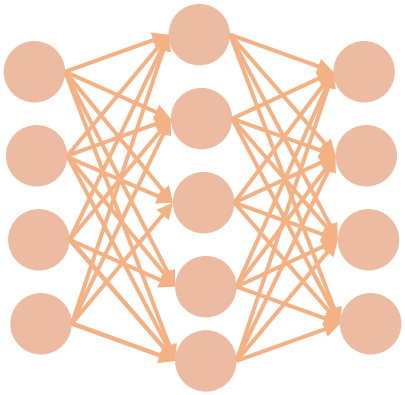
An Evaluation of Methods to Port Legacy Code to SGX Enclaves

Kripa Shanker, Arun Joseph, Vinod Ganapathy

Computer Systems Security Laboratory

IISc Bangalore

Trusted Execution Environment



Cloud Provider can steal user's or organization's machine learning model.

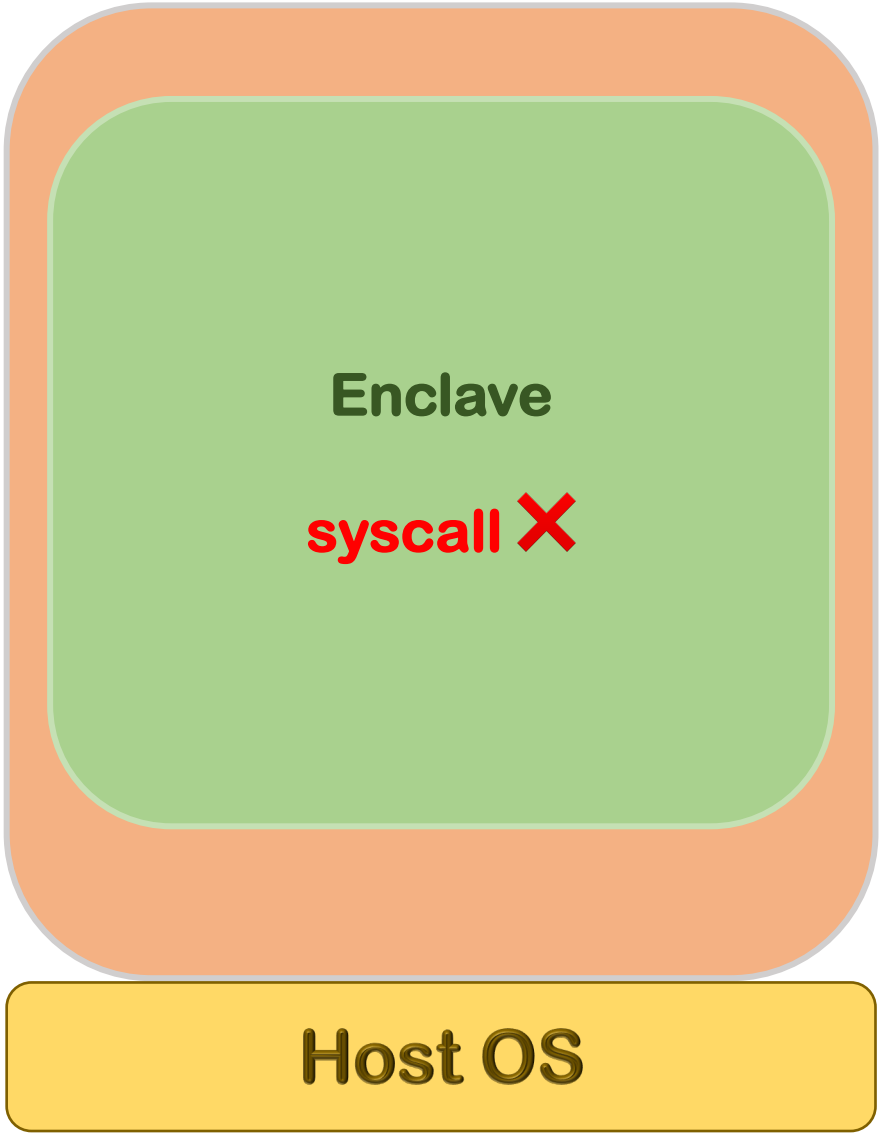
Image Courtesy: edureka.co

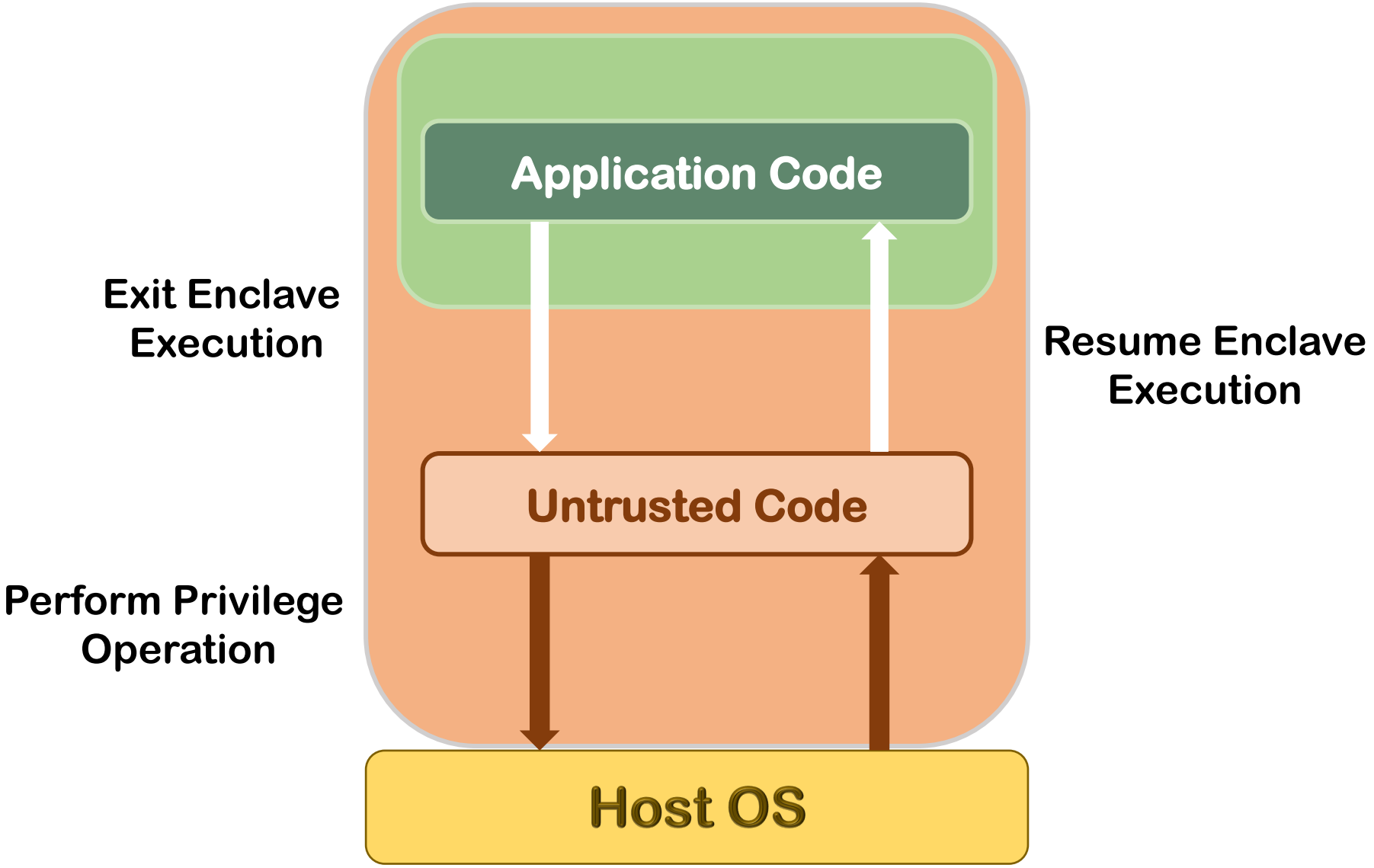
Intel SGX

