29th USENIX Security Symposium
August 12, 2020

UNIVERSITÄT DUISBURG ESSEN

Open-Minded

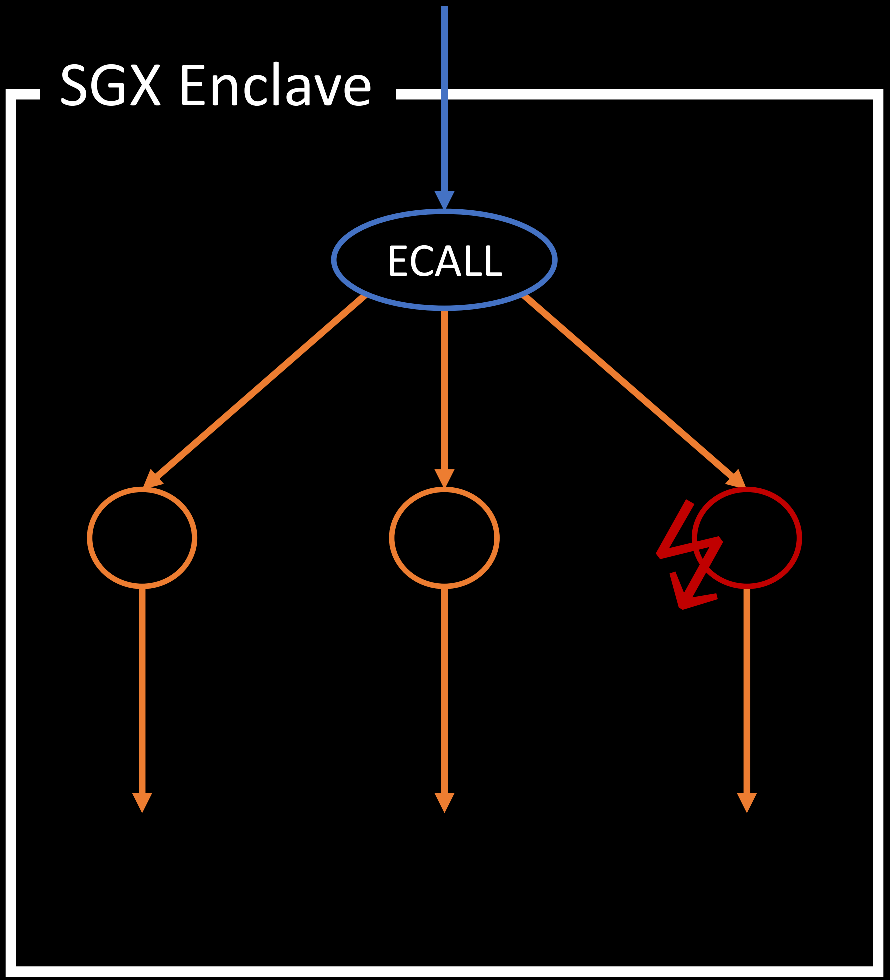
TEEREX:
Discovery and Exploitation of Memory Corruption Vulnerabilities in SGX Enclaves

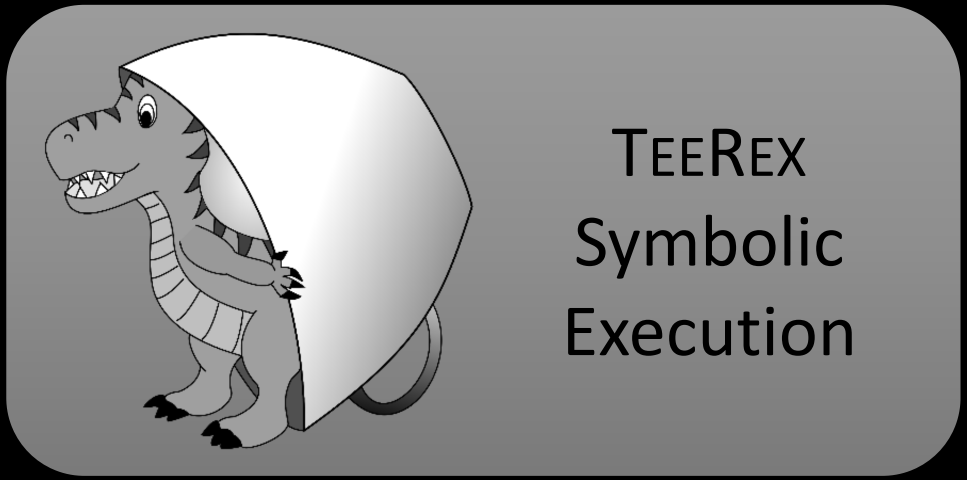
Tobias Cloosters, Michael Rodler, Lucas Davi

Secure Software Systems

University of Duisburg-Essen, Germany

Find memory corruption vulnerabilities specific to SGX enclaves





Successfully exploited:

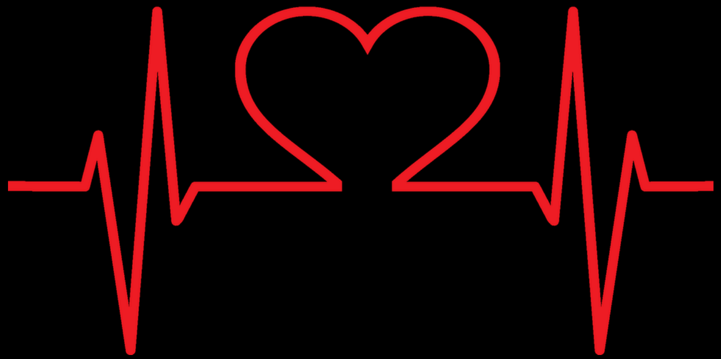
- Code from Intel, Baidu/Apache, WolfSSL
- Fingerprint Drivers
  - Synaptics (Lenovo/HP): CVE-2019-18619
  - Goodix (Dell): CVE-2020-11667

Motivation: Why SGX?

