Stony Brook University

LeakLess: Selective Data Protection against Memory Disclosure and Transient Execution Attacks for Serverless Environments

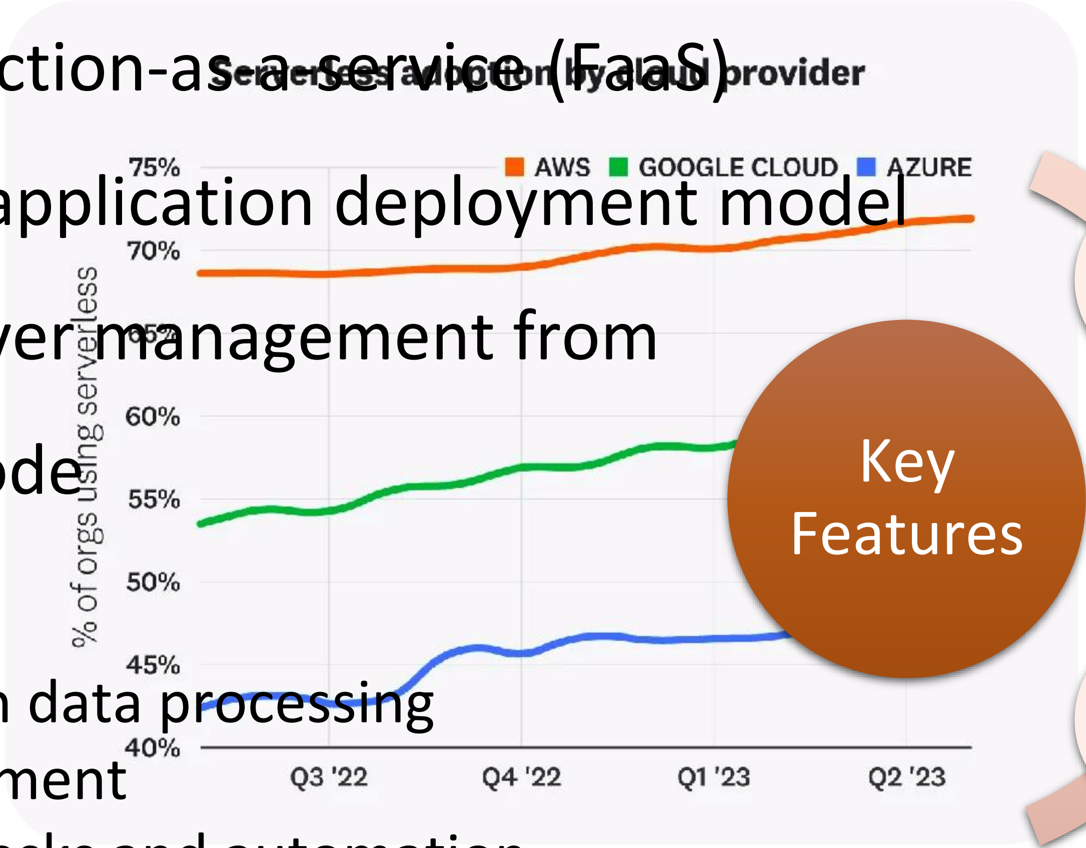
Maryam Rostamipoor, Seyedhamed Ghavamnia, Michalis Polychronakis

