A Survey on SGX Enclave Privileged Side-Channel Attacks

Amin Fallahi

Ph.D. Student

Syracuse University

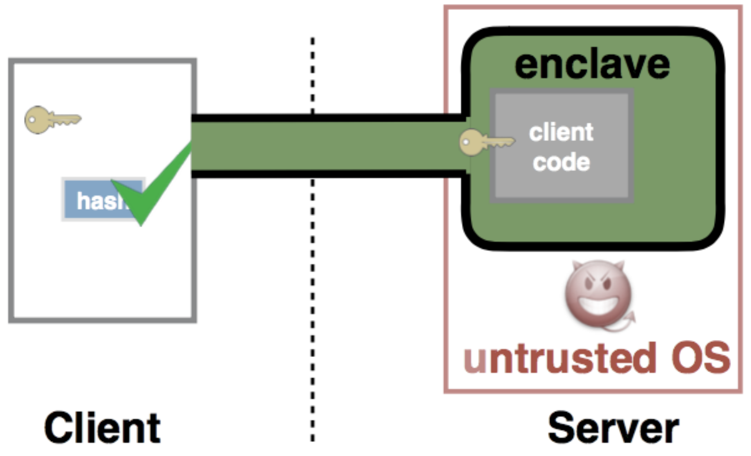
Department of Electrical Engineering and Computer Science

MAY 14, 2018

Contents

Introduction

- Computing environment
  - Shared resources executing user programs
- Security inside the cloud
  - A hardware solution needed
- Intel SGX
  - Secure user code and data in the public cloud
  - Protecting data inside the enclave
- Privileged Operating System
  - Kernel space, hardware control



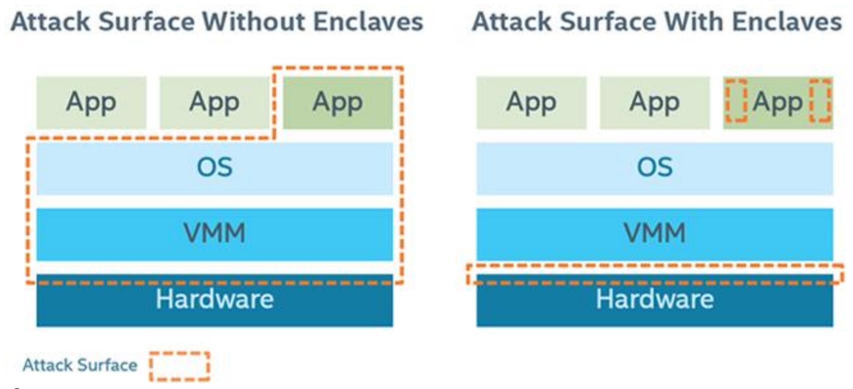
Introduction

Intel SGX

- 17 new instructions
- Enclave created by OS
  - Contiguous physical memory
  - Base address randomly assigned from EPC
- Hardware level memory encryption engine
- Virtual to physical address translation done by OS
- Interrupt ⇨ context switch – AEX
- ECALLs & OCALLs
- TPM: Trusted Platform Module
- TXT: Trusted Execution Technology

Introduction

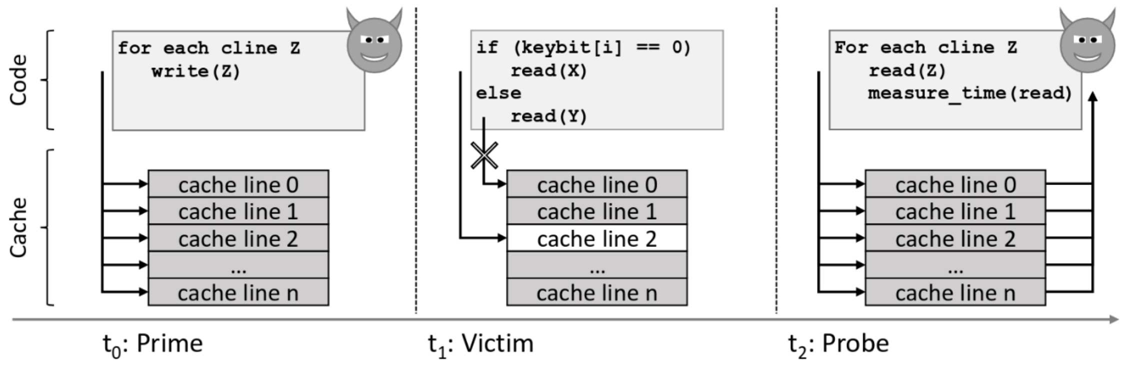
- Side-Channels
    - Attacks by analyzing system behavior
    - Cache attacks: Monitoring cache accesses
    - Timing attacks: Measuring time between computations
    - Page-fault attack: Monitoring page-faults and analyzing the pattern
- Attack Model
    - Enclave protected by SGX
        - No stranger has the key
    - Operating System can manage task queues, interrupts, and exceptions
    - Viewing data transfer between memories and caches
    - Program pinned to one core, no excessive interrupts, isolated