- How to reliably protect sensitive data and code from disclosure and modification?

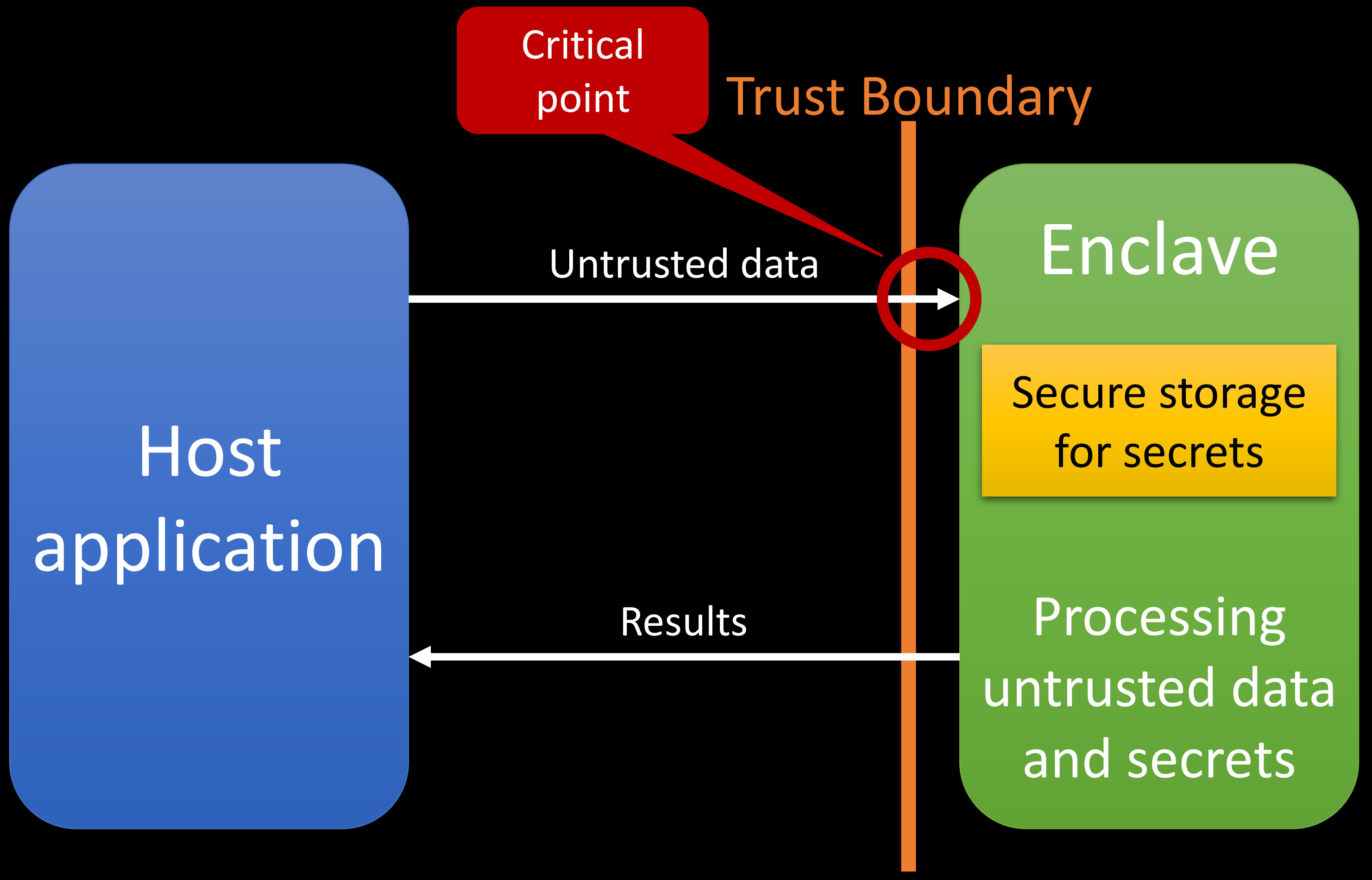
Passwords

Intellectual Property



Medical records

System Model of SGX



Critical point

Trust Boundary

Host application