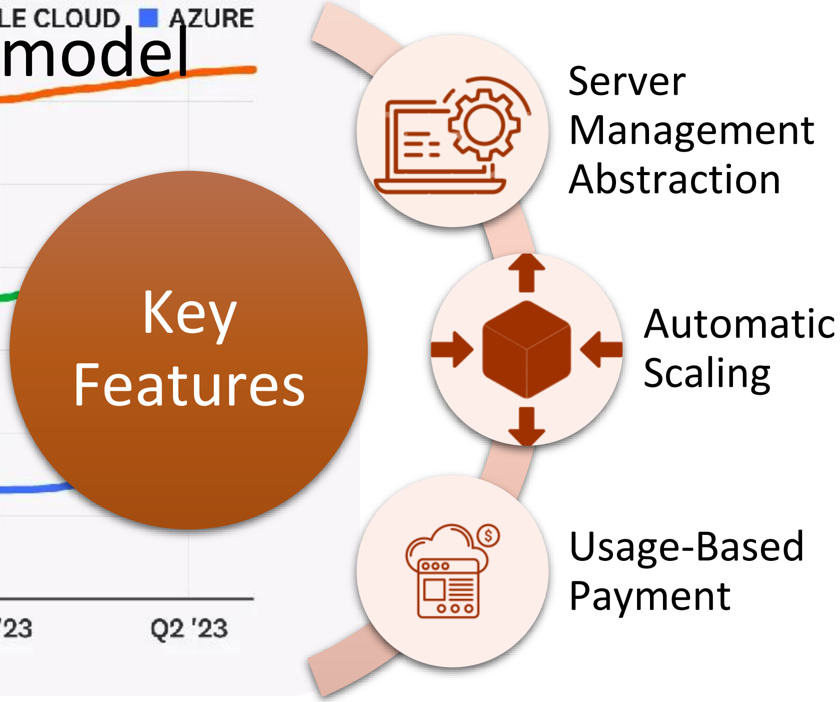
hexlab

FAR BEYOND

What is Serverless Cloud Computing?

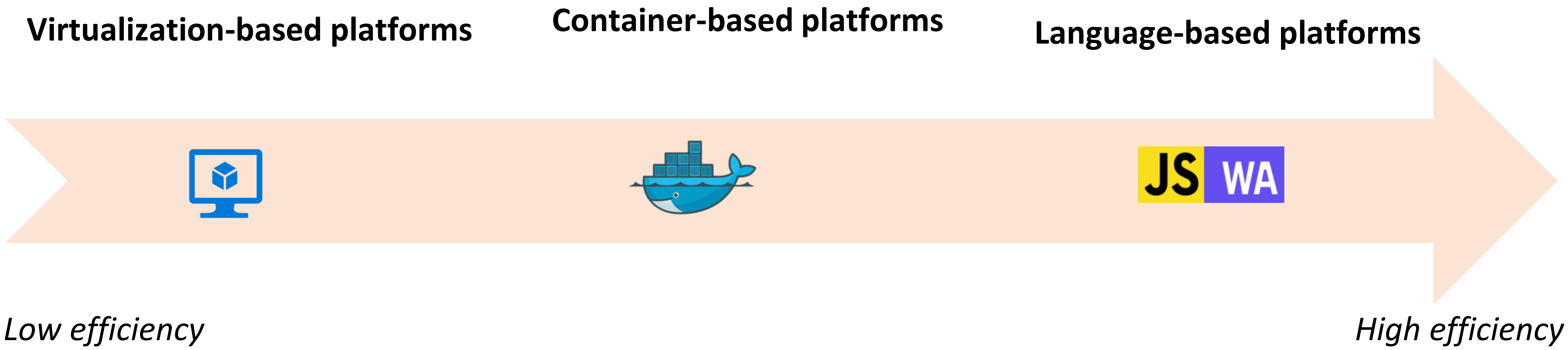
- Known as function-as-a-service (FaaS)
- Cloud-based application deployment model
- Abstracts server management from application code
- Applications:
  - Event-driven data processing
  - API management
  - Scheduled tasks and automation





Security risk: functions are executed on shared servers used by multiple teams or organizations.

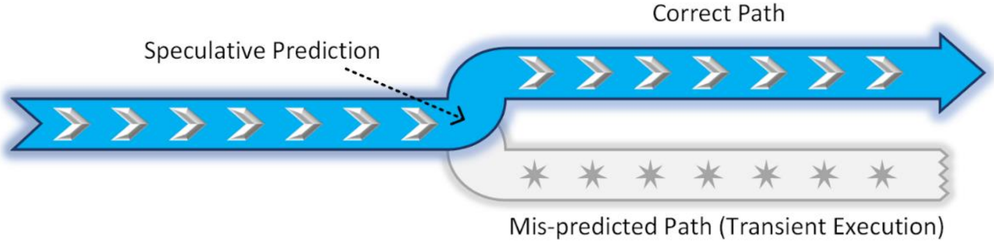
Function Isolation



Language-level sandboxing strongly guards against memory disclosure vulnerabilities but makes it vulnerable to **transient execution** attacks.

