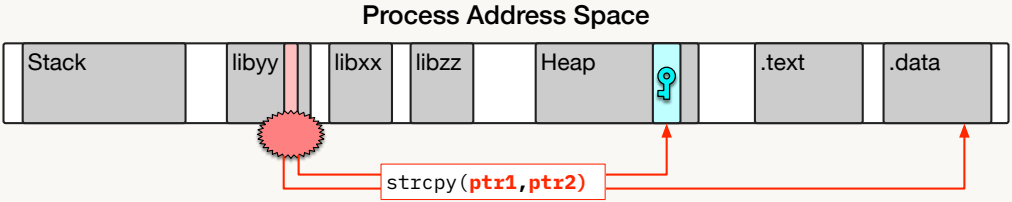
CAPACITY: Cryptographically-Enforced In-Process Capabilities for Modern ARM Architectures

Kha Dinh Duy, Kyuwon Cho, Taehyun Noh, Hojoon Lee

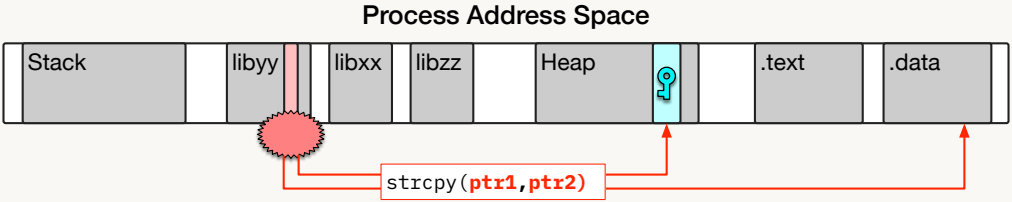
Systems Security Lab
Dept. of Computer Science and Engineering
Sungkyunkwan University

December 12, 2023

Perils of Monolithic Programs Designs



Perils of Monolithic Programs Designs



- A single vulnerability *anywhere* may be enough for adversary to compromise program

Process-based Isolation

Real-world examples: modern web browsers (Chrome, Safari, etc..) and OpenSSH

Process-based Isolation

Real-world examples: modern web browsers (Chrome, Safari, etc..) and OpenSSH

Limitations

- Must be incorporated from design
- High engineering costs
- High performance overhead due to IPC

In-Process Isolation

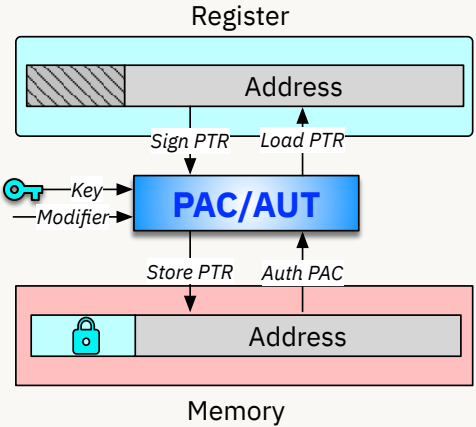
- **PKU-based Isolation in x86.** Isolates memory pages into multiple domains in page-granularity
  - ERIM (SEC ’19), Hodor (ATC ’19),
- **Reference monitor.** Supplements PKU memory isolation through syscall filtering
  - Jenny (SEC ’22), Cerberus (EuroSys ’22)

CAPACITY Research Statement

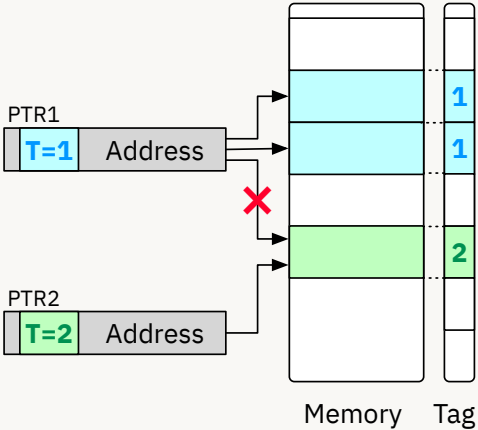
CAPACITY explores *capability*-based in-process isolation design inspired by the direction of new hardware security in modern ARM architecture.