SGX is a set of extensions to the Intel Instruction Set Architecture that creates enclaves.

Code and data that are placed inside this enclave are protected from malicious adversaries.

Applications must place their private code and data inside this enclave.




Application Code
Exit Enclave Execution
Resume Enclave Execution
Untrusted Code
Perform Privilege Operation
Host OS

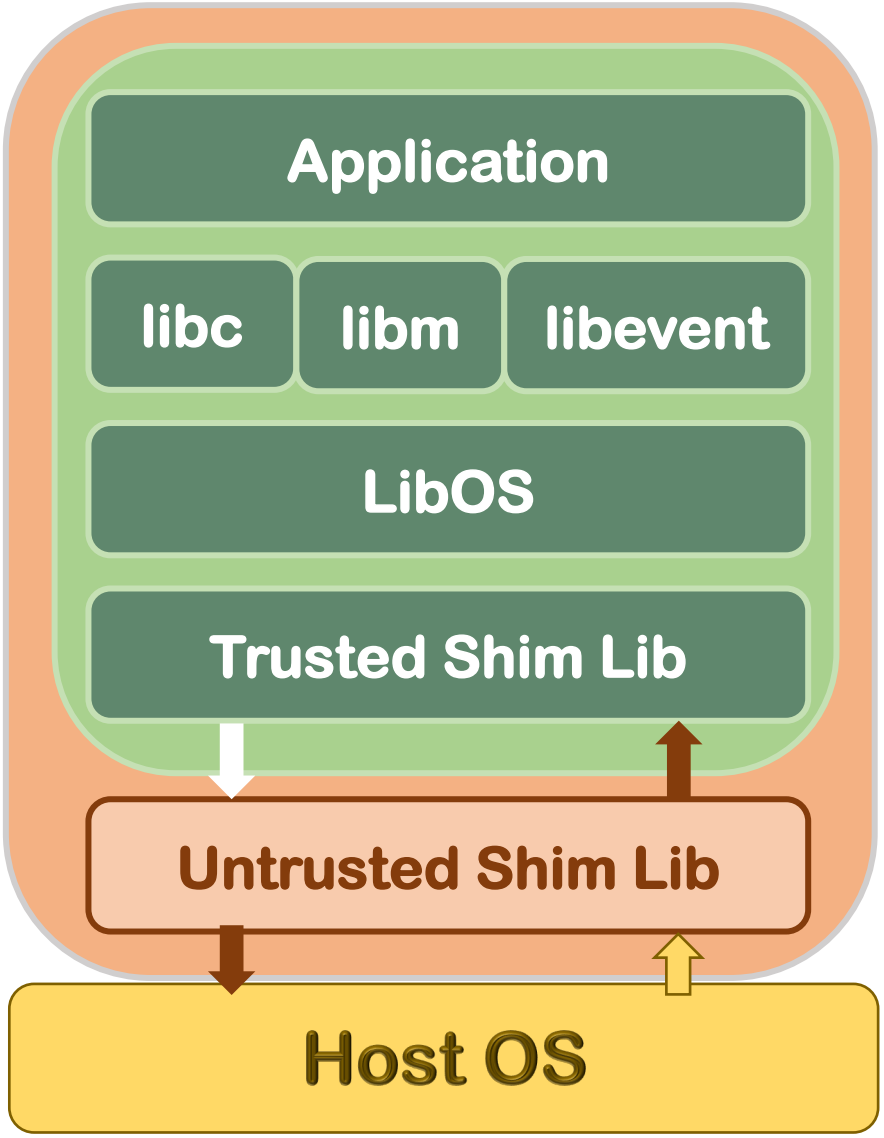
Contributions

Evaluated merits and costs of three methods to port applications to SGX Enclaves.