Transient Execution Attacks

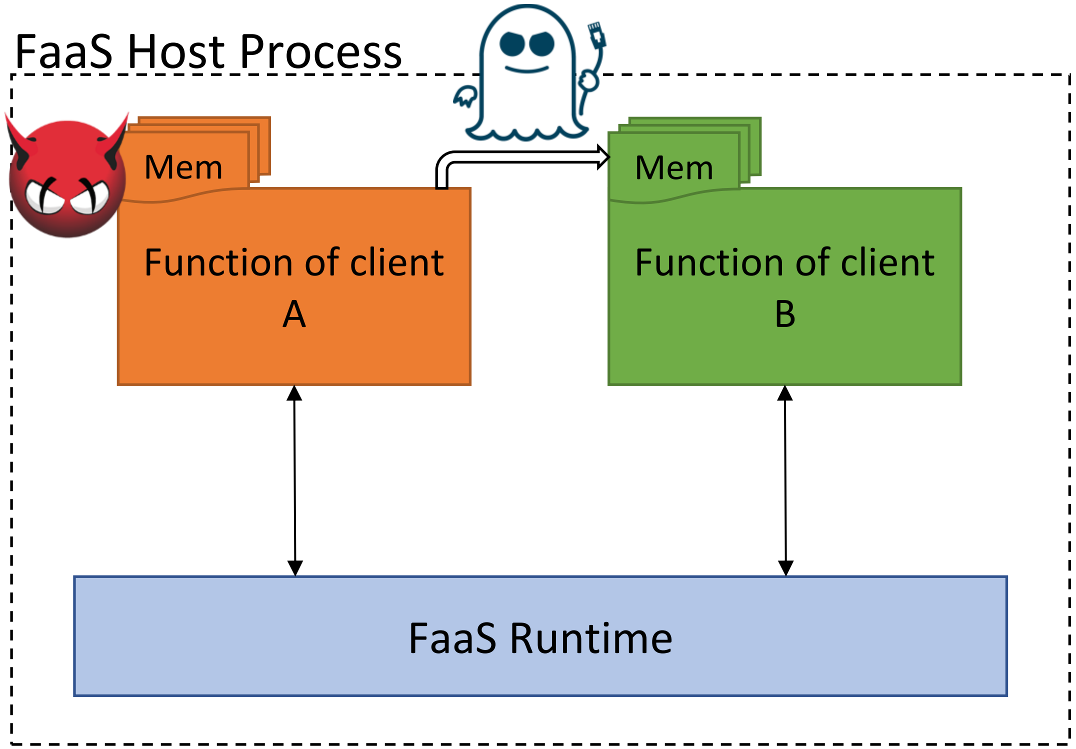
- Speculative execution
  - CPU tries to guess what code needs to be executed next
  - The results of the speculative execution are discarded
  - The cache is affected by these reads



- Spectre [Kocher et al. 2019]
  - Exploits speculative execution
  - Uses cache timing attack to leak sensitive information

Stony Brook University

Using Spectre to Break Function Isolation




```
if (addr in heap) {
  x = read(*addr);
  y = read(x);
}
```

