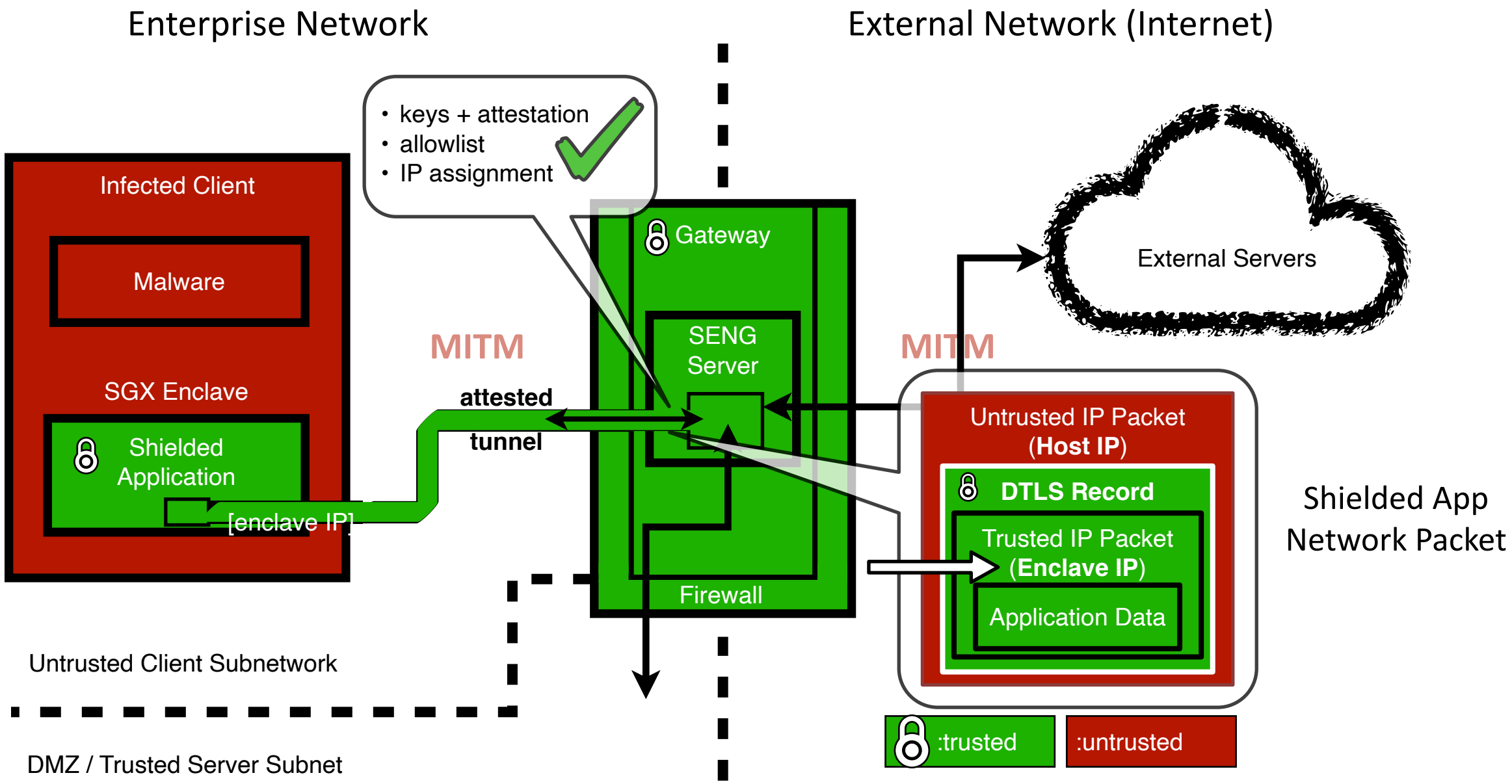


Shielded App Network Packet

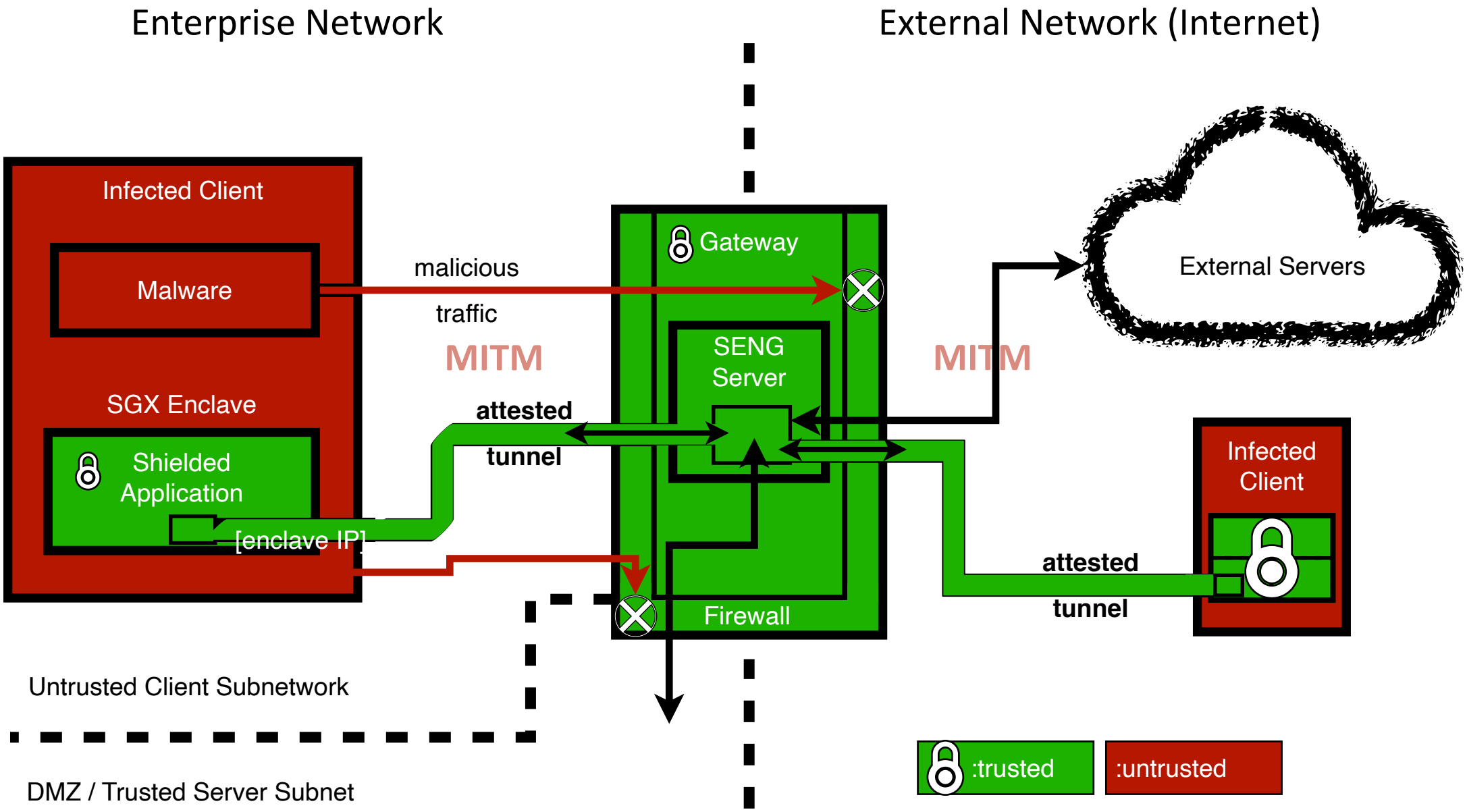
How to centrally attribute and authorize shielded traffic?

- trusted TCP/IP network stack (lwIP)
- provides trusted Socket + DNS API
- **DTLS protected IP-level tunnel**