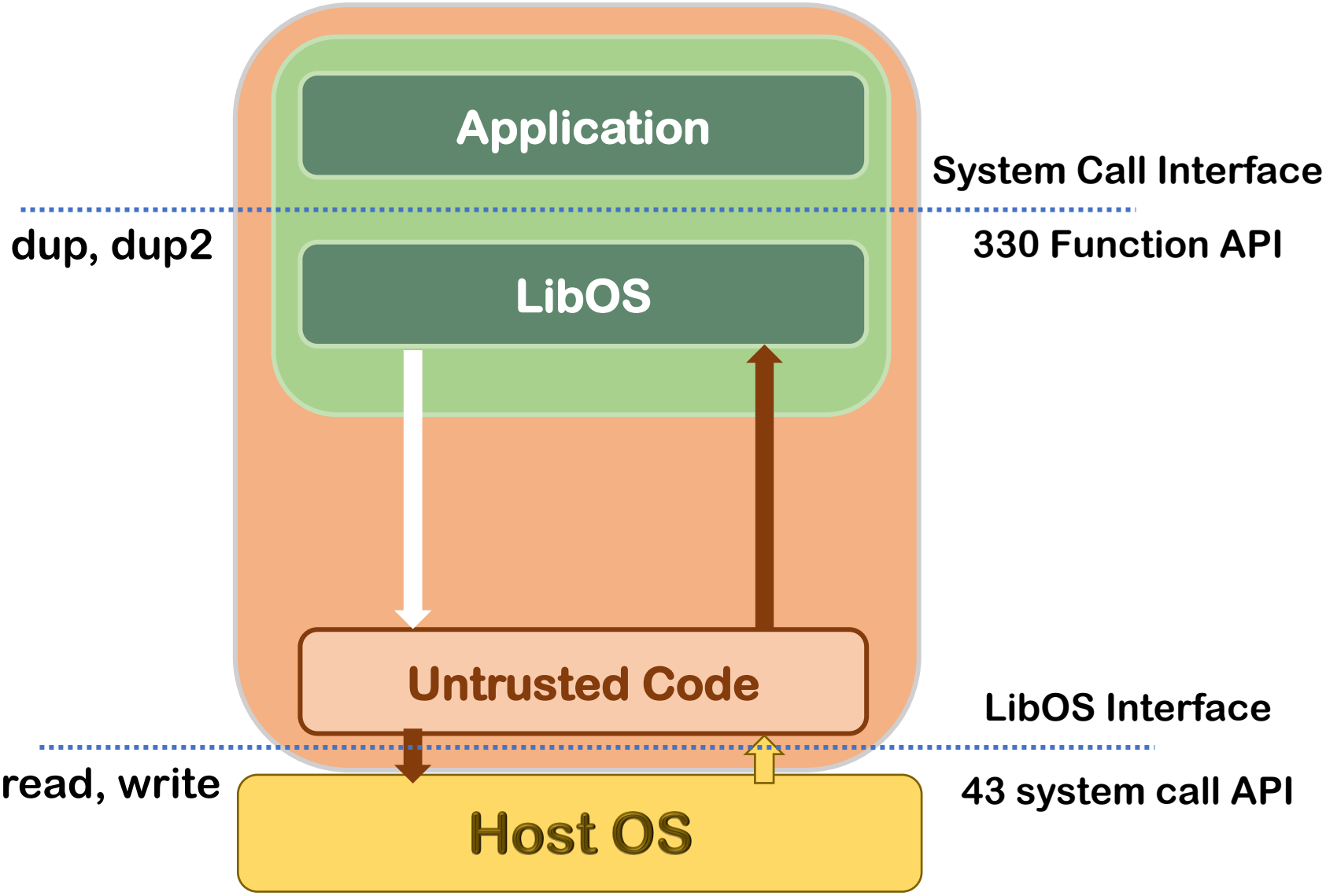
Library OS	Library Wrapper	Instruction Wrapper

1. What is the effort required, e.g., code changes, setting configurations, to obtain a working enclave?
2. What is the effort required to re-engineer a working enclave, e.g., by moving code out of or into the enclave?
3. How much trusted code does each method require?
4. What is the performance cost of each of the three methods?