Attacks

Cache-Based Attacks

- Intentional Cache-Misses ⇨ Make program access memory
- Time, trace, access patterns
- Prime+Probe
  - Prime: attacker executes conflicting cache lines ⇨ cache miss
  - Probe: executing all the cache lines and measuring execution time



```
for each cline Z
   write(Z)
```

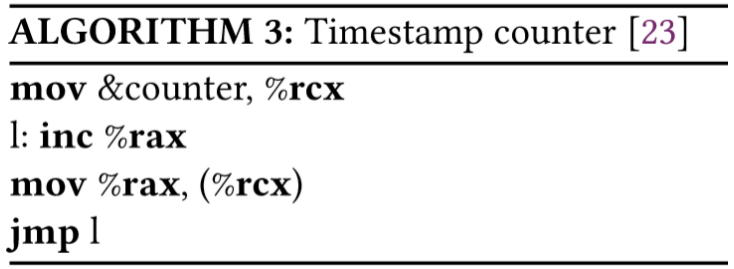
```
if (keybit[i] == 0)
   read(X)
else
   read(Y)
```

```
For each cline Z
   read(Z)
   measure_time(read)
```

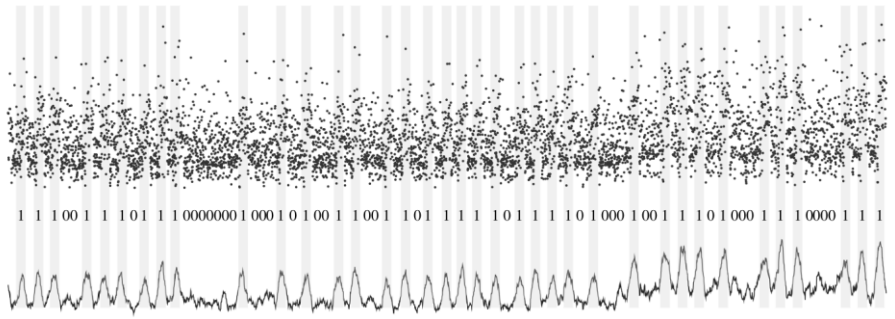
Attacks

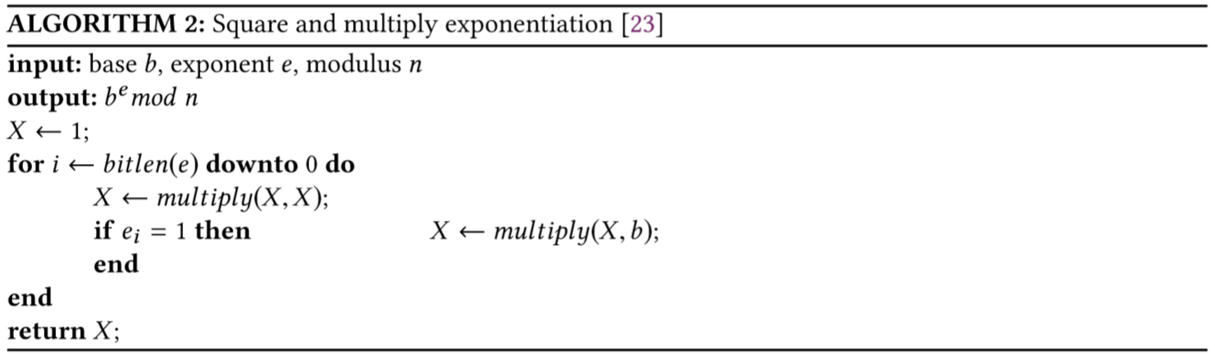
Cache-Based Attacks

- RSA key extraction
    - Modular exponentiation
    - Large e, d, and n such that $(m^e)^d \equiv m \ (mod\ n)$
    - Knowing m, e, and n it is extremely difficult to find d
- Square and multiply
    - Typical algorithm ⇨ find the exponent
    - Secret dependent memory accesses
- Challenges
    - High precision timer
    - Eviction set generation


ALGORITHM 3: Timestamp counter [23]

```
mov &counter, %rcx
l: inc %rax
mov %rax, (%rcx)
jmp l
```