- Fully exploits new ARM features that inherently exhibit characteristics of *capability*

New Hardware Security Features on ARM



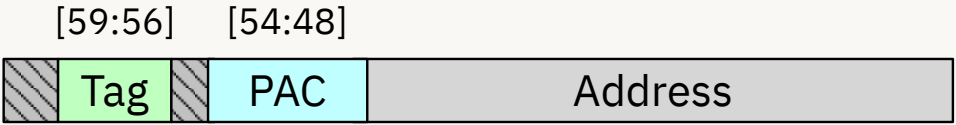
Pointer Authentication (PA)



Memory Tagging Extension (MTE)

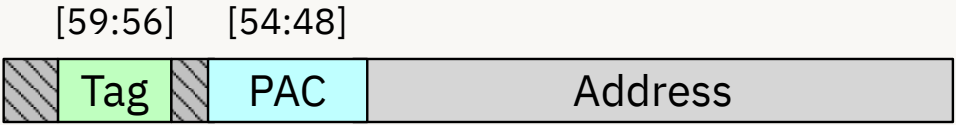
PA+MTE

- PA and MTE can be **simultaneously enabled** to make pointer *authenticated* and *tagged*



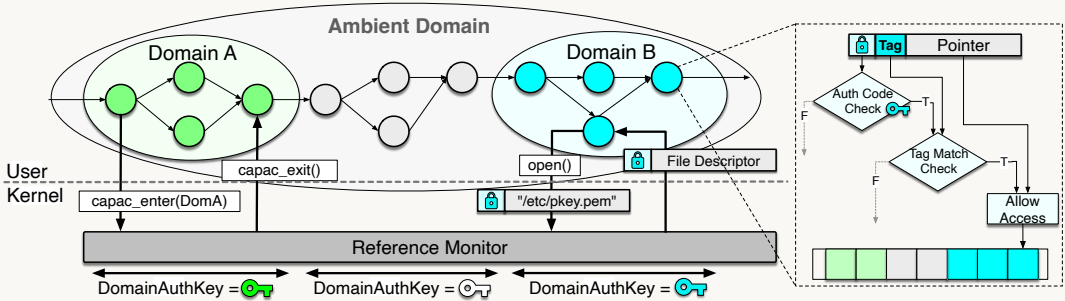
PA+MTE

- PA and MTE can be **simultaneously enabled** to make pointer *authenticated* and *tagged*