Stony Brook University

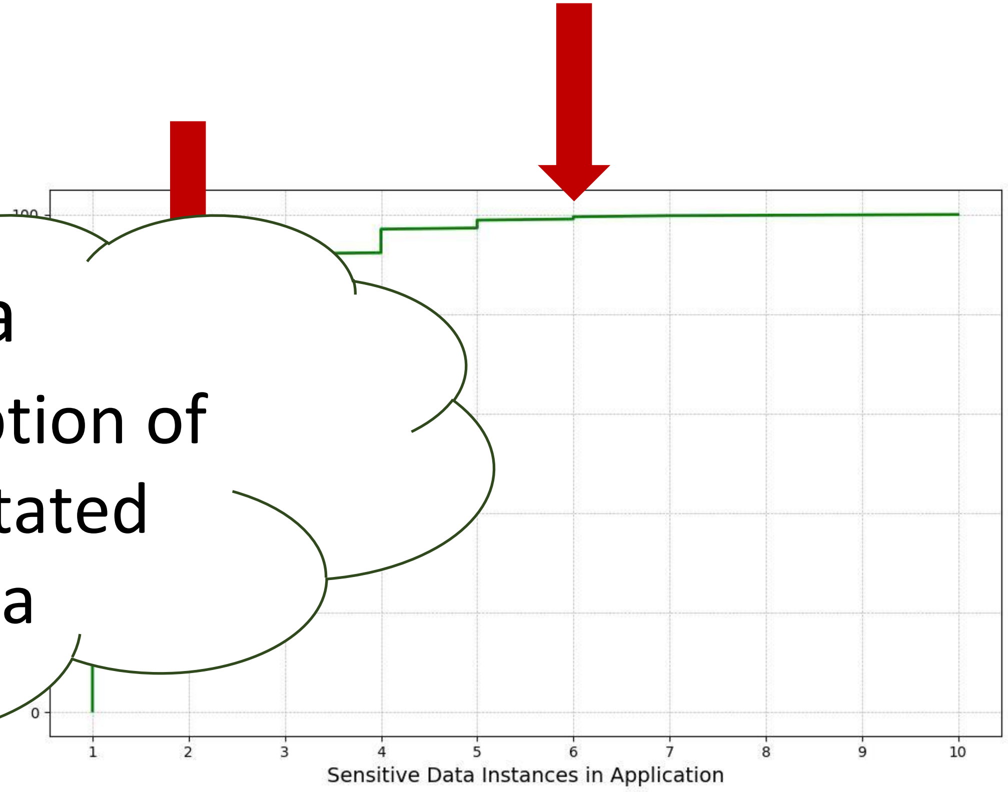
Transient Execution Protection Approaches in Serverless Platforms

- Preventing sandbox breakouts by hardening functions [Narayan et al., 2021]
- Identifying and isolationg suspicious workloads [Schwarzl et al., 2022]

The focus is predominantly on identifying and preventing variant branches of Spectre attacks.

Need for Selective Data Protection

- Only **42%** of the applications handle sensitive data
- About **80%** of thes just one or two
- Most applicatio sensitive data objects



Our idea

in-memory encryption of developer-annotated sensitive data

Implication: Selective data protection is justified as opposed to securing all data objects of a given application.

Need for I/O Brokering

Internal sensitive data

- Originates from within a function
- Is used internally by the function
- Is transmitted to external entities
- External services' API tokens

A s

managing communications

and

External sensitive data

- Originates from external entities
- Is received by a function
- Can be transmitted to external entities
- Credit card numbers, user passwods

Implication 2: Prior works on selective data encryption are not applicable in this setting, as they are designed to protect only internal application data that is not transmitted to outside.

Sensitive Data Annotation

Source Code Annotation

```
#[LEAKLESS_SECRET]
const AUTH_TOKEN: &str = "secret-
token";

let mut res =
 spin_sdk::outbo
 http::Request::
 .method("GET")
 .header("Authorization",
 HeaderValue::from_bytes(AUTH_TOKEN))

.uri("https://backend_domain.com/image.
jpg")
 .body(None)?,
)?;
```

Sensitive Data Marshaling

```
GET /image.jpg HTTP/1.1
Host: backend_domain.com
Authorization: LEAKLESS_A
```