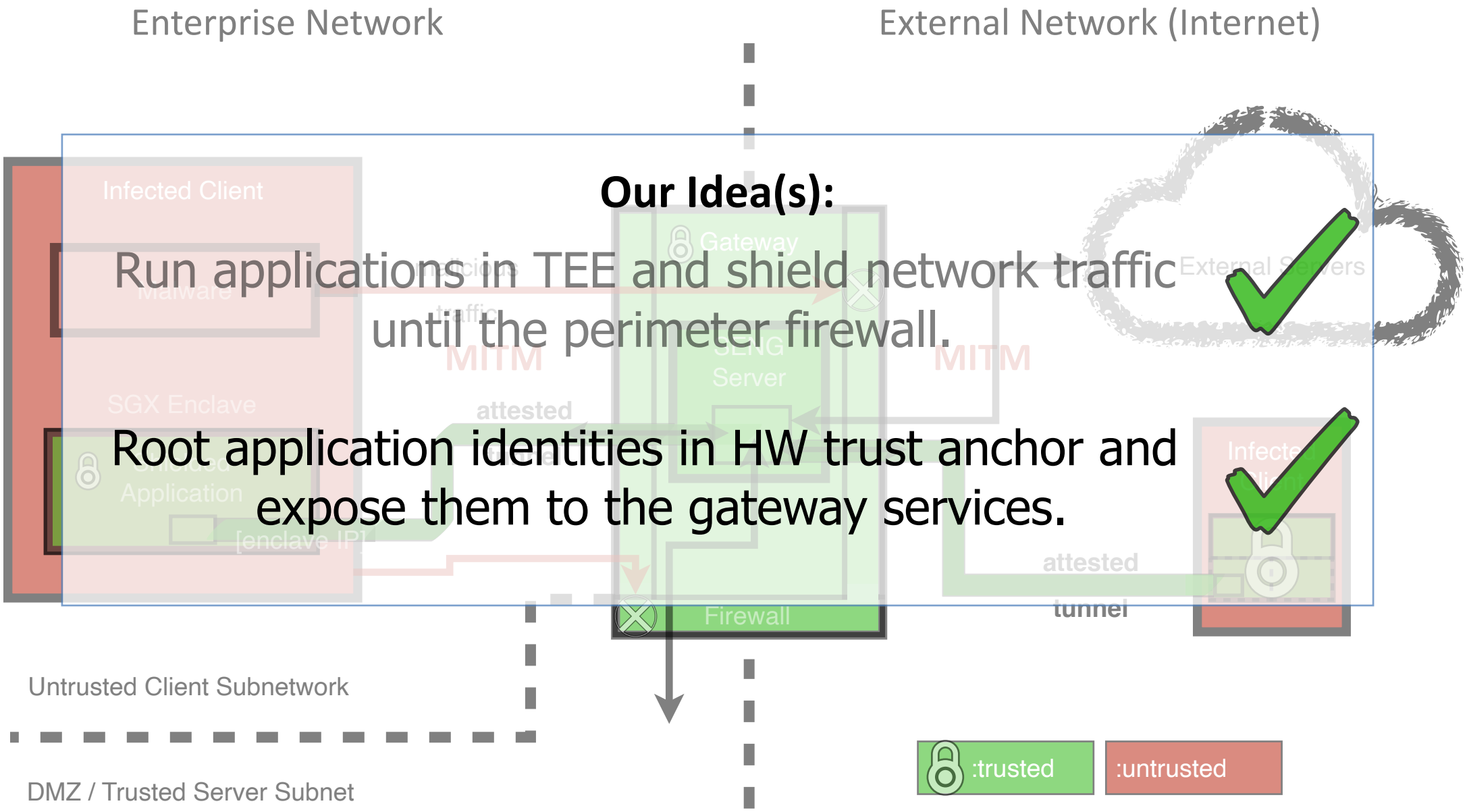
SENG Server: Shielded Traffic Attribution and Authorization



SENG Server: Shielded Traffic Attribution and Authorization



SENG Server: Shielded Traffic Attribution and Authorization



Our Idea(s):

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

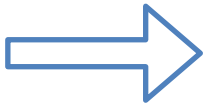
Root application identities in HW trust anchor and expose them to the gateway services.

How to define and enforce per-application firewall rules?

SENG's Enclave Subnetworks: per-application firewall rules

"traditional" firewall rules

No.	**source**	destination	dst Port	…
1	$_workstations	**$_external**	443	…
2	$_workstations	**$_SQL_DB**	5432	…
3	**$_any**	$_FTP_Srv	989, 990	…



SENG's *per-application* rules (with enclave subnetworks)

No.	**source**	destination	dst Port	…
1	**$ws_Firefox72**	$_external	443	…
2	**$ws_psql_tls**	$_SQL_DB	5432	…
3	**$any_filezilla**	$_FTP_Srv	989, 990	…