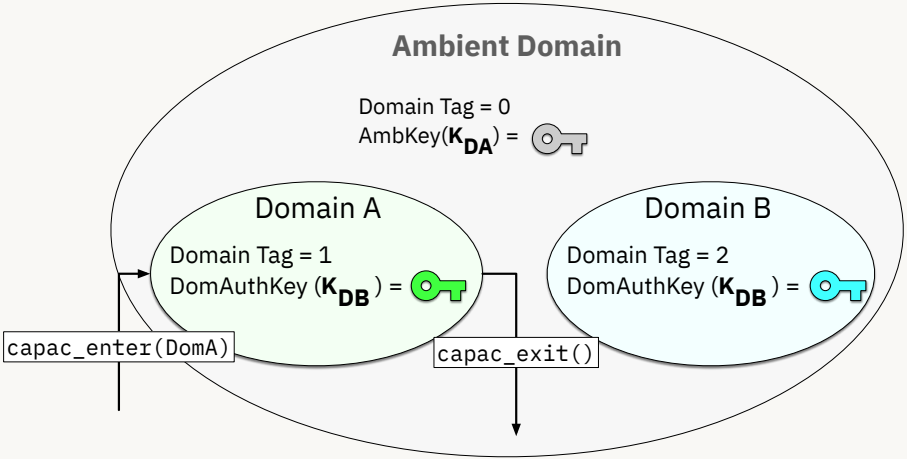


PA+MTE inherently exhibit characteristics of **capability-based access control**

CAPACITY Overview

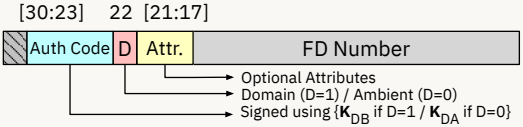


CAPACITY Domains

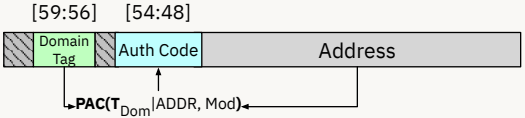


Non-forgeable References

- **Domain-private references**: Signed/Authenticated with DomAuthKey ($\mathbf{K}_{DB}$), exclusive to owner domain
- **Ambient references**: Signed/Authenticated with Ambient Key ($\mathbf{K}_{DA}$), valid in all domain (e.g., `stdin`)

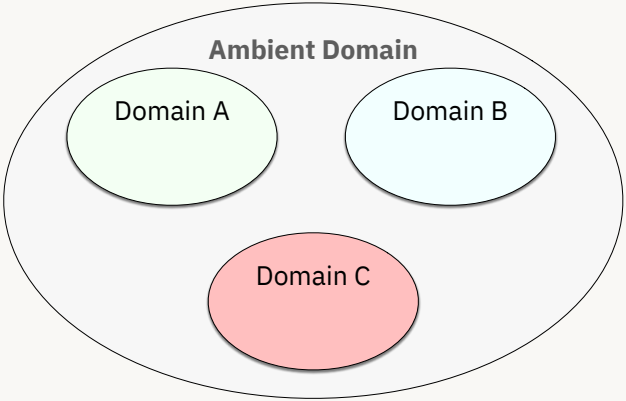


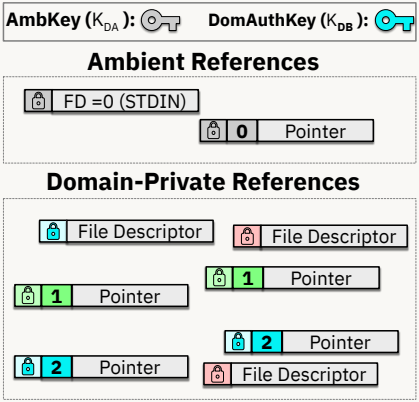
CAPACITY signed file descriptor



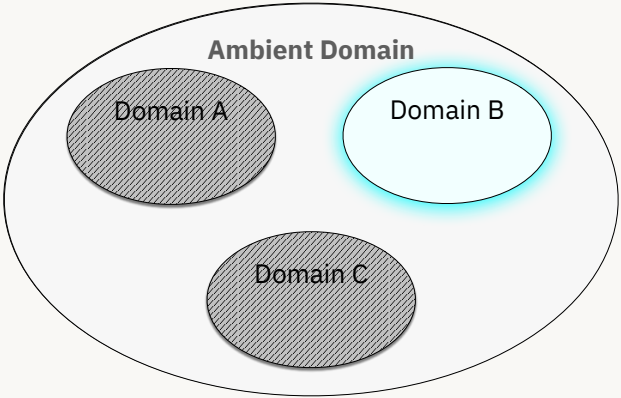
CAPACITY tagged & signed pointer

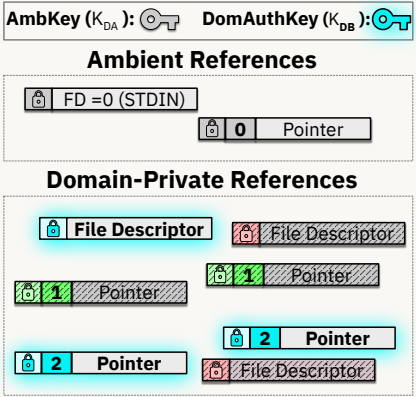
Domains Switching and Reference Validity