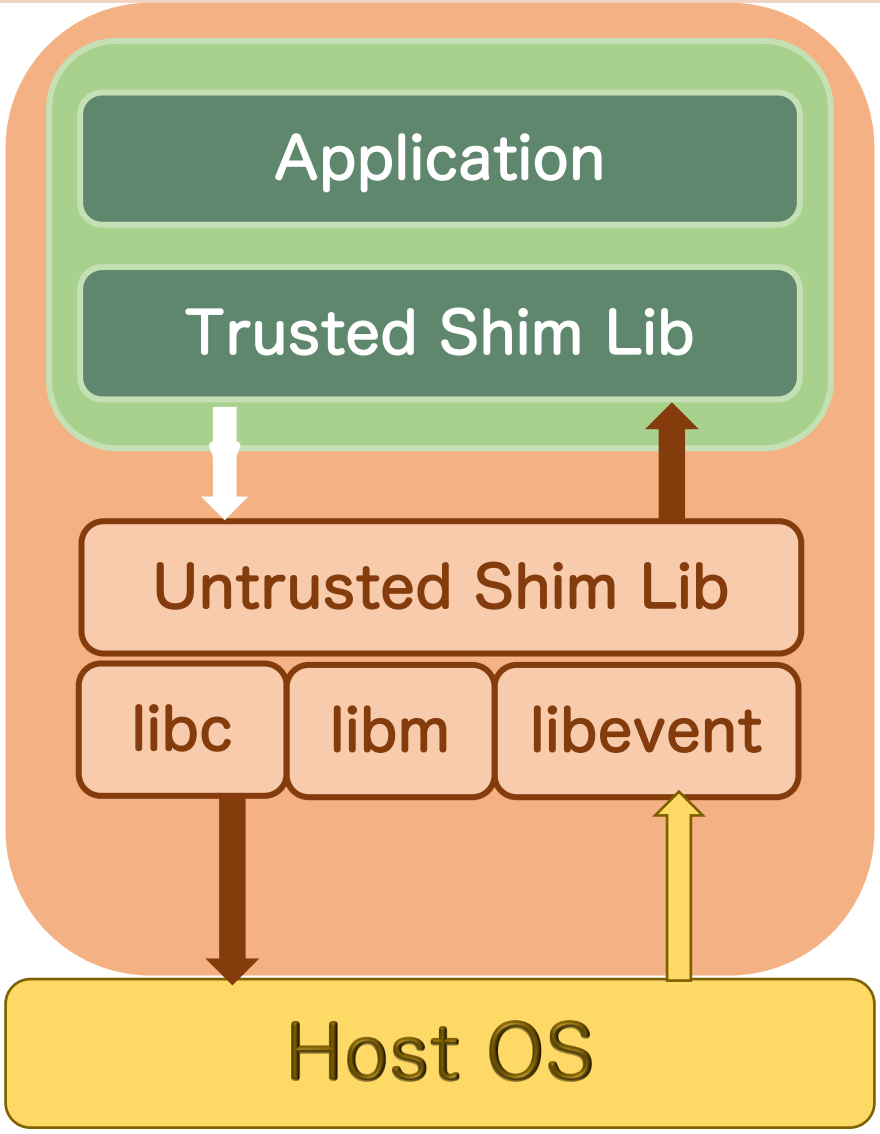
Method 1: Library OS



Library OS


Application
LibOS
Untrusted Code
Host OS
System Call Interface
330 Function API
LibOS Interface
43 system call API
dup, dup2
read, write

Method 2: Library Wrapper



Library Wrappers

```
main(argc, argv){
…
read(fd, buf, len);
…
}
```

Application

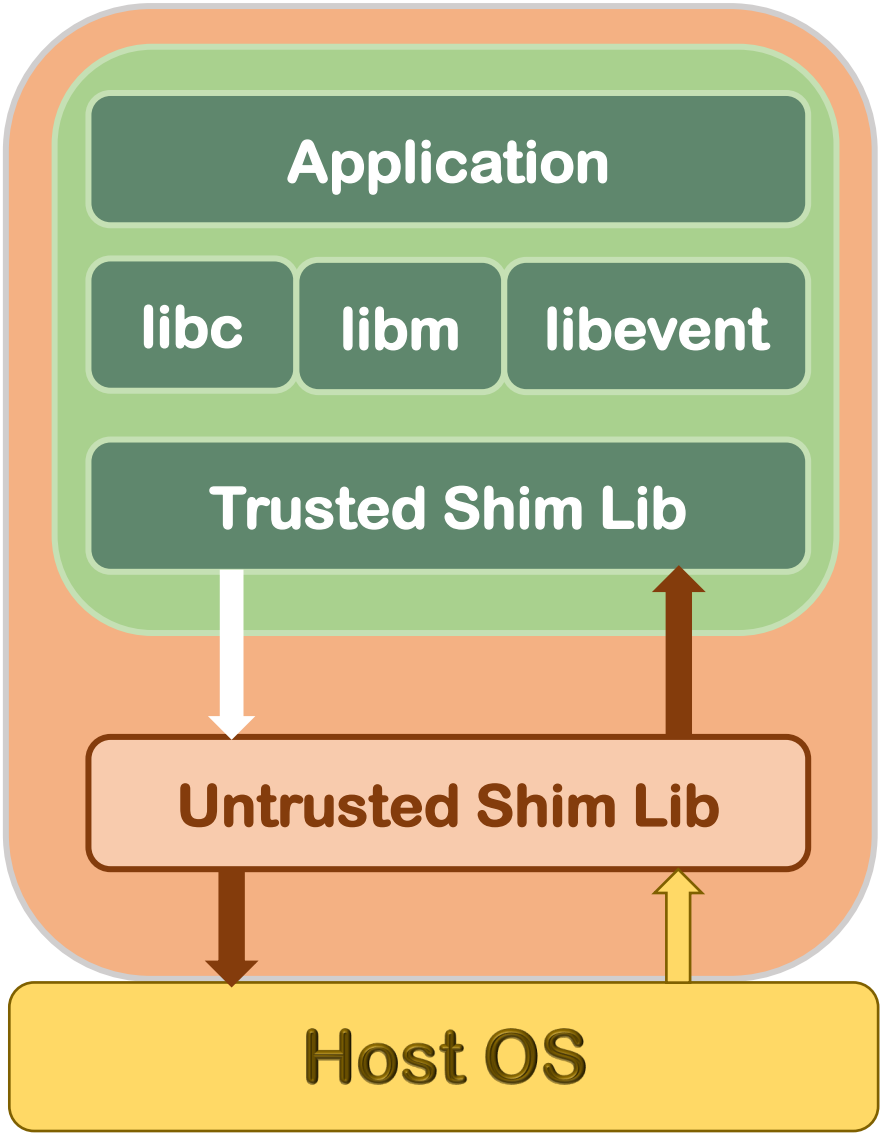
```
int read(int fd, char *buf, int len){
        return ocall_read(fd, buf, len);
}
```

Trusted Shim layer

```
int ocall_read(int fd, char *buf, int len){
        return read(fd, buf, len);
}
```

Untrusted Shim Layer

Method 3: Instruction Wrapper



```
__syscall:
        movq %rdi,%rax
        movq %rsi,%rdi
        movq %rdx,%rsi
        movq %rcx,%rdx
        movq %r8,%r10
        movq %r9,%r8
        movq 8(%rsp),%r9
        syscall
        ret
```

musl-libc: syscall.s

```
int __syscall(long arg0, ...){
/* copy arguments outside enclave */
...
ocall_syscall()
...
/* copy results back to enclave */
}
```