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

SENG's Enclave Subnetworks: per-application firewall rules

"traditional" firewall rules

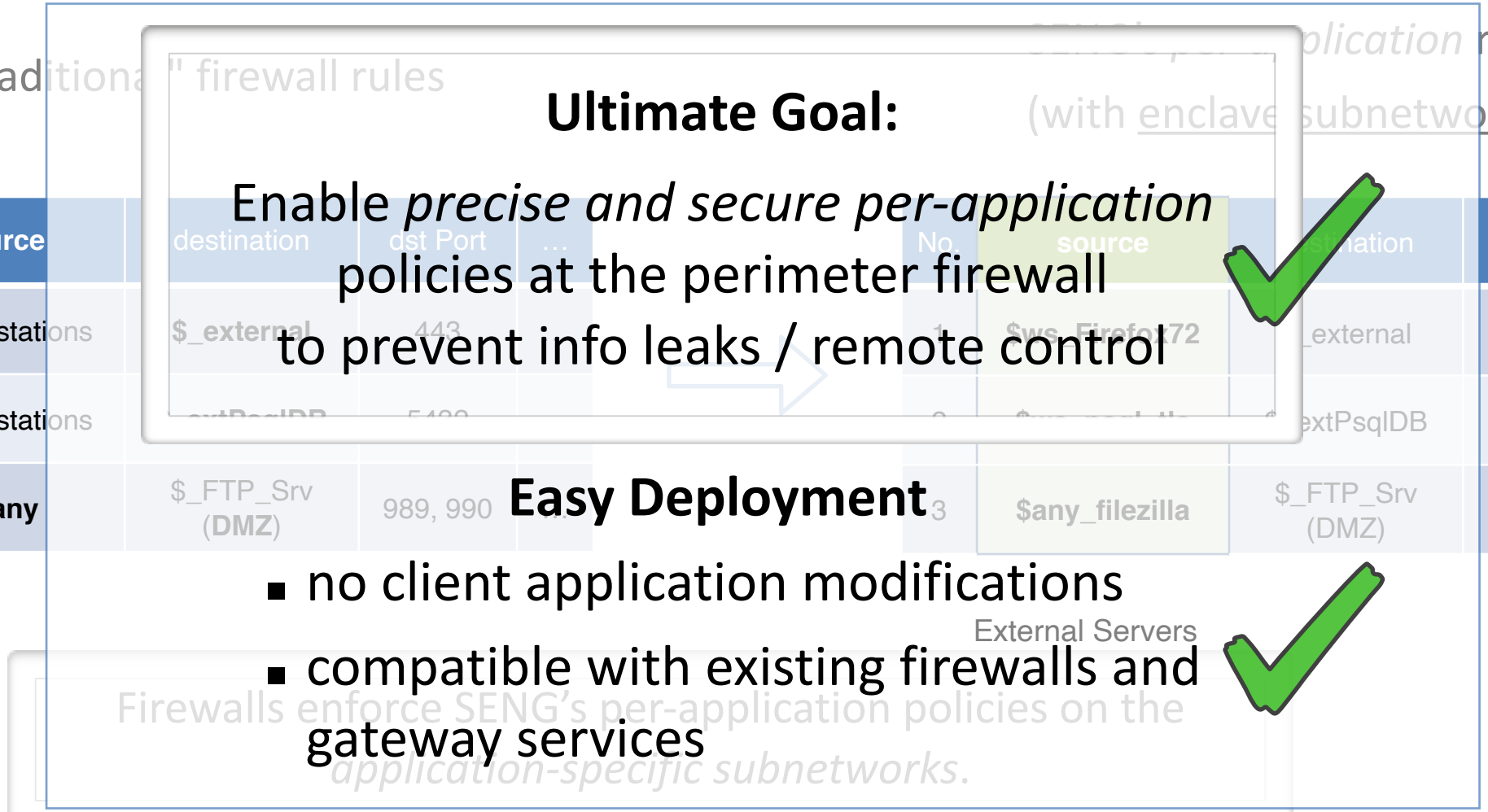
No.	source	destination	dst Port	…
1	$_workstations	$_external	443	…
2	$_workstations	extPsqlDB	5432	…
3	**$_any**	$_FTP_Srv (**DMZ**)	989, 990	…

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



How does SENG perform compared … to Graphene-SGX? … to Native?