Language-agnostic Annotation

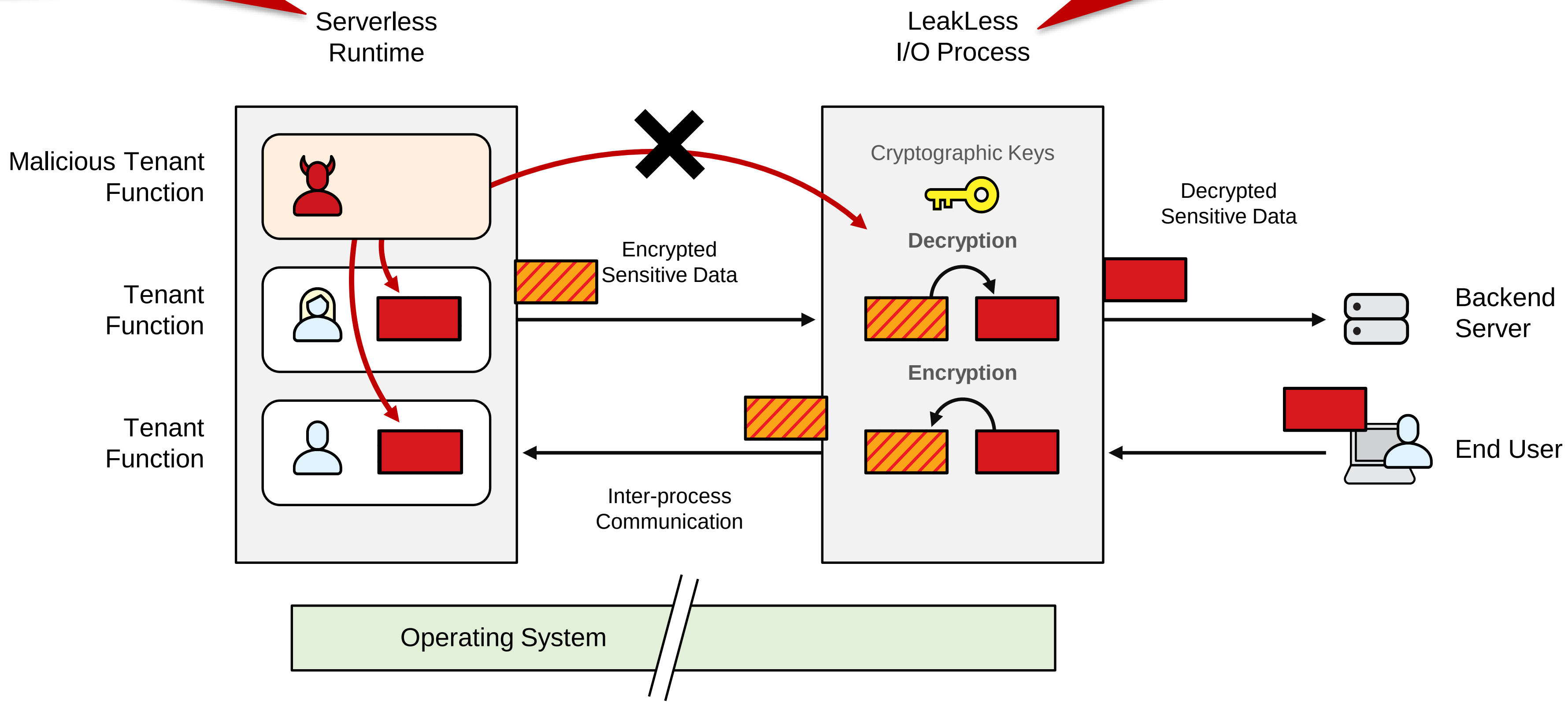
```
tion of Secrets in
Configuration file
Authentication_Token = {default =
"secret-token", secret
= true, leakless_secret = true}

// Rust code
let authToken =
config::get("Authentication_Token").exp
ect("could not get variable");
```

Stony Brook University

Design

An encryption key for each Tenant application

Protecting against cross-process transient execution attacks



LeakLess protects sensitive data in Serverless platforms by always keeping it encrypted in memory and handles cryptographic operations using a separate I/O module.