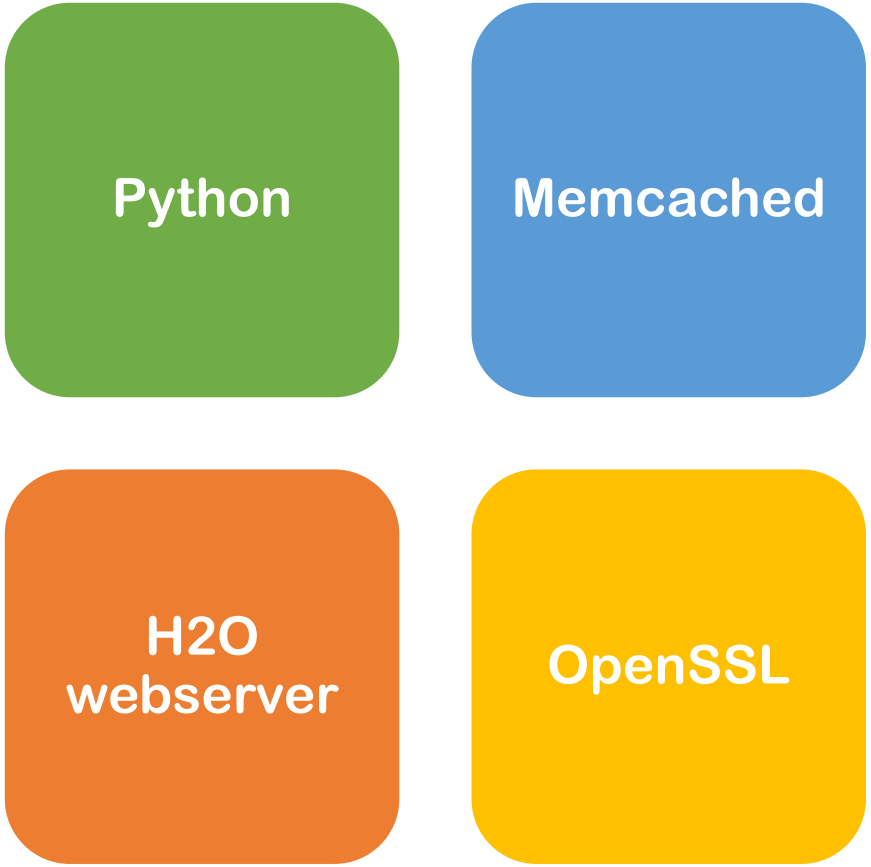
trusted shim layer

```
int ocall_syscall(){
        /* setup argument and perform system call
        and return results. */
        ...
        syscall(arg0, ...);
        }
```

Untrusted Shim Layer

RQ1: Porting Effort

Benchmark Applications



RQ1: Porting Effort

	Python	Memcached	H2O	OpenSSL
Library OS	✅	✅	✅	✅
Library Wrapper				
Instruction Wrapper				

The library OS takes care of system call required by Application.

RQ1: Porting Effort

	Python	Memcached	H2O	OpenSSL
Library OS	✅	✅	✅	✅
Library Wrapper	❌	❌	✅	✅
Instruction Wrapper				

Missing Library Wrappers

Deeply Nested Structures

API changes across version

Missing Library Wrappers

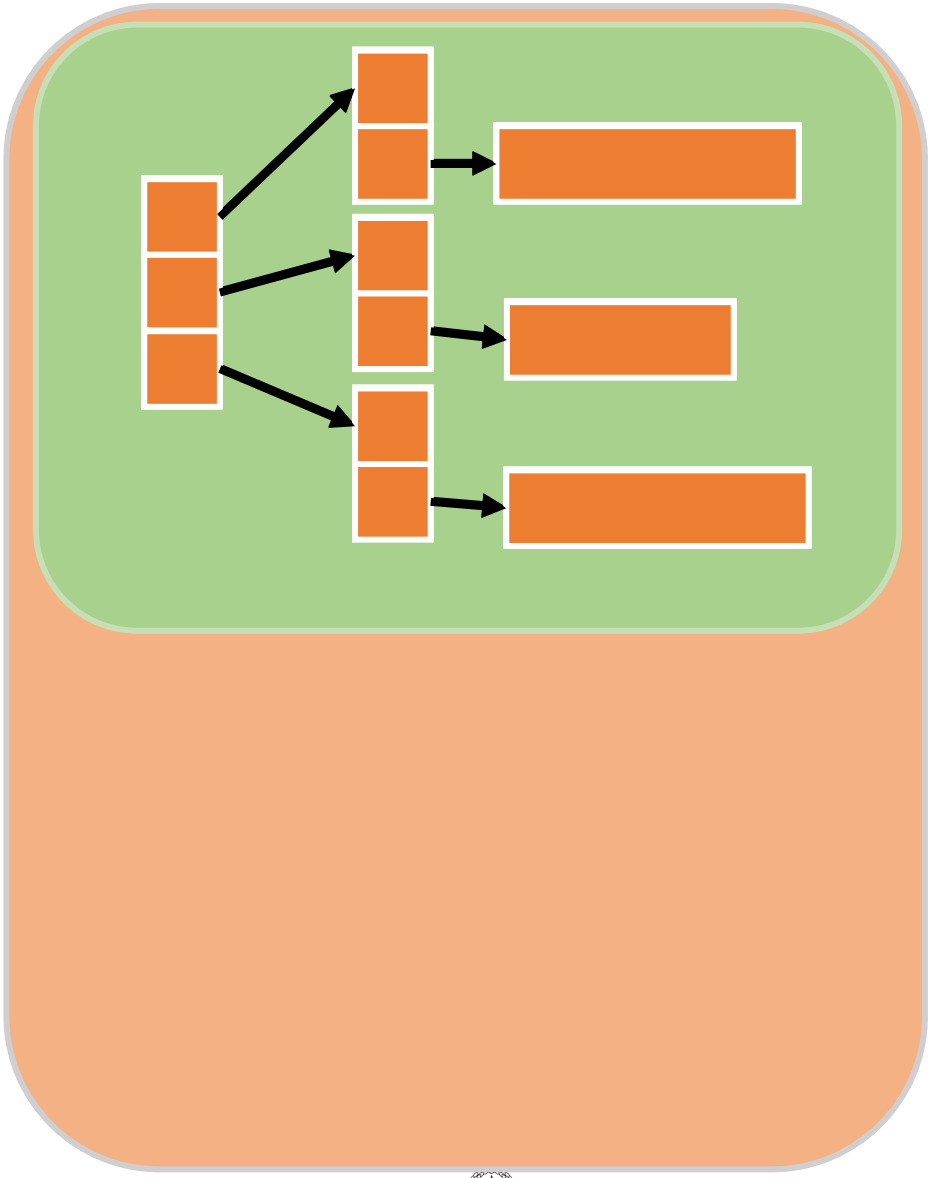
	Number of API
glibc	2021
libevent	69

Library wrappers are need for all the depend libraries used by application.