Untrusted data

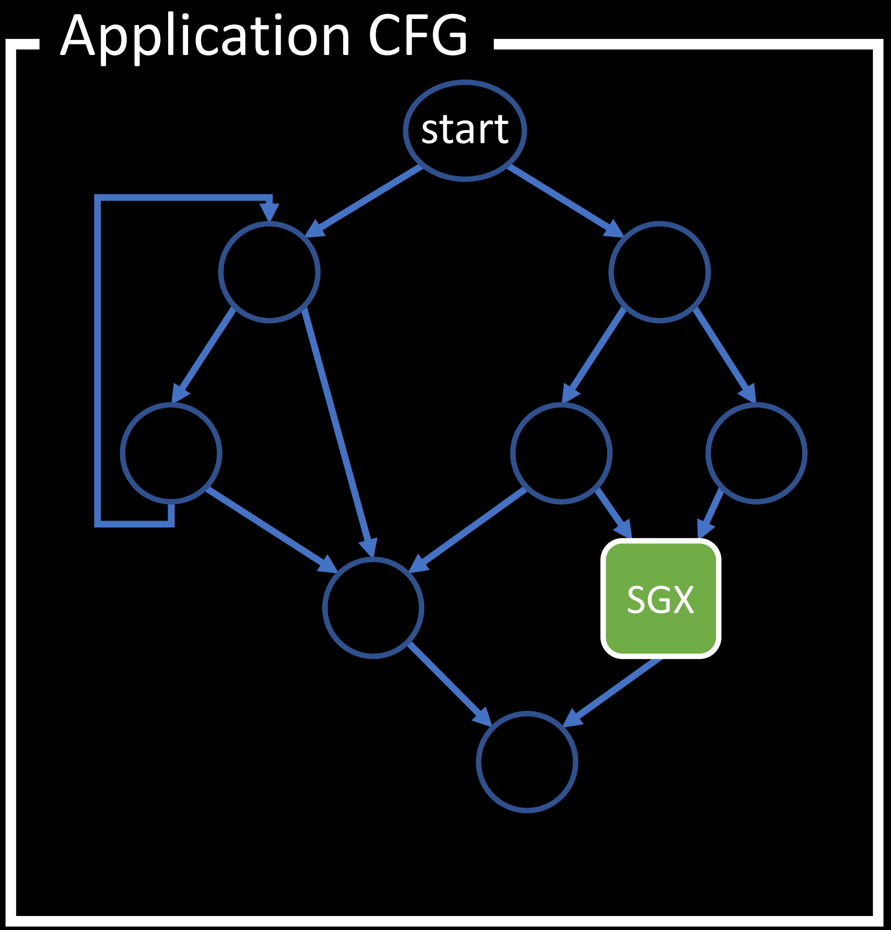
Results

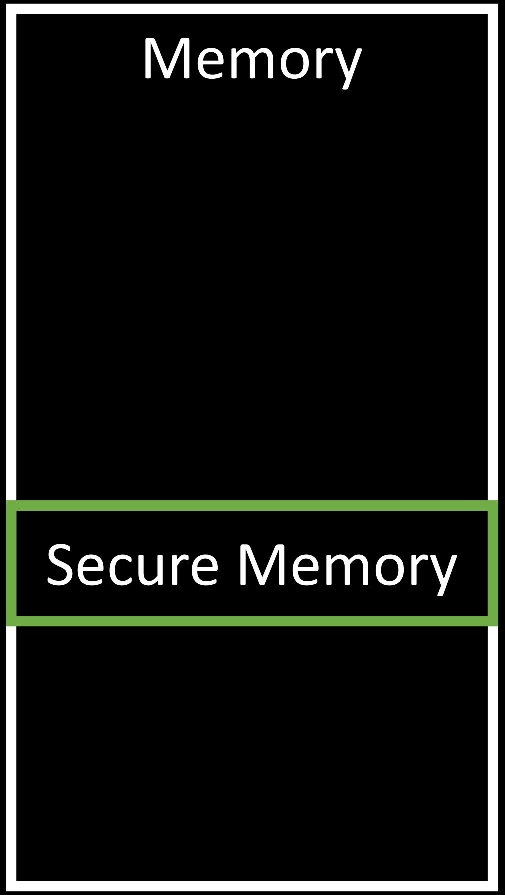
Enclave

Secure storage for secrets

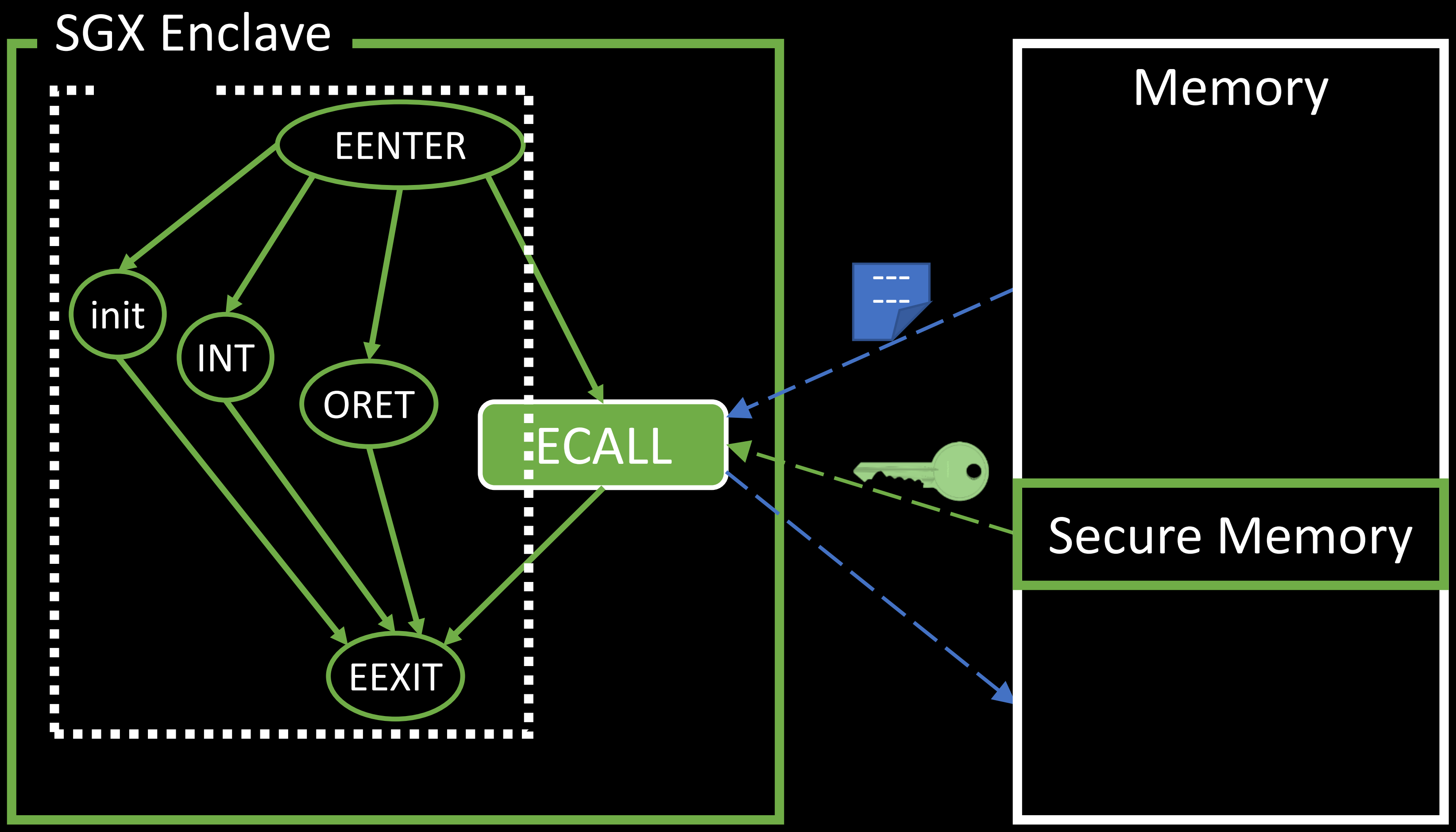
Processing untrusted data and secrets

SGX – Application Layout