SENG Runtime Performance: Client Applications

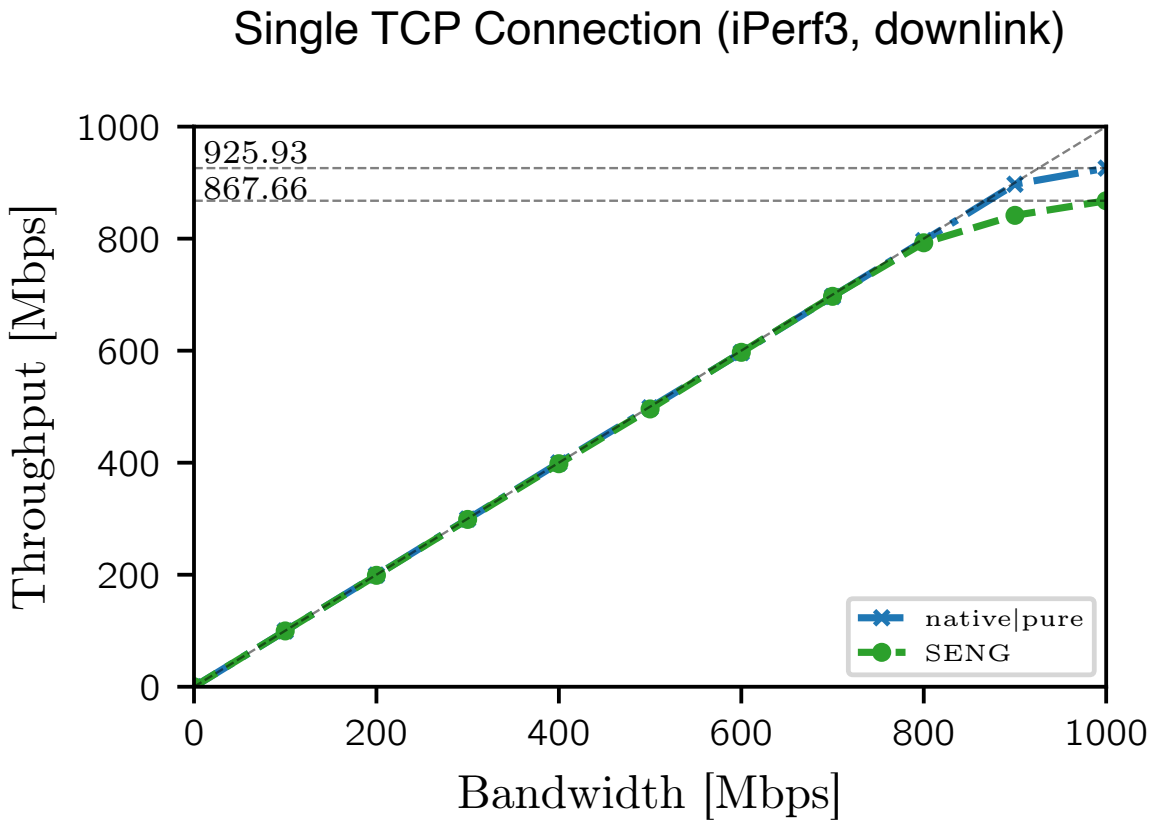
- "native": Linux native
- "pure": Graphene-SGX (LibOS)
- local setup, 1 Gbps LAN

<u>TCP throughput</u> (iPerf3):

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

<u>HTTP download</u> (cURL):

- SENG: 8.8 - 14.1 % overhead (< 1sec)
- (files: 1 MB, 10 MB, …, 1 GB)