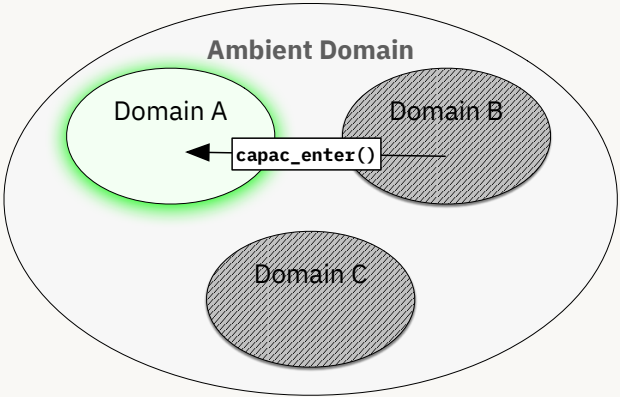


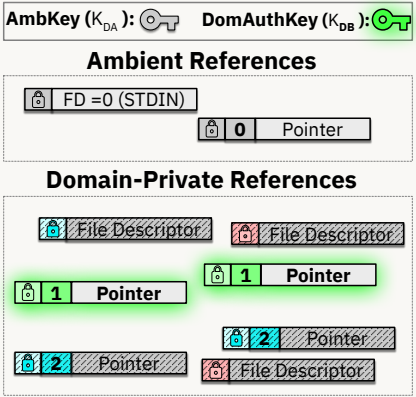
Domains Switching and Reference Validity