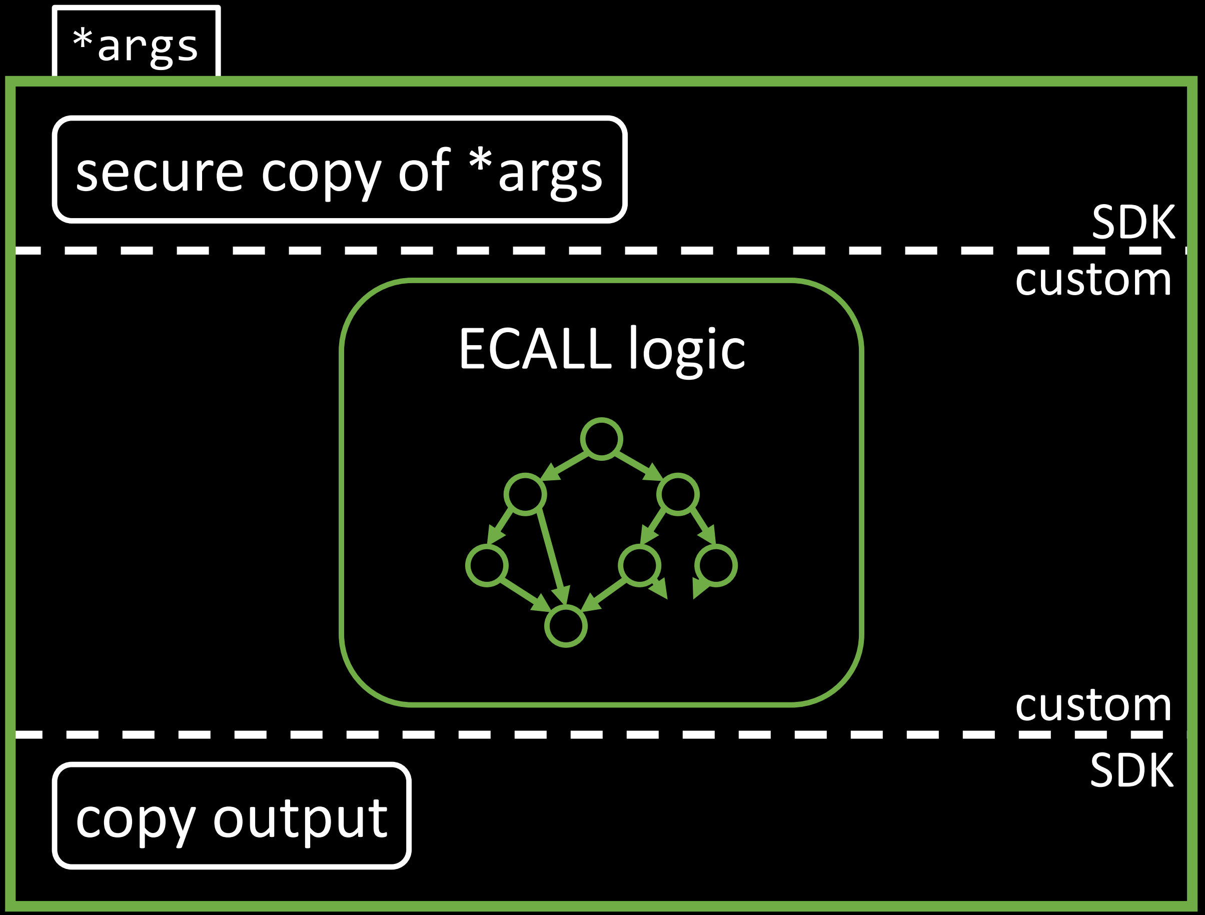

Application CFG



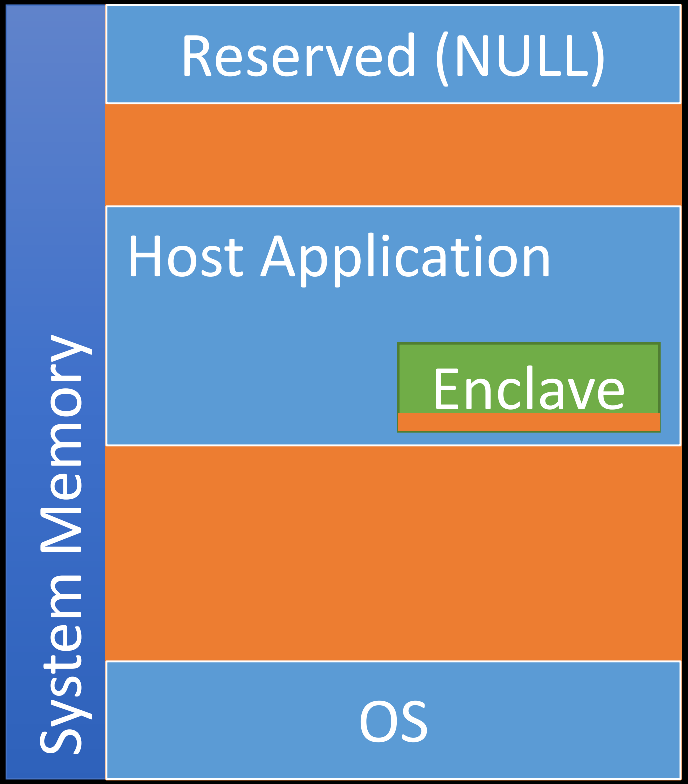
SGX – Trusted Runtime



SGX – ECALL

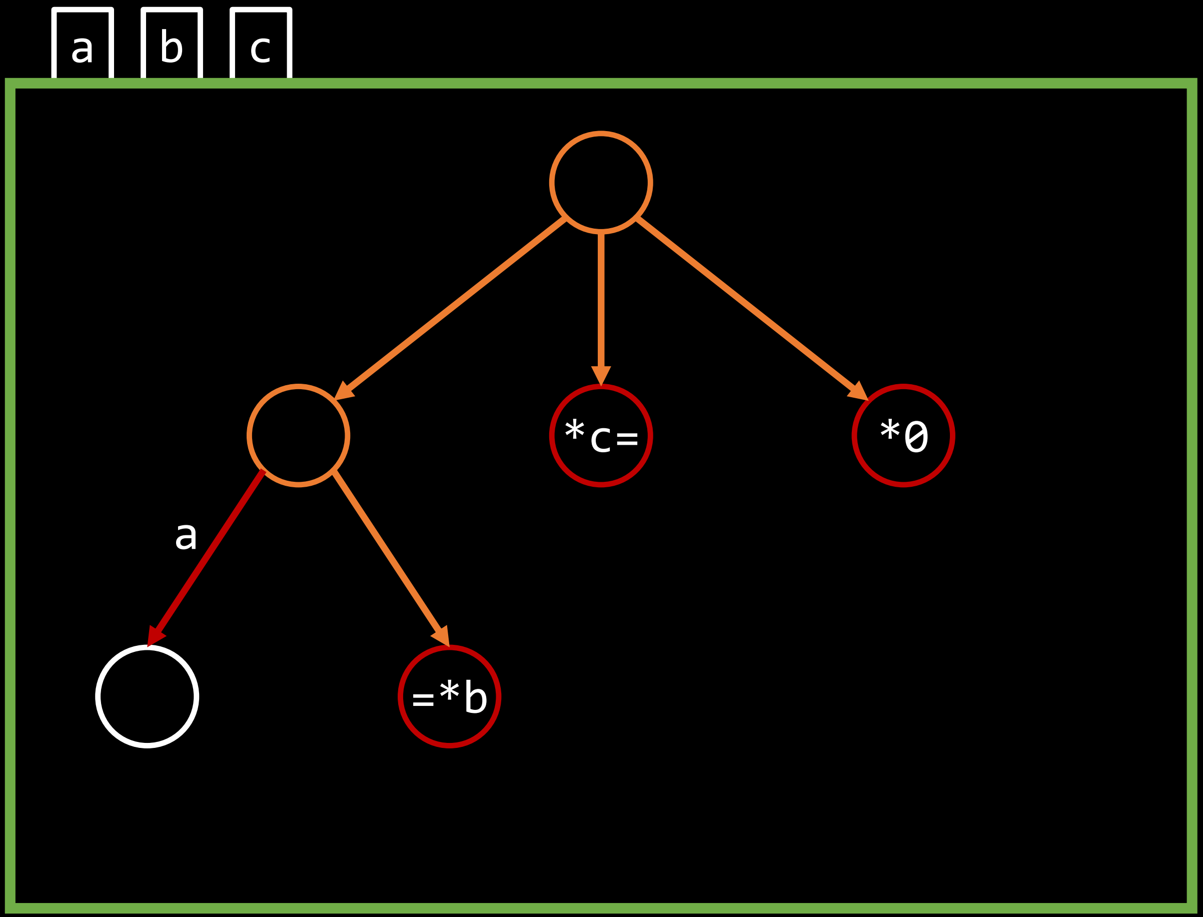

*args
secure copy of *args
SDK
custom
ECALL logic