NGINX Server Application: SENG Runtime performance

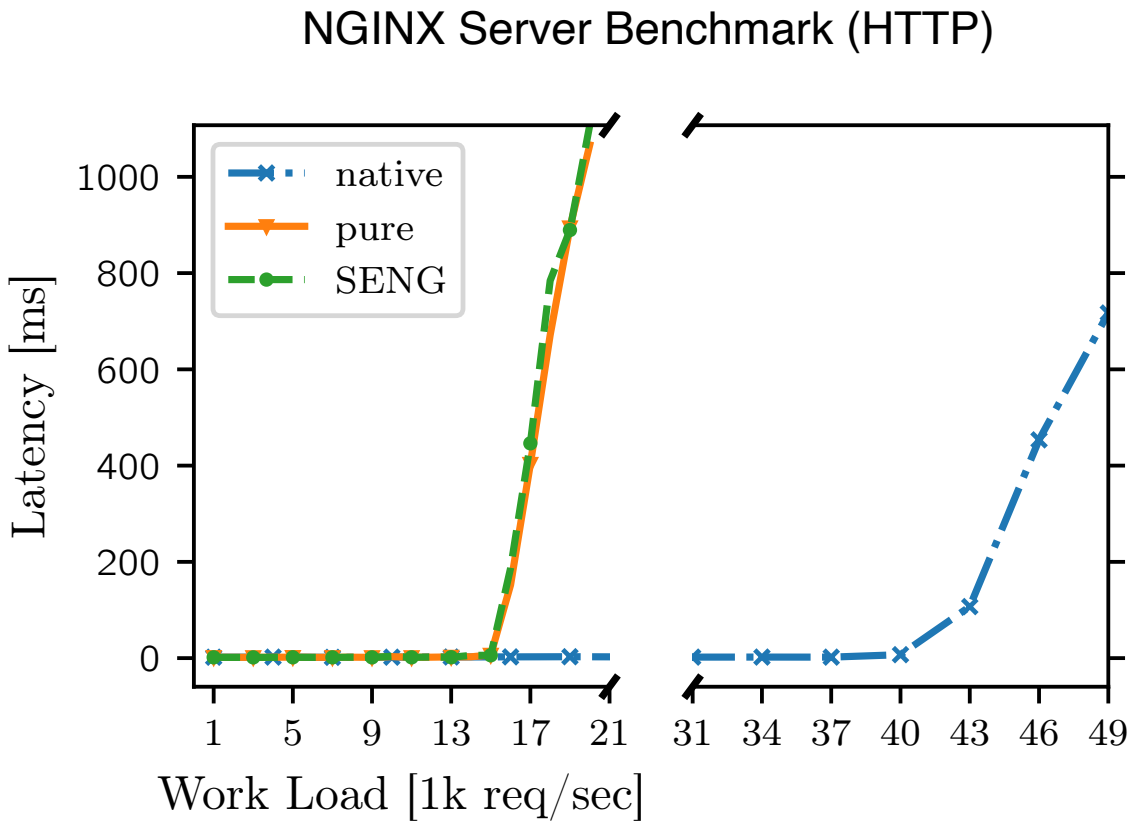
HTTP response latency (NGINX):

- 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?

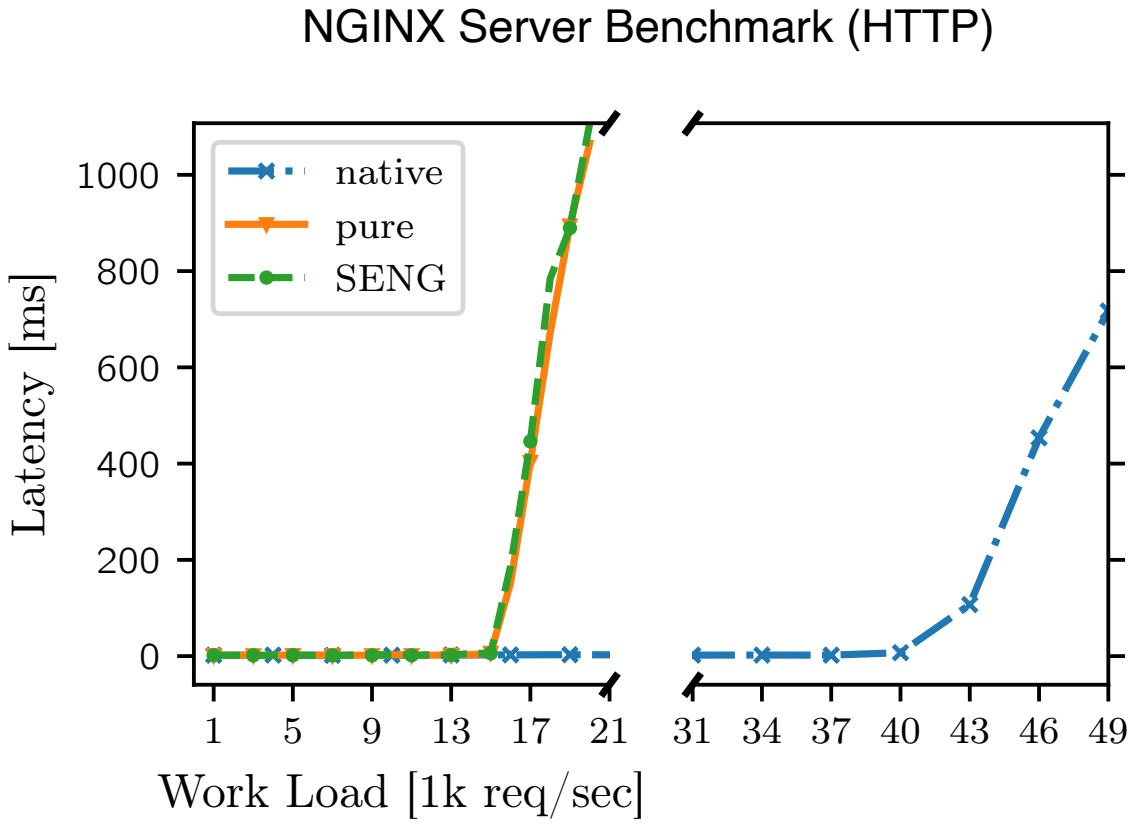


NGINX Server Application: SENG Runtime performance

Problem:

Our Graphene-SGX version only supports synchronous syscalls, no batch mode

==> Will faster primitives help?