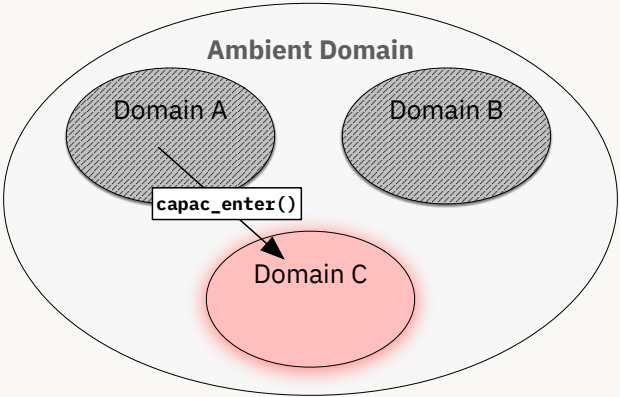


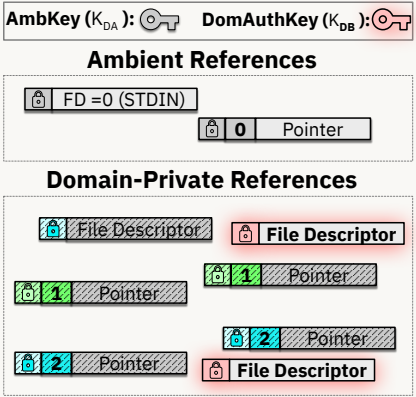
Domains Switching and Reference Validty



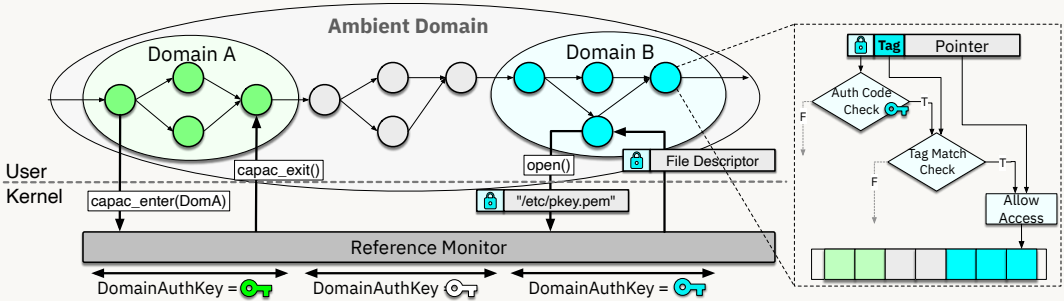


Domains Switching and Reference Validty

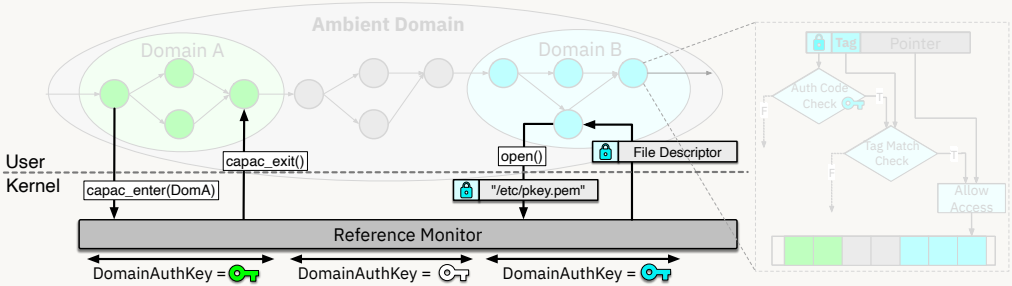




CAPACITY Design and Implementation

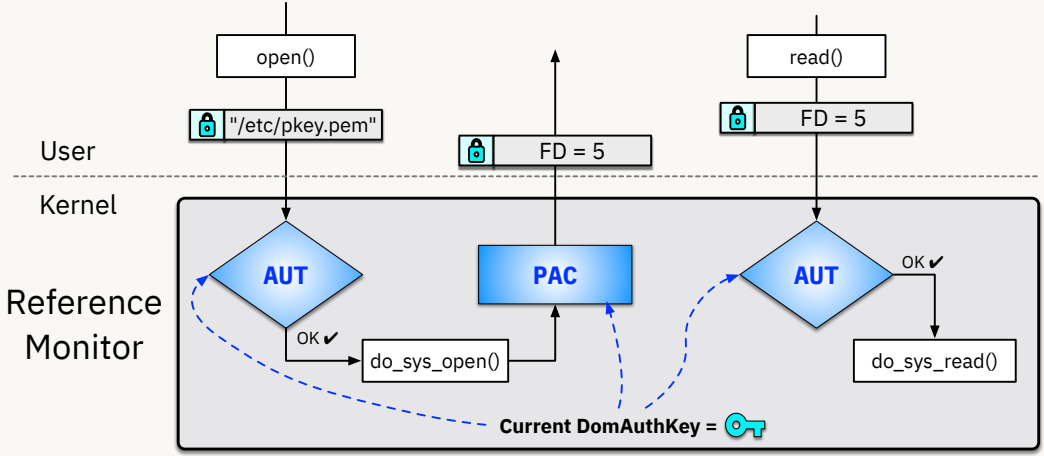


CAPACITY Design and Implementation

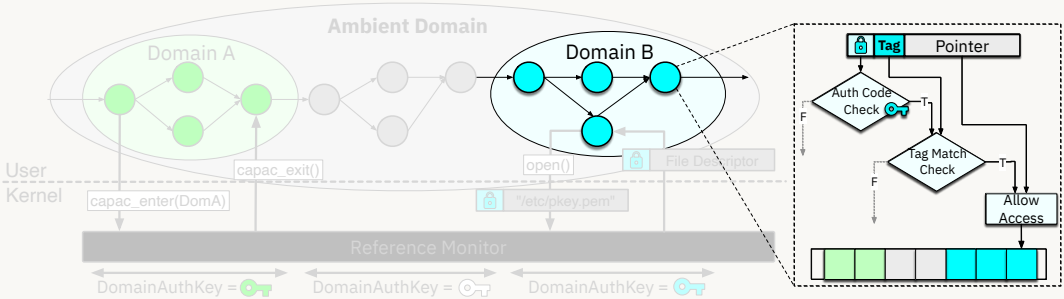


CAPACITY reference monitor

CAPACITY Reference monitor and File References

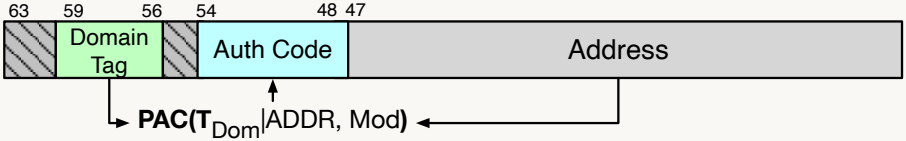


Instrumentation Framework and Domain Memory Isolation



CAPACITY instrumentation framework and Runtime library

Instrumentation Framework and Domain Memory Isolation



- **Tagged domain-private stack:** Instrumentation of domain functions to provide tagged stacks
- **Tagged domain-private heap:** Extended heap allocator to tag memory and provide tagged pointers

Domain-aware Pointer Load/Store Instrumentation

Annotated program

```
1  void sensitive_func(
2      DOM_PRIV void* sensitive_ptr,
3               void* ambient_ptr
4  )
5      ...
```

```
1  // Domain-Private PTR
2  pacdb   ptr, mod          PTR-Sign
3  str     ptr, [mem]
4  ...
5  ldr     ptr, [mem]
6  autdb   ptr, mod          PTR-Auth
```

```
1  // Ambient PTR
2  pacda   ptr, mod          PTR-Sign
3  str     ptr, [mem]
4  ...
5  ldr     ptr, [mem]
6  autda   ptr, mod          PTR-Auth
```

Programming Model

```
1     ...
2   void load_secret(DOM_PRIV crypto_ctx_t* ctx,
3                    const char * key_path){
4
5
6
7
8
9   ...
10 }
```

: Domain-private references : Ambient references