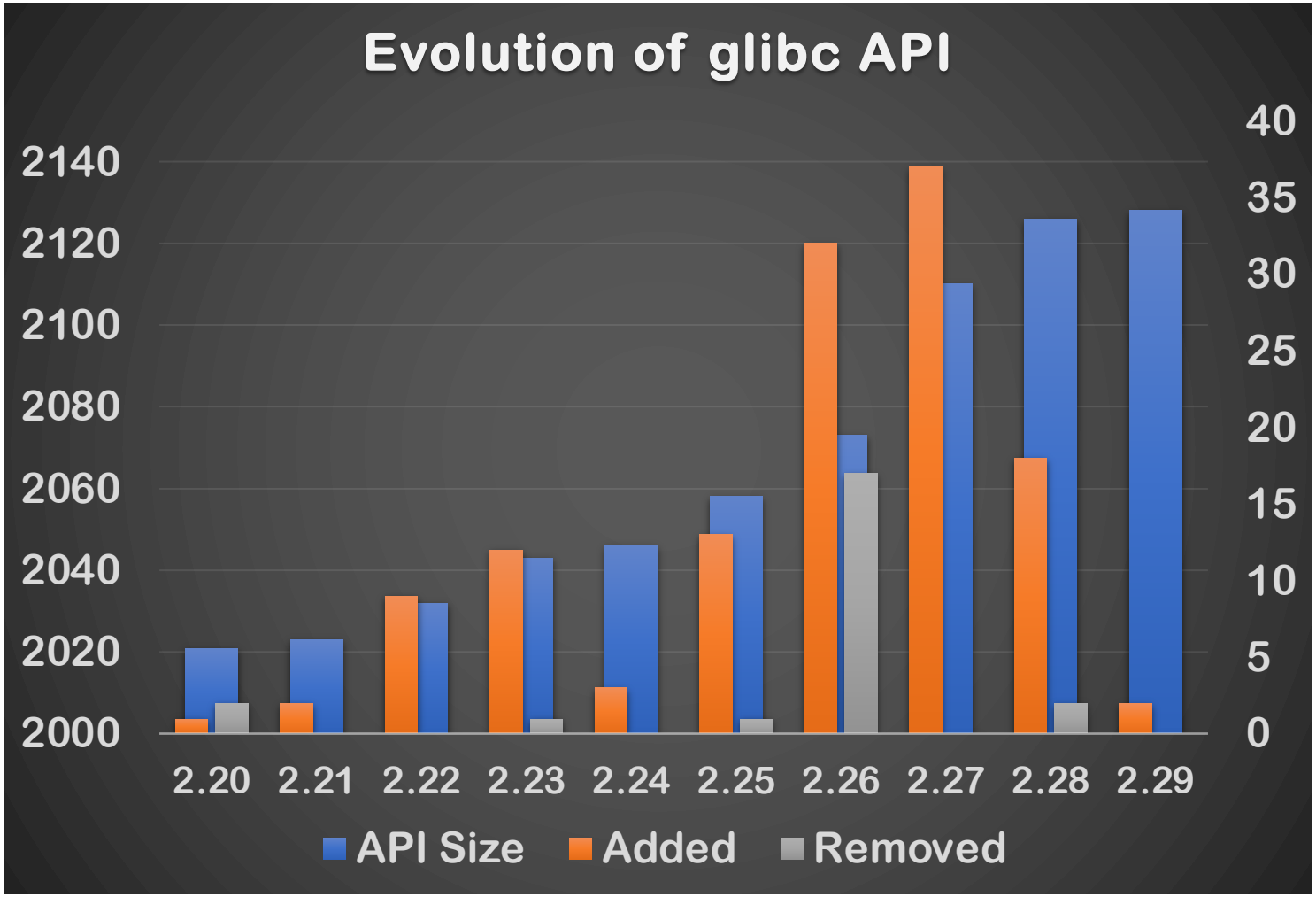
Deeply Nested Structure

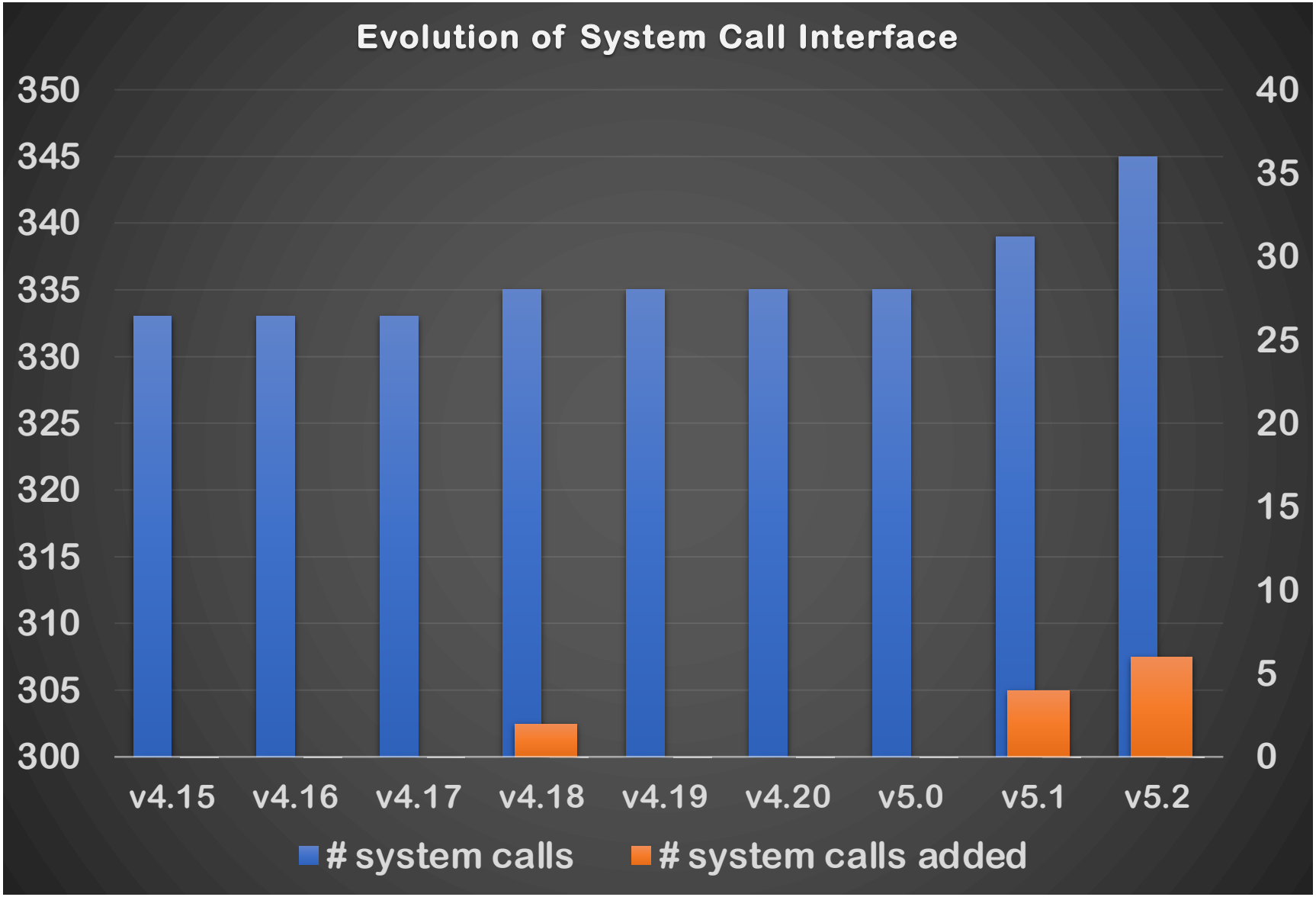
```
ssize_t readv(int fd,
const struct iovec *iov, int iovcnt);

struct iovec{
        void  *iov_base;
        size_t iov_len;
}
```



API Changes across versions



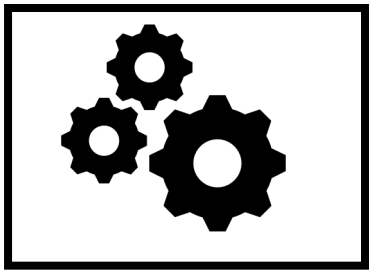