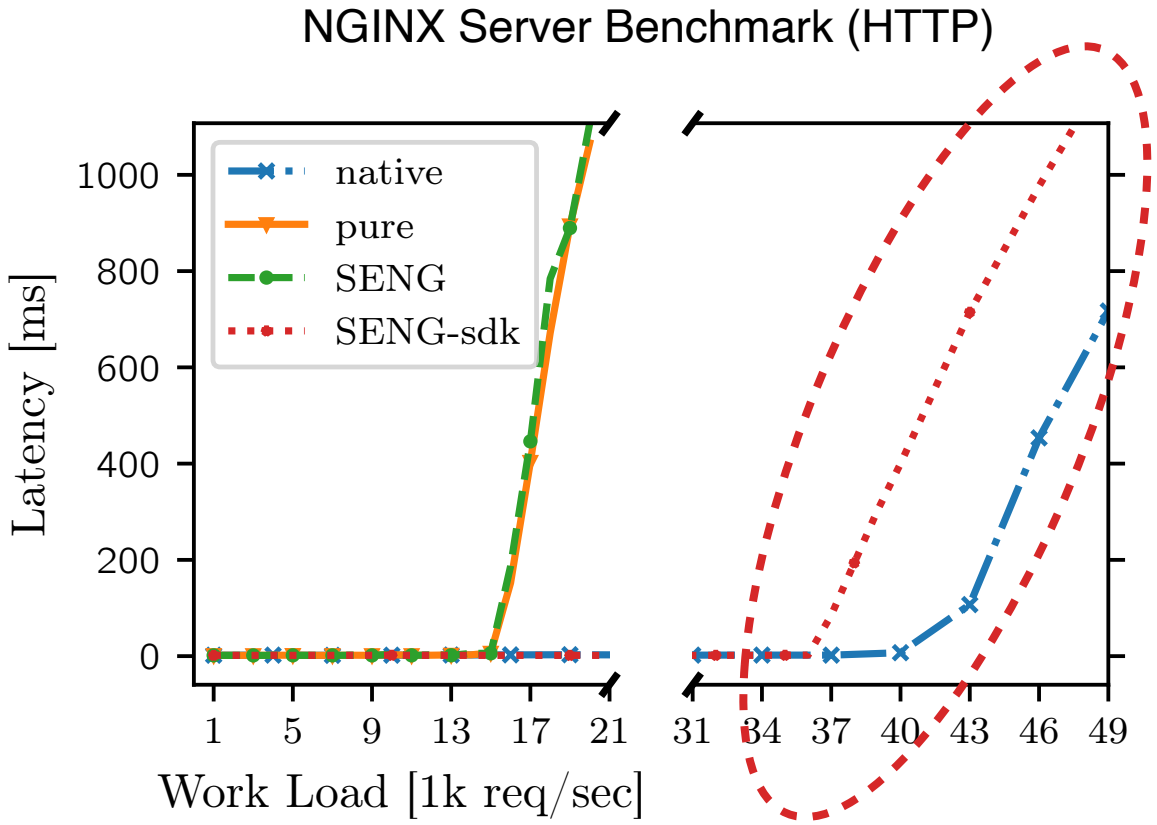


NGINX Server Application: SENG SDK (w/o LibOS)

Problem:

Our Graphene-SGX version only supports synchronous syscalls, no batch mode

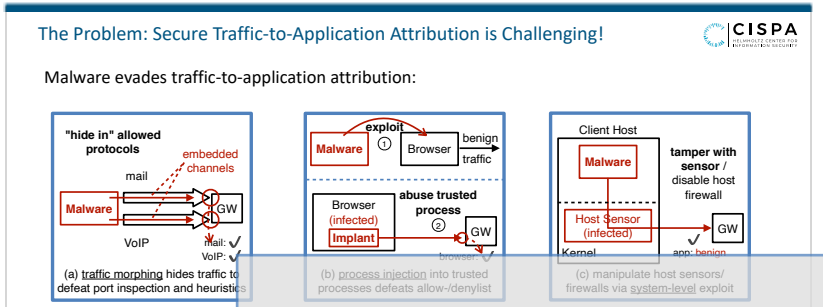
==> Will faster primitives help?