ALGORITHM 2: Square and multiply exponentiation [23]

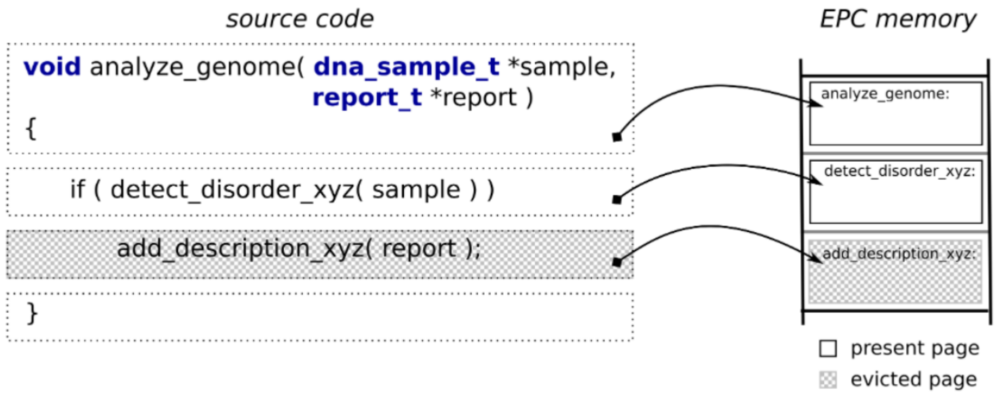
```
input: base $b$, exponent $e$, modulus $n$
output: $b^e \ mod \ n$
$X \leftarrow 1$;
for $i \leftarrow bitlen(e)$ downto 0 do
    $X \leftarrow multiply(X, X)$;
    if $e_i = 1$ then        $X \leftarrow multiply(X, b)$;
    end
end
return $X$;
```

Attacks

Page Table Based Attacks

- Basic page-fault attack
  - AEX on fault ⇨ base address of faulting page revealed
  - Cause intentional page faults ⇨ obtain page level trace
  - Monitor a specific page

```
void CountLogin( GENDER s ) {
    if ( s == MALE ) {
        gMaleCount ++;
    } else {
        gFemaleCount ++;
    }
}
```



```
void analyze_genome( dna_sample_t *sample,
                     report_t *report )
{
    if ( detect_disorder_xyz( sample ) )
        add_description_xyz( report );
}
```

```
char* WelcomeMessage( GENDER s ) {
    char *mesg;

    // GENDER is an enum of MALE and FEMALE
    if ( s == MALE ) {
        mesg = WelcomeMessageForMale();
    } else { // FEMALE
        mesg = WelcomeMessageForFemale();
    }
    return mesg;
}
```

Attacks

Page Table Based Attacks

- Problem: page-faults cause a high overhead => attack detection
  - Solution: Use page-table access and dirty bits
- Problem: TLB caches PTEs and flags are not updated in PT
  - Solution: Flush TLB using an IPI
- Problem: Lots of IPIs
  - Solution: Time difference between two repeatedly accessed pages ⇨ one interrupt
- Problem: Still interrupts!
  - Solution: TLB is shared between two hyperthreads on same core ⇨ invalidate TLB entries ⇨ no IPIs

Attacks

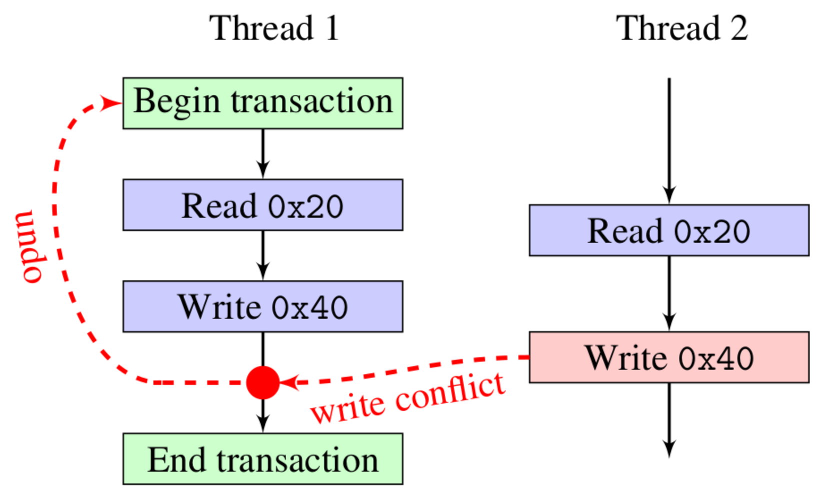
Other Attacks

- Flush+Reload
  - Flush a memory line and wait for victim to access it
- Flush+Flush
  - Flush a cache line using clflush instruction
- DRAMA
  - Caching disabled
  - Allocate two memory lines with different virtual addresses but same physical address
  - Regularly access one of the memory lines
- Cache-DRAM
  - Prime+Probe + DRAMA