Stony Brook University

Supported Types of Sensitive Data and Operations

- **Immu**
  - Sensit
  - Comp
  - Identi
- Comm
  - Com
  - 83%

```
//Annotation for signing requests
S3_sign_key = { default = "7IhhnziifKKdcf0",
leakless_secret = true, leakless_operation =
"request-sign" }

//Annotation for verifying JWT tokens
JWT_secret = { default = "secret_key", leakless_secret =
true, leakless_operation = "verify-jwt" }
```

Compatibility Assessment

Categorization of the different types of sensitive data objects found in serverless applications.

Data Set	LeakLess Supported Sensitive Data types						Unsupported Sensitive Data Types		
	Database Password	Database Name	Authentication Secret	Password	JWT Signing or Verification Key	Request Signing Key	Other Crypto Key	Modified Auth. Secret	Modified Password
Wonderless Dataset	41	36	367	27	23	185	18	1	12
Serverless Framework	5	5	60	0	0	37	1	0	0
Fastly	0	0	7	0	0	4	4	0	1
Cloudflare	1	1	88	2	3	3	12	3	0
Spin	5	5	5	2	0	0	1	1	0
Total	52	47	527	31	26	229	36	5	13

LeakLess supports 94% (912 out of 966) of the identified sensitive data objects.

Performance Evaluation

Throughput reduction and latency increase for six real-world serverless applications

Application	Latency (ms)					Throughput (req/s)				
	Orig.	I/O Only		I/O + Encryption		Orig.	I/O Only		I/O + Encryption	
		Remote	Local	Remote	Local		Remote	Local	Remote	Local
Authentication Using Stored Tokens	1,267	1,310 (3.3)	1,356 (7.0)	1,310 (3.3)	1,357 (7.1)	769	747 (2.8)	723 (5.9)	747 (2.8)	723 (5.9)
Authenticating Users at the Edge	1,285	1,290 (0.4)	1,310 (1.9)	1,290 (0.4)	1,320 (2.7)	766	764 (0.0)	749 (2.2)	765 (0.0)	739 (3.5)
Using Stored Passwords	1,267	1,310 (3.4)	1,356 (7.0)	1,310 (3.4)	1,356 (7.0)	769	749 (2.6)	722 (6.1)	751 (2.3)	722 (6.1)
Signing JWT Keys	1,240	1,280 (3.2)	1,350 (8.8)	1,280 (3.2)	1,360 (9.6)	788	766 (2.7)	722 (8.3)	766 (2.7)	721 (8.5)
Signing Requests	1,466	1,470 (0.0)	1,560 (6.4)	1,470 (0.0)	1,574 (7.3)	666	660 (0.0)	632 (5.1)	660 (0.0)	620 (6.9)
Transmitting User-Provided Secrets	1,340	1,360 (1.4)	1,460 (6.9)	1,365 (1.8)	1,470 (9.7)	730	724 (0.8)	671 (8.0)	723 (0.9)	670 (8.2)

Our results demonstrate that LeakLess offers robust protection while incurring a slight throughput decrease of up to **2.8%**.

Stony Brook University

Security Evaluation

- Serverless environments that utilize sandboxed language to isolate functions are vulnerable to Spectre attack [Schwarzl et al., 2022]
- Recreated the attack scenario:
  - A client sends 1000 concurrent requests to the victim
  - A custom program repeatedly dumps the memory of the main process using gcore

Stony Brook University

Summary

- Development of **LeakLess** for selective data protection on serverless platforms
  - Future-proof against memory leaks and execution attacks
- Integration of in-memory encryption with a dedicated I/O module
  - Overcomes previous data protection constraints
- Analyzed a set of **1,074** real-world serverless applications
- 91% compatibility of **LeakLess** with apps that handle sensitive information
- Slight performance impact with LeakLess
  - Throughput reduced by a maximum of **2.8%**

Stony Brook University

Thank You!

FAR BEYOND