Clemmys
Towards Secure Remote Execution in FaaS

Bohdan Trach, Oleksii Oleksenko, Franz Gregor,
Pramod Bhatotia, Christof Fetzer
FaaS Paradigm of Cloud Computing
FaaS Paradigm of Cloud Computing

- Less boilerplate work 😊
- Easy autoscaling 😊
How does FaaS work?
How does FaaS work?
How does FaaS work?

Diagram:
- Gateway
- Controller
  - Worker 1: Function A
  - Worker 2: Function B, Function C
How does FaaS work?

- Gateway
- Controller
- Worker 1
  - Function A
  - Function B
- Worker 2
  - Function C
How does FaaS work?

Gateway → Secret → Controller → Function A → Function B → Function C
How does FaaS work?

Support for **function chaining** is an important requirement for serverless computing.
How does FaaS work?

Support for **function chaining** is an important requirement for serverless computing.
How does FaaS work?

Support for **function chaining** is an important requirement for serverless computing.
How does FaaS work?
How does FaaS work?
Is Faas secure?

- Less boilerplate work 😊
- Easy autoscaling 😊

![Diagram showing Gateway, Controller, Worker 1 (Function A), Worker 2 (Function C), and Controller connecting to Function B.]
Is Faas secure?

- Less boilerplate work 😊
- Easy autoscaling 😊
- Low-trust environment 😞
Why is FaaS insecure?

Inspection of network traffic:
- Gateway
- Controller
- Worker 1: Function A
- Worker 1: Function B
- Worker 2: Function C
Why is FaaS insecure?

- Gateway
- Controller
- Worker 1
  - Function A
  - Function B
- Worker 2
  - Function C

Inspect Network Traffic
Inspect Process Memory
State-of-the-Art: Computing on Untrusted Systems

- High performance overhead 😞
- Low flexibility 😞

Related Work:
- nGraph-HE [IACR 2019/350]
- PySyft
State-of-the-Art: Computing on Untrusted Systems

- Acceptable overhead 😊
- Arbitrary workloads 😊

Related Work:
- S-FaaS [CoRR abs/1810.06080]
What is Intel SGX?

User Application (Untrusted Memory)

Operating System
What is Intel SGX?

- Adds *enclave* abstraction
What is Intel SGX?

- Adds *enclave* abstraction
  - Encrypted in RAM only
What is Intel SGX?

- Adds *enclave* abstraction
  - Encrypted in RAM only
  - Not accessible from outside
What is Intel SGX?

- Adds *enclave* abstraction
  - Encrypted in RAM only
  - Not accessible from outside
  - Developer-specified entry points
What are the limitations of Intel SGX?

- High overheads for:
  - Secure memory paging
  - Enclave startup with large heap
Why do Intel SGX limitations matter?

Function startup time as an optimization target:

- SAND, SOCK [ATC’18]
Why do Intel SGX limitations matter?

Function startup time as an optimization target:

- SAND, SOCK [ATC’18]

Problem for SGXv1 enclaves
Why do Intel SGX limitations matter?

Function startup time as an optimization target:

- SAND, SOCK [ATC’18]

Problem for SGXv1 enclaves

- Can be solved with SGXv2

Additional optimizations are worth investigating.
Problem Statement

How to execute a **wide range** of user functions in FaaS in a **trustworthy** and **efficient** manner?
Outline

- Motivation
- Design
- Evaluation
- Summary
What is Clemmys?

Based on Apache OpenWhisk
What is Clemmys?

1. Trustworthy environment for function execution

Based on Apache OpenWhisk
What is Clemmys?

1. Trustworthy environment for function execution

2. Message format for secure function chaining

Based on Apache OpenWhisk

SGX Enclave  Native Application
What is Clemmys?

Based on Apache OpenWhisk

1. Trustworthy environment for function execution
2. Message format for secure function chaining
3. Function startup time optimizations (SGXv2)
What is Clemmys?

1. Trustworthy environment for function execution
2. Message format for secure function chaining
3. Function startup time optimizations (SGXv2)
4. Key management and deployment scheme

Based on Apache OpenWhisk

SGX Enclave

Native Application
What is Clemmys?

- **SGX Enclave**
- **Native Application**
- **Function A**
- **Function B**
- **Function C**
- **Controller**
- **Gateway**
- **Key Mgmt Service**

**Based on Apache OpenWhisk**

- **1. Trustworthy environment for function execution**
- **2. Message format for secure function chaining**
- **3. Function startup time optimizations (SGXv2)**
- **4. Key management and deployment scheme**

TLS ➔ **Gateway** ➔ Plaintext Metadata + Encrypted Data ➔ **Controller** ➔ Plaintext Metadata + Encrypted Data ➔ **Function A** ➔ **Function B** ➔ **Function C** ➔ **Key Mgmt Service**
What is Clemmys?