custom
SDK
copy output

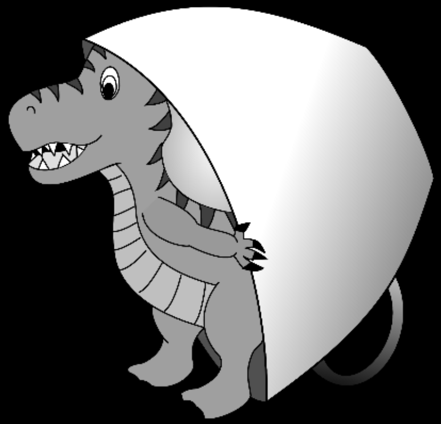
Large attack surface



- Trust input data: exploitable
- Trust system calls: exploitable
- Use NULL-pointer: exploitable
- One corruptible byte: exploitable
- Trust pointers to enclave memory: exploitable

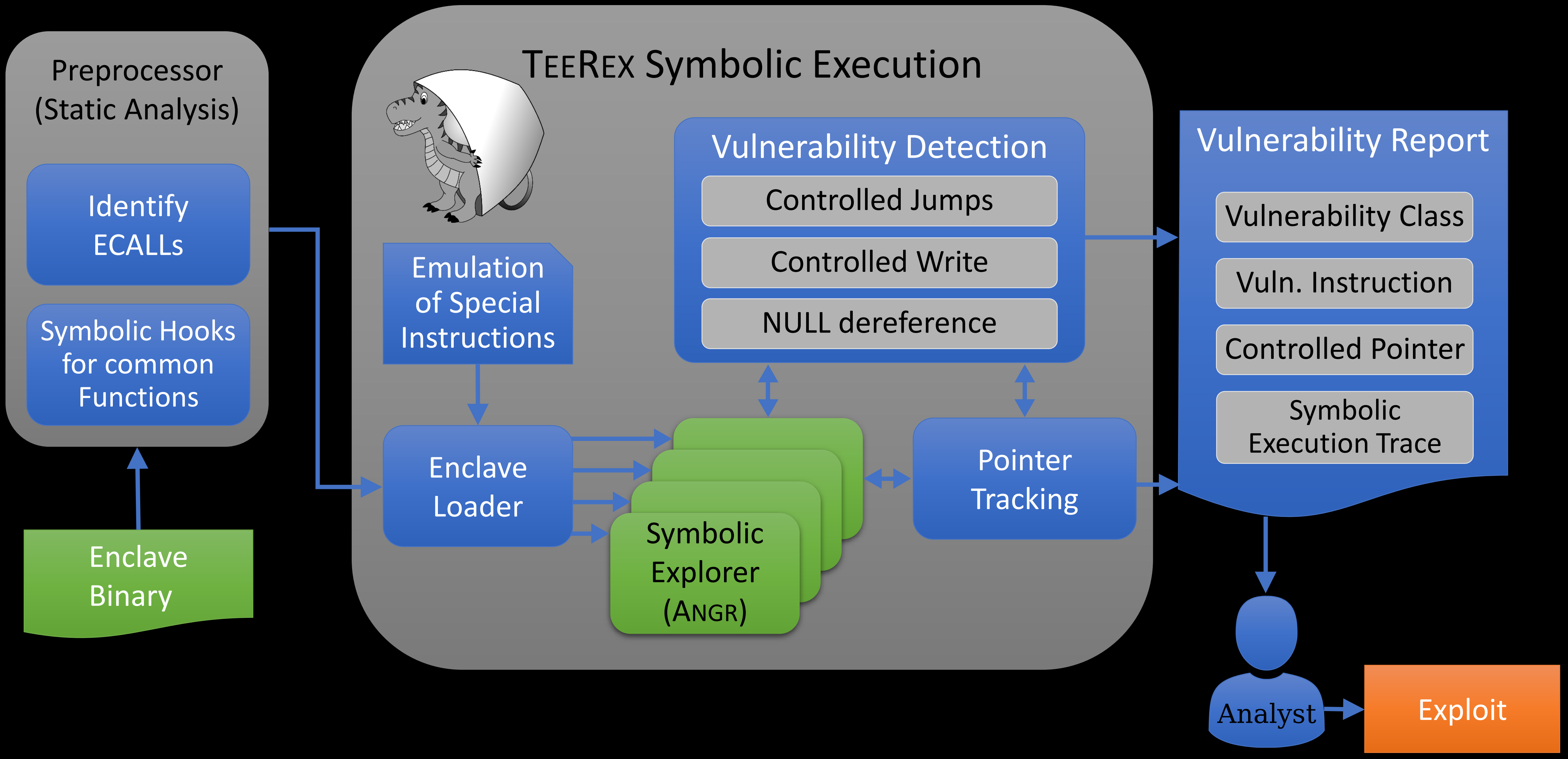
Symbolic Execution
Vulnerability Detection





- Controlled Jump
- Controlled Memory Access
- NULL-pointer Dereference

TeeRex Architecture



Preprocessor (Static Analysis)