Action : CAPACITY actions ANNOT: CAPACITY annotations

Programming Model

```
1     ...
2   void load_secret(DOM_PRIV crypto_ctx_t* ctx,
3                    const char * key_path){
4     // Import the secret key from the filesystem
5     int fd                                                  FD-Sign
6         = open(key_path, O_RDONLY);                         PATH-Auth
7
8
9   ...
10 }
```

: Domain-private references : Ambient references

Action : CAPACITY actions ANNOT: CAPACITY annotations

Programming Model

```
1     ...
2   void load_secret(DOM_PRIV crypto_ctx_t* ctx,
3                    const char * key_path){
4     // Import the secret key from the filesystem
5     int fd                                                    FD-Sign
6         = open(key_path, O_RDONLY);                           PATH-Auth
7     ctx->secret_key = capac_malloc(KEY_LEN + 1);              PTR-Sign
8
9   ...
10 }
```

: Domain-private references : Ambient references

Action : CAPACITY actions ANNOT: CAPACITY annotations

Programming Model

```
1     ...
2   void load_secret(DOM_PRIV crypto_ctx_t* ctx,
3                    const char * key_path){
4     // Import the secret key from the filesystem
5     int fd                                                    FD-Sign
6         = open(key_path, O_RDONLY);                           PATH-Auth
7     ctx->secret_key = capac_malloc(KEY_LEN + 1);              PTR-Sign
8     read(fd, ctx->secret_key, KEY_LEN);                       FD-Auth / PTR-Auth
9   ...
10 }
```

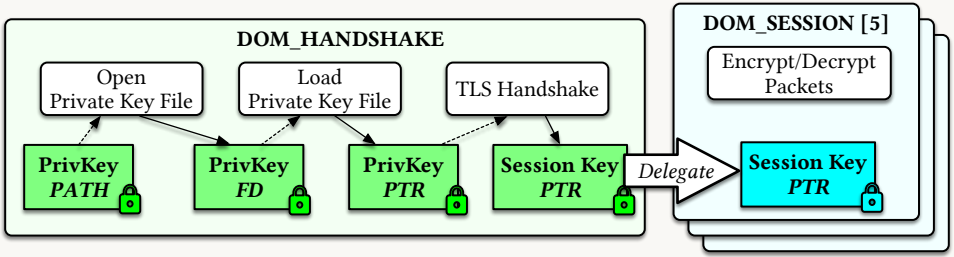
: Domain-private references : Ambient references