1. Trustworthy environment for function execution

2. Message format for secure function chaining

3. Function startup time optimizations (SGXv2)

4. Key management and deployment scheme

Based on Apache OpenWhisk

SGX Enclave  Native Application
Components of Clemmys

- **Internal encryption**
- Function chain verification
- Function startup optimizations
- Function deployment and key management
How does Clemmys secure communication?

EPC paging → slow!
How does Clemmys secure communication?
How does Clemmys secure communication?

Idea: separate controller metadata (plaintext) from function arguments (encrypted)
How does Clemmys secure communication?

Idea: separate controller metadata (plaintext) from function arguments (encrypted)
Components of Clemmys

- Internal encryption
- **Function chain verification**
- Function startup optimizations
- Function deployment and key management
Why should function chain order be enforced?

- Naive encryption does not preserve function order.
Why should function chain order be enforced?

- Naive encryption does not preserve function order.
Why should function chain order be enforced?

- Naive encryption does not preserve function order.
- Message format should preclude these attack vector.
Why should function chain order be enforced?

- Naive encryption does not preserve function order.
- Message format should preclude these attack vectors.

See paper for technical details
Components of Clemmys

- Internal encryption
- Function chain verification
- **Function startup optimizations**
- Function deployment and key management
Startup Optimizations

1. SGXv1 Enclave Creation
Startup Optimizations

1. SGXv1 Enclave Creation

![Enclave VM Range](image)
Startup Optimizations

1. SGXv1 Enclave Creation

![Diagram of Enclave VM Range with EPC Size and Paged out!]
Startup Optimizations

1. SGXv1 Enclave Creation

Enclave VM Range

EPC Size

Paged out!
Startup Optimizations

1. SGXv2 Enclave Creation

SGXv2 allows adding pages at runtime
1. SGXv2 Enclave Creation

SGXv2 allows adding pages at runtime
Startup Optimizations

1. SGXv2 Enclave Creation

SGXv2 allows adding pages at runtime
Startup Optimizations

1. SGXv2 Enclave Creation
2. EPC Batch Augmentation
Startup Optimizations

1. SGXv2 Enclave Creation
2. EPC Batch Augmentation

Enclave VM Range

- Block of N pages augmented at once
- Memory access inside enclave
Startup Optimizations

1. SGXv2 Enclave Creation
2. EPC Batch Augmentation
3. Memory zeroing on deallocation

Enclave VM Range

Freshly allocated region of heap memory

Need to be explicitly zeroed with SGXv1
Startup Optimizations

1. SGXv2 Enclave Creation
2. EPC Batch Augmentation
3. Memory zeroing on deallocation

Enclave VM Range

Freshly allocated region of heap memory

Guaranteed to be zero-filled with SGXv2
Startup Optimizations

1. SGXv2 Enclave Creation
2. EPC Batch Augmentation
3. Memory zeroing on deallocation
Components of Clemmys

● Internal encryption
● Function chain verification
● Function startup optimizations
● Function deployment and key management
How is Clemmys function deployed?

Client

Gateway

Palaemon

Controller

Function A

Function B

Function C

SGX Enclave

Native Application
How is Clemmys function deployed?

- Palaemon - remote attestation and configuration service
- Transparent configuration management:
  - Environment variables and command-line arguments
How is Clemmy's function deployed?

Client → Remote Attestation (Intel) → Palaemon → Function A, Function B, Function C

Gateway → Controller

SGX Enclave  Native Application  Trust Established
How is Clemmy's function deployed?

- Client
  - Upload configuration (chains, secrets)
  - Palaemon

- Gateway
- Controller

- Function A
- Function B
- Function C

- SGX Enclave
- Native Application
- Trust Established
How is Clemmys function deployed?

- **Client**
  - Upload functions (Docker images)
  - Gateway

- **Palaemon**
  - Controller

- **Function A**
- **Function B**
- **Function C**

- **SGX Enclave**
- **Native Application**
- **Trust Established**
How is Clemmys function invoked?

- Client
- Gateway
- Palaemon
- Controller
- Function A
- Function B
- Function C
How is Clemmys function invoked?

Remote attestation via Palaemon at launch

Client

Gateway

Palaemon

Controller

Function A

Function B

Function C

SGX Enclave  Native Application  Trust Established
How is Clemmys function invoked?
How is Clemmys function invoked?