Identify ECALLs

Symbolic Hooks for common Functions

Enclave Binary

TEEREX Symbolic Execution

Emulation of Special Instructions

Enclave Loader

Symbolic Explorer (ANGR)

Vulnerability Detection

Controlled Jumps

Controlled Write

NULL dereference

Pointer Tracking

Vulnerability Report

Vulnerability Class

Vuln. Instruction

Controlled Pointer

Symbolic Execution Trace

Analyst

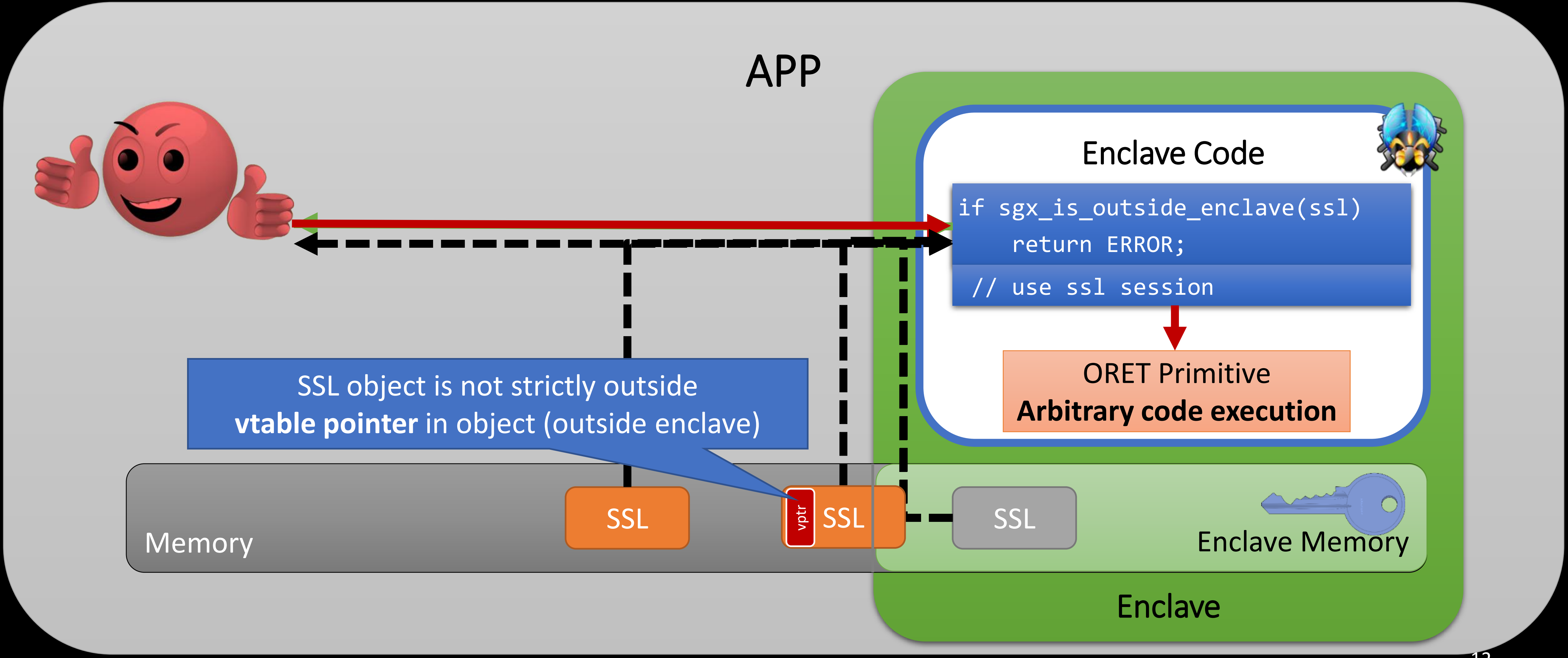
Exploit

Exploits in Public Enclaves found with TeeRex

Project	Exploit	Fixed	Source Code	Target
Intel SGX GMP Example	√	√	√	Linux amd64
Baidu Rust SGX SDK “tlsclient”	√	√	√	Linux amd64
TaLoS	√	Not planned	√	Linux amd64
WolfSSL Example Enclave	√	√	√	Linux amd64
Synaptics Fingerprint Driver	√	√	×	Windows amd64 CVE-2019-18619
Goodix Fingerprint Driver	√	√	×	Windows amd64 CVE-2020-11667
SignalApp Contact Discovery	×	-	√	Linux amd64

Exploit Source Code: https://github.com/uni-due-syssec/teerex-exploits

Baidu/Apache Rust SDK: tlsclient
Pointers to overlapping memory



```
if sgx_is_outside_enclave(ssl)
    return ERROR;
 // use ssl session
```

ORET Primitive
Arbitrary code execution
SSL object is not strictly outside
vtable pointer in object (outside enclave)
SSL
vptr
SSL
SSL
Memory
Enclave Memory
Enclave

Limited Exploit Primitives

```
int global_mem = 0;
int* global_addr;
```

```
void arbitrary_write(int* a, int b)
{
    *a = b;
}
```

```
void no_user_input()
{
    global_mem = 42;
}
```

Controlled Address

```
void limited_value(int* a)
{
OR: *a = 42;
    *a = global_mem;
}
```

```
void limited_size(int* a, char b)
{
    *(char*)a = b;
}
```

