Designing a Provenance Analysis for SGX Enclaves

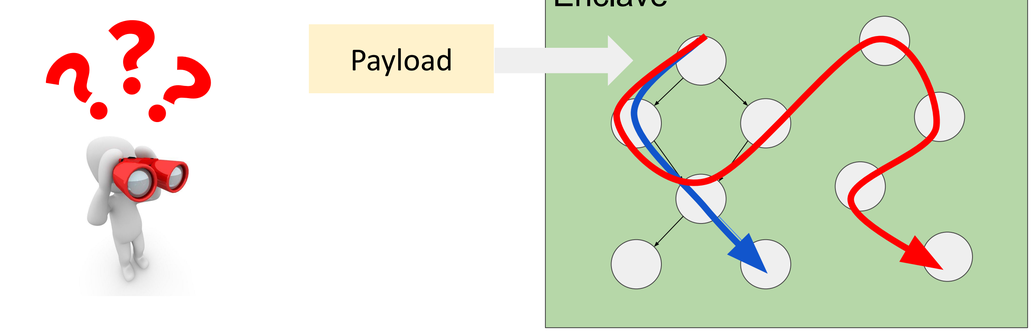
ACSAC 2022

Flavio Toffalini, Mathias Payer, Jianying Zhou, Lorenzo Cavallaro

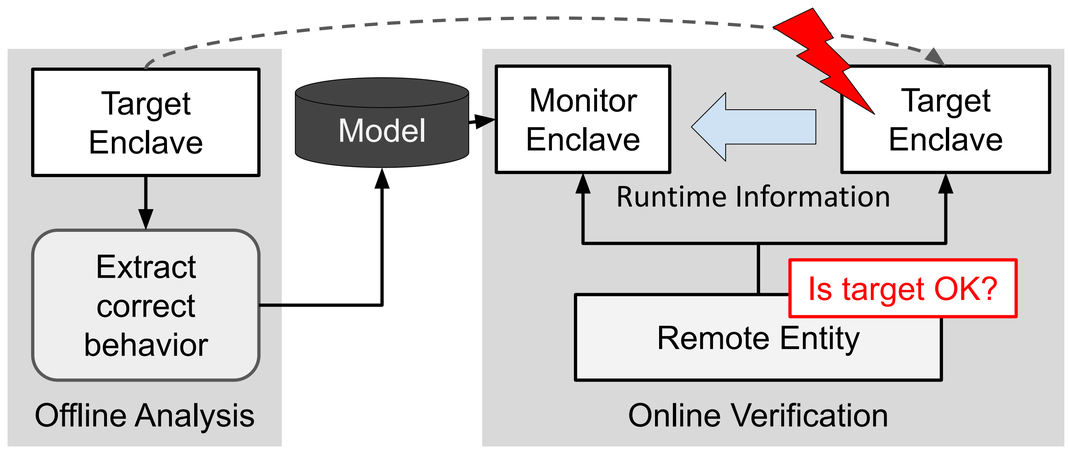
SGX Prohibits Provenance

Adversaries use memory corruption errors to mount code-reuse attacks

External observers cannot distinguish **correct** and **hijacked** executions



Design



Properties guaranteed:

1) Secure streaming of runtime information
2) Detecting code-reuse attacks

Design: Tracing

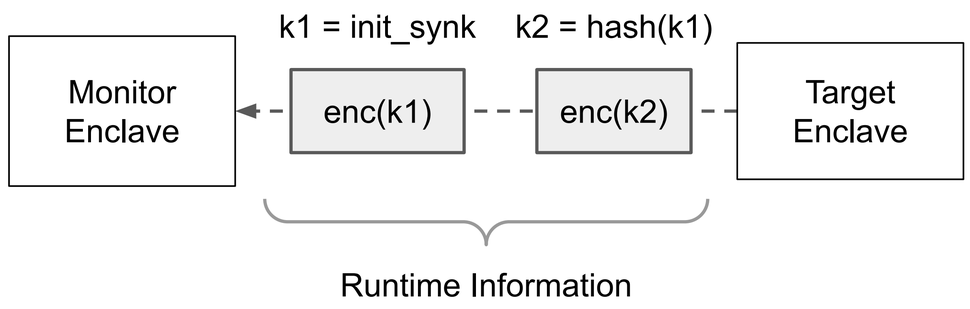
```
int fun(int a) {
 …
 // trace the indirect
 // jump to the caller
 trace(ret_addr(0));
 return 0;
}
```

```
key_t prv_key;

void trace(uint8_t addr) {
   msg_t msg = encr(addr, prv_key);
   send_to_monitor(msg);
   prv_key = hash(prv_key);
}
```

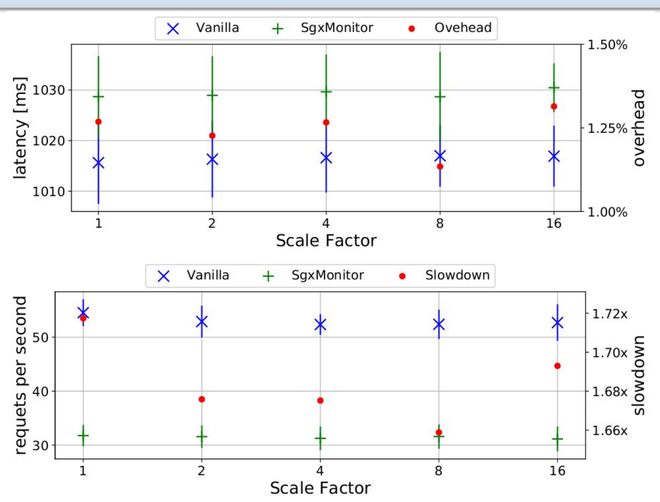
Monitor Enclave M

Design: Streaming



Messages are chained, losing one reveals an attack.
They all have the same size, so no information of their content.

Evaluation: Overhead



Macrobenchmark over StealthDB (PostgreSQL's SGX plugin) has limited overhead

Evaluation: Model

Use Case	# functions	% CFG explored	# fun. static
Contact	71	96.4%	1
libdvdcss	56	91.4%	9
StealthDB	44	96.6%	0
SGX-Biniax2	49	91.6%	4
Unit-test	17	94.0%	0

Symex explores the majority of the functions
We fallback to static analysis only for few cases

Partially Sponsored By

European Research Council
Established by the European Commission