Client → Gateway → Controller → Palaemon → Function A, B, C

- SGX Enclave
- Native Application
- Trust Established
How is Clemmy's function invoked?

Client -> Gateway -> Palaemon -> Controller

- TLS API Request
- Function A
- Function B
- Function C

- SGX Enclave
- Native Application
- Trust Established
How is Clemmys function invoked?

Client \(\rightarrow\) TLS API Request \(\rightarrow\) Gateway

\(\rightarrow\) Plaintext Metadata + Encrypted Data \(\rightarrow\) Controller

\(\rightarrow\) Trust Established

Function A
Function B
Function C
How is Clemmym's function invoked?

Client → Gateway

- TLS API Request
- Plaintext Metadata + Encrypted Data

Palaemon

Controller → Worker Platform

- Plaintext Metadata + Encrypted Data

Function A

Trust Established
How is Clemmys function invoked?

1. Platform launches the enclave using the plaintext metadata
How is Clemmym's function invoked?

1. Platform launches the enclave using the plaintext metadata
2. Enclave performs remote attestation and configuration with Palaemon
How is Clemmyns function invoked?

1. Platform launches the enclave using the plaintext metadata
2. Enclave performs remote attestation and configuration with Palaemon
3. Enclave decrypts and processes the request

Diagram:
- **Client**
  - TLS API Request
- **Gateway**
  - Plaintext Metadata + Encrypted Data
- **Controller**
  - Plaintext Metadata + Encrypted Data
- **Function A**
- **Worker Platform**
- **Palaemon**

Legend:
- SGX Enclave
- Native Application
- Trust Established
Outline

- Motivation
- Design
- Evaluation
- Summary
Gateway Overhead

Gateway

Controller

Function A
Function B
Function C

VS.

Gateway

Controller

Function A
Function B
Function C

SGX Enclave
Native Application
Gateway Overhead

The graph compares the response time (in seconds) for various operations using Native and SGX Gateway methods. The x-axis represents different operations: matrix multiply, string match, word count, dedup, x264, and fluidanimate. The y-axis shows the response time, with lower values indicating better performance. The trend indicates that SGX Gateway generally has a lower response time across all operations compared to Native, suggesting it is more efficient.
Gateway Overhead

Number of functions running on the worker node
Gateway Overhead

lower ➝ better
Gateway Overhead

lower ➝ better
Minimal overhead (~1-5%) over native API Gateway
Function Overhead

Gateways

Controller

Function

VS.

Gateways

Controller

Function

SGX Enclave

Native Application
Function Overhead

lower ➝ better
Function Overhead

lower → better

Response time (s)
Function Overhead

lower ➝ better
Function Overhead

Minimal overhead over native functions (up to 25%)
SGXv2 Optimizations

VS.

SGXv2 Optimizations

SGX Enclave  Native Application
SGXv2 Optimizations

Speedup normalized by the SGXv1 function run time
SGXv2 Optimizations

higher ➝ better

Speedup (w.r.t. SGXv1)

- vips
- fluidanimate
- bodytrack
- ferret
- dedup
- facesim
- streamcluster
- blackscholes
- x264
- canneal
- swaptions
- mean
SGXv2 Optimizations

- **SGXv2** - all optimizations
- **SGXv2(NB)** - no batched augmentation
- **SGXv2(NB, NO)** - no batched augmentation and memory zeroing on deallocation
SGXv2 Optimizations

- SGXv2 - all optimizations
- SGXv2(NB) - no batched augmentation
- SGXv2(NB, NO) - no batched augmentation and memory zeroing on deallocation
SGXv2 Optimizations

- SGXv2 - all optimizations
- SGXv2(NB) - no batched augmentation
- SGXv2(NB, NO) - no batched augmentation and memory zeroing on deallocation
SGXv2 Optimizations

10 times lower latency on Phoenix benchmarks with SGXv2
10% lower latency from additional optimizations on a few benchmarks
Summary

Clemmys is:

- **Secure** - protects functions using enclave
- **Fast** - achieves near-native performance
- **Flexible** - does not restrict workloads
Summary

Clemmys is:

- **Secure** - protects functions using enclave
- **Fast** - achieves near-native performance
- **Flexible** - does not restrict workloads

Thank You for your attention!

bohdan.trach@tu-dresden.de
Funding

This project was funded by the European Union’s Horizon 2020 program under grant agreement No. 690588 (Selis), and BMBF No. 03ZZ0517A (FastCloud)