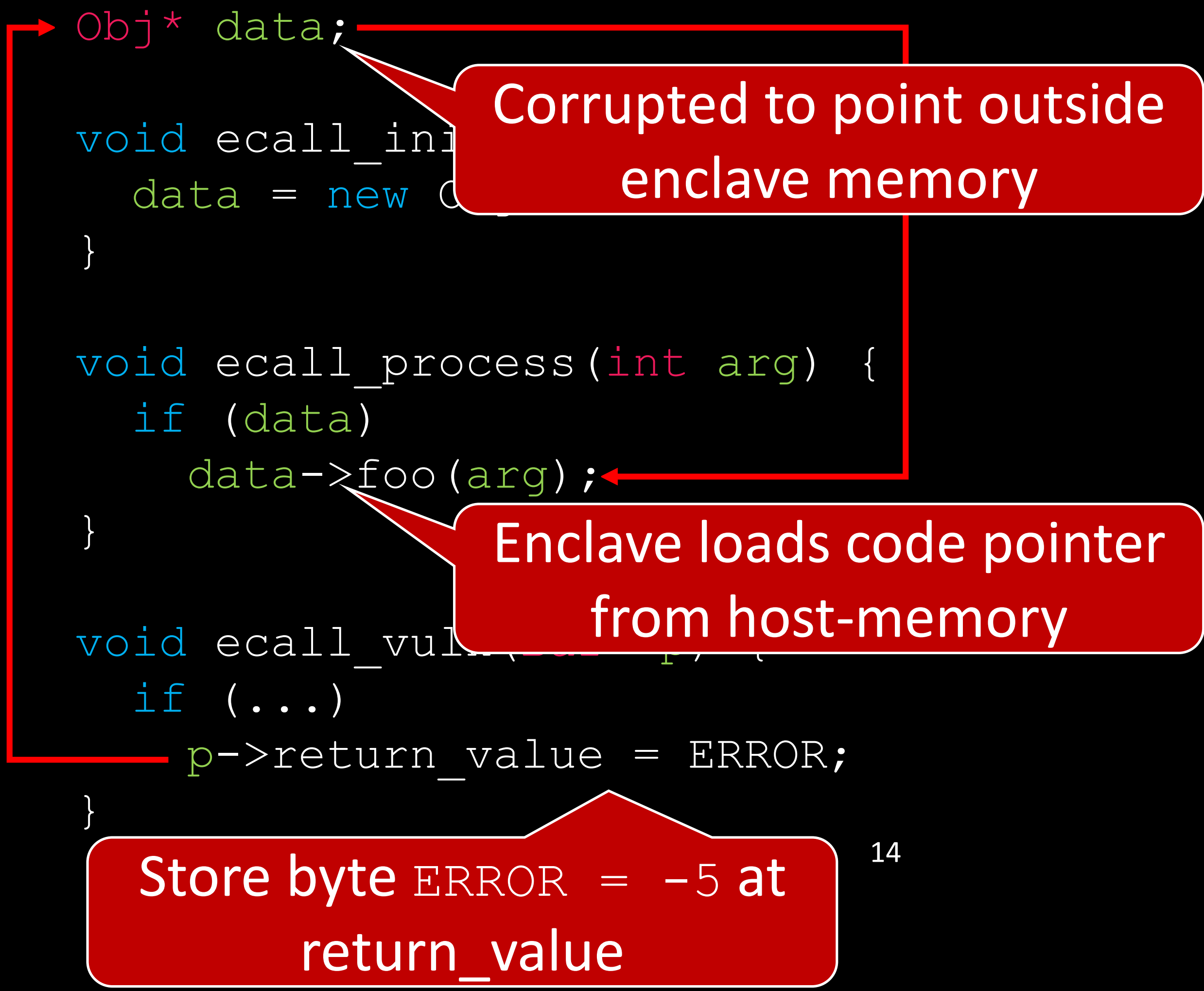
Controlled Value

```
void fixed_address(int b)
{
    global_mem = b;
}
```

```
void limited_address(int b)
{
    *global_addr = b;
}
```

Exploiting using Multiple Limited Primitives:

- `ecall_process` trusts `data`
  - In secure memory
  - Never leaves enclave
  - NULL checked
- Attacker can corrupt `data`
  - `ecall_vuln` writes a small constant to an unchecked address
  - Changing one byte moves `data` to unsecure memory
- Combined the attacker can execute arbitrary code



```
Obj* data;

void ecall_ini
  data = new O
}

void ecall_process(int arg) {
  if (data)
    data->foo(arg);
}

void ecall_vul
  if (...)
    p->return_value = ERROR;
}
```

Corrupted to point outside enclave memory

Enclave loads code pointer from host-memory

Store byte ERROR = -5 at return_value

Conclusions

- Enclave boundary is a highly critical attack surface
- Current development practices do not consider such vulnerabilities
  - Increase awareness
  - Automatic analysis tools needed

- TeeRex can automatically detect vulnerabilities!

29th USENIX Security Symposium
August 12, 2020

UNIVERSITÄT DUISBURG ESSEN

Open-Minded

Secure Software Systems
https://www.syssec.wiwi.uni-due.de/

Tobias Cloosters, Michael Rodler, Lucas Davi

tobias.cloosters@uni-due.de, michael.rodler@uni-due.de, lucas.davi@uni-due.de

TEEREX: Discovery and Exploitation of Memory Corruption Vulnerabilities in SGX Enclaves

Exploit Source Code: https://github.com/uni-due-syssec/teerex-exploits