Evolution of System Call Interface
350
345
340
335
330
325
320
315
310
305
300
40
35
30
25
20
15
10
5
0
v4.15
v4.16
v4.17
v4.18
v4.19
v4.20
v5.0
v5.1
v5.2
system calls
system calls added

RQ1: Porting Effort

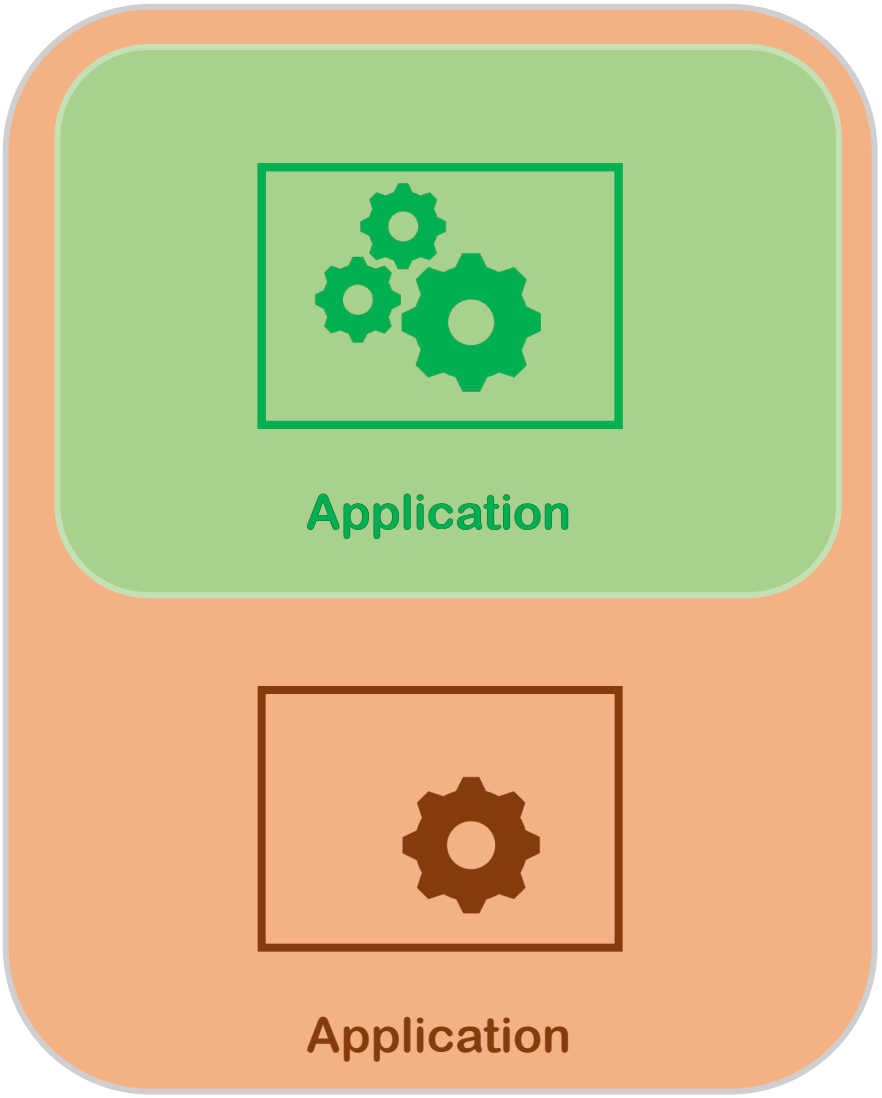
	Python	Memcached	H2O	OpenSSL
Library OS	✅	✅	✅	✅
Library Wrapper	❌	❌	✅	✅
Instruction Wrapper	✅	✅	✅	✅