Action : CAPACITY actions ANNOT: CAPACITY annotations

Evaluation method

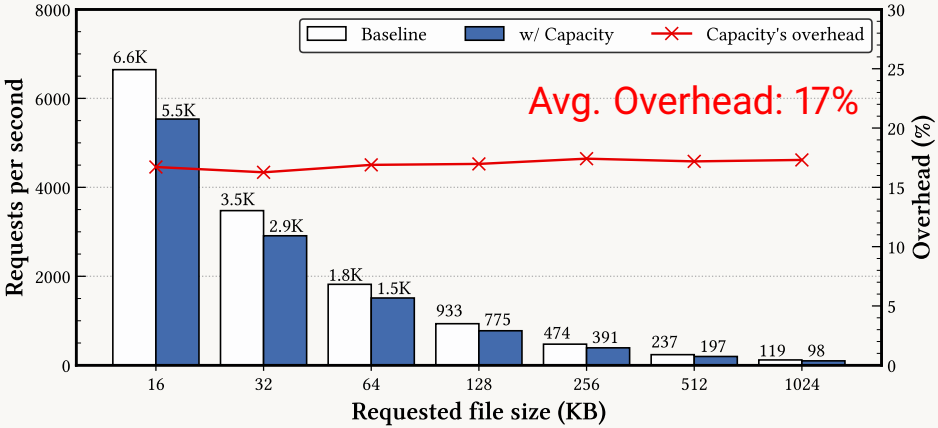
- **Microbenchmarks** on runtime library, reference monitor and instrumented code
  - Reference monitor incurs **2.65% slowdown** on average
- **Three real-world programs**: Nginx+LibreSSL, OpenSSH, and wget
- All experiments were conducted on M1 Mac Mini with Asahi Linux
  - PA is supported natively
  - MTE was *not* available in real hardware
    $\rightarrow$ functional evaluation (QEMU) + emulated overheads

NGINX Webserver prototype



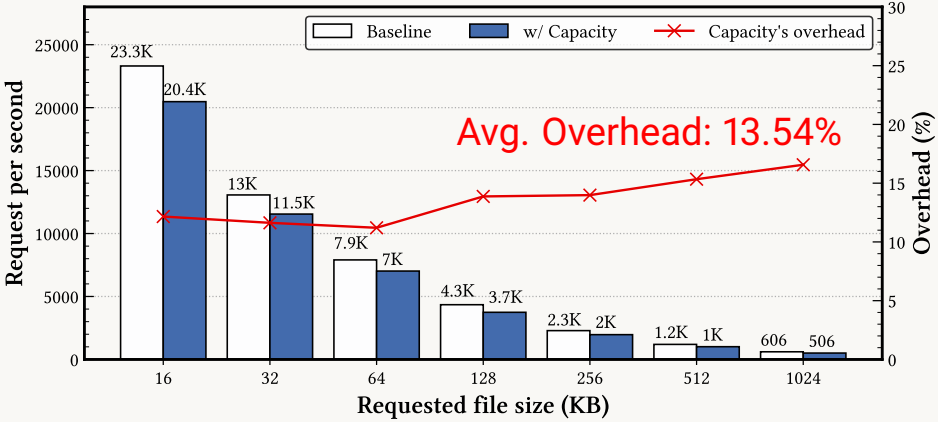
Lifecycle protection of server private key

Webserver prototype benchmark



Webserver throughput on single-threaded experiment

Webserver prototype benchmark



Webserver throughput on multi-threaded experiment

More details

- **Implementation details**
  - Maturing instrumentation framework to be compatible with real-world programs
  - Supporting capability token delegation among domains
- **Thorough security analysis**
  - How does CAPACITY prevent domain impersonation?
  - How difficult is it to forge the signed pointers?
  - etc...

👉 For more details, please check out our paper!

Thank you

Q&A time!!