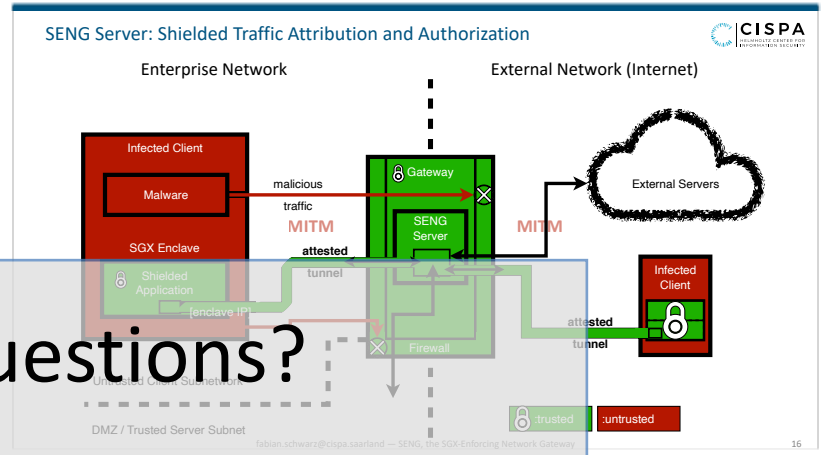


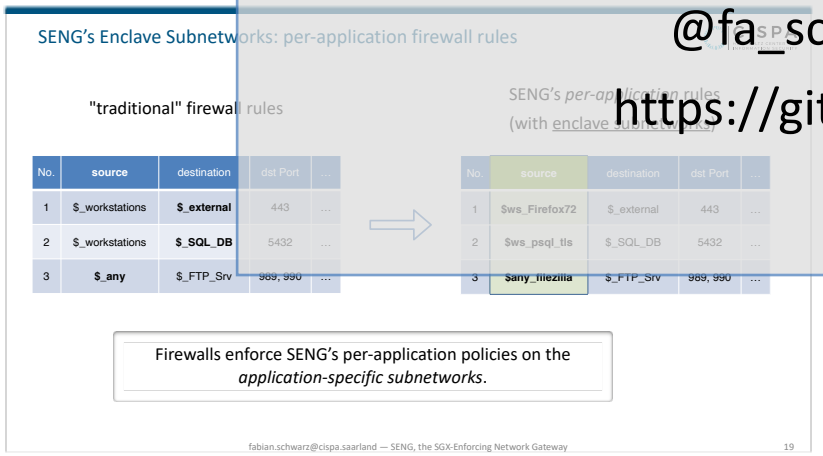
SENG SDK ("SENG-sdk"):

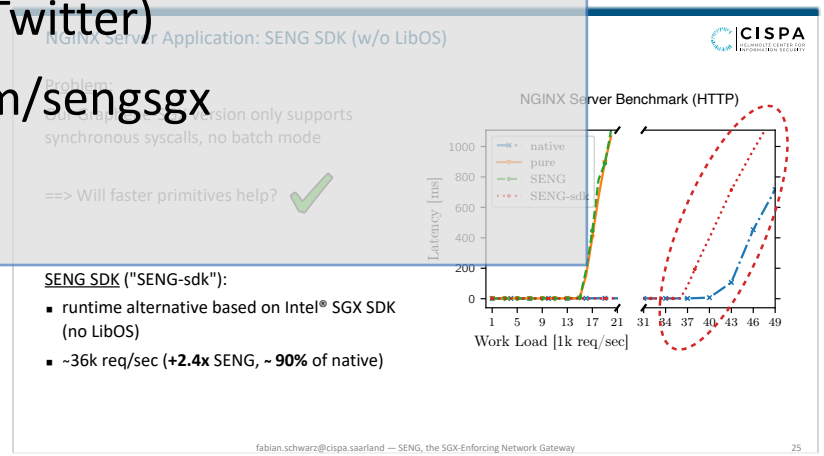
- runtime alternative based on Intel® SGX SDK (no LibOS)
- ~36k req/sec (**+2.4x** SENG, **~ 90%** of native)

Summary: SENG, the SGX-Enforcing Network Gateway









Thank you! Questions?

fabian.schwarz@cispa.saarland

@fa_schwarz (Twitter)

https://github.com/sengsgx