System call API is small and changes slowly

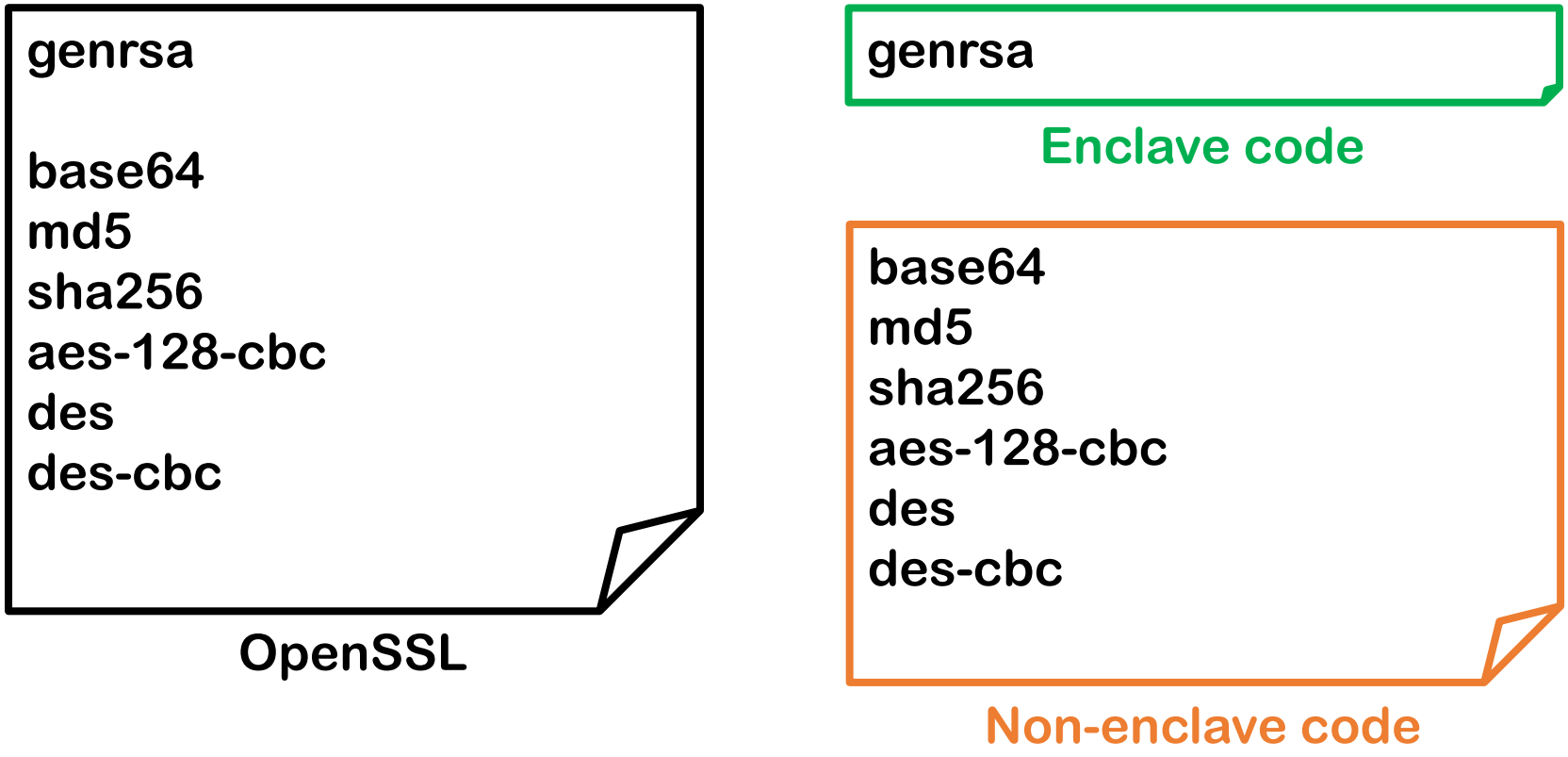
RQ2: Application Re-Engineering Effort



Application



OpenSSL is divided such that RSA key generation algorithm run inside enclave



RQ2: Application Re-Engineering Effort

Library OS	• Applications can't be re-engineered
Library Wrapper	• Need wrapper for ecall interfaces
Instruction Wrapper	• Need Wrappers for ecall interfaces

Ecall Interface

filename: enclave.edl

```
public int ecall_genrsa_main(int argc, [user_check]char **argv);
```

filename: genrsa.c

```
int MAIN(int argc, char **argv)
{
    return wrap_genrsa_main(argc, argv);
}
```

filename: function_wrapper.cpp

```
int wrap_genrsa_main(int argc, char *argv[]){
    int ret = -1;

    sgx_status_t status = SGX_ERROR_UNEXPECTED;
    status = ecall_genrsa_main(enclave_id, &ret, argc, argv);
    assert(status == SGX_SUCCESS);

    return ret;
}
```

RQ3: TCB Size

Trusted Code Inside Enclave

	Library OS	Library Wrapper	Instruction Wrapper
LibOS	31,742	n/a	n/a
libc	1,222,912	n/a	82,978
Shim	n/a	14,506	1,934
SDK	n/a	119,545	119,545

Library OS > Instruction Wrapper > Library Wrapper

Conclusion

We conclude that there is no one porting model that works best for all the four research questions that we considered.

Rapid Prototyping

- Library OS ✅
- Library Wrapper ❌
- Instruction Wrapper ✅

Evolution

- Library OS ✅
- Library Wrapper ❌
- Instruction Wrapper ✅

Flexibility to Re-engineer

- Library OS ❌
- Library Wrapper ✅
- Instruction Wrapper ✅

Source Code not Available

- Library OS ✅
- Library Wrapper ❌
- Instruction Wrapper ❌

We

Kripa
kripashanker@iisc.ac.in

Arun
arunj@iisc.ac.in

Vinod
vg@iisc.ac.in

Thank you very much for your Patience.

Checkout our Paper

Checkout our Artifacts