Defenses

Defense against cache-based attacks

- Basic approach: Pin code/data to cache
- Using Intel TSX
  - Introduced with Haswell
  - Critical sections management
  - Monitoring serialization
  - Restricted Transactional Memory
    - New Instruction set interface
- Cloak
  - Preload code/data
  - Execute algorithm inside transaction

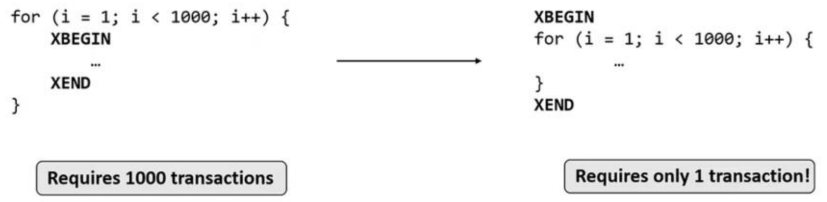


```
xbegin();
preload all sensitive data/code;
run algorithm;
xend();
```

Defenses

Defense against page table based attacks

- T-SGX: run code inside transaction => page fault => attack detection => program termination
- Problem: Frequent timer interrupt
  - Program take too long => exception => never terminate
- Problem: TSX buffers all memory changes inside cache
  - single transaction, lots of accesses => cache conflict => transaction abort
- Break program into tiny blocks
  - Problem: transaction setup has cost => performance
- Merge continuous blocks
- Problem: boundaries leak
  - Solution: springboard

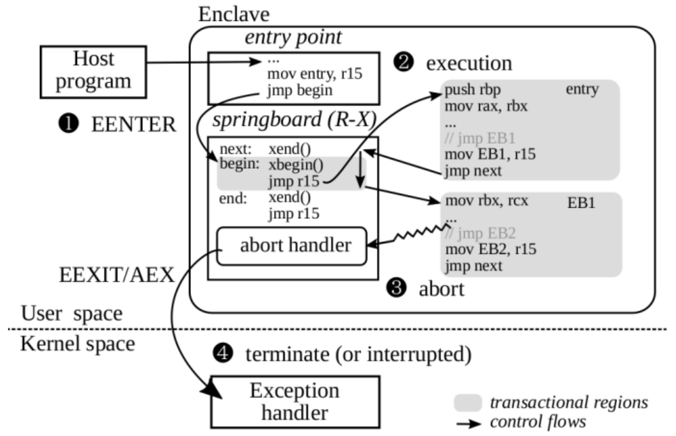


```
for (i = 1; i < 1000; i++) {
    XBEGIN
      …
    XEND
}
```

Requires 1000 transactions

```
XBEGIN
for (i = 1; i < 1000; i++) {
      …
}
XEND
```

Requires only 1 transaction!



Defenses

Other defense methods

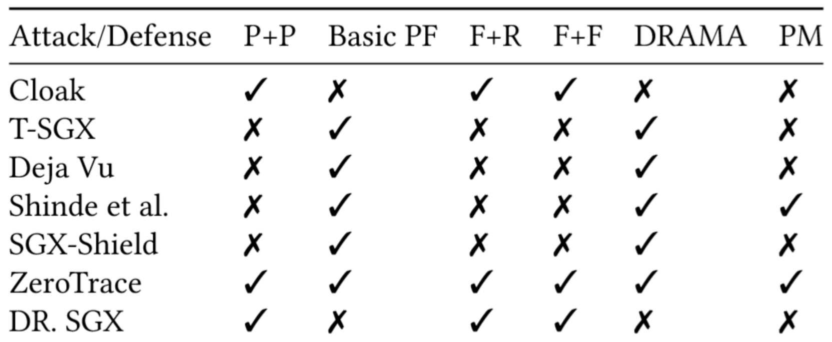
- Déjà vu
  - Reference clock thread inside a transaction
- Shinde et al.
  - Deterministic page access profile
  - Introducing fake acceses
- SGX-Shield
  - ASLR
- ZeroTrace
  - Use of ORAM + SGX
- Dr. SGX
  - Data location randomization

Analysis

- Cloak
  - Makes cache-pinning possible to some extent
  - Execution time can leak
  - Aborts do not cancel concurrent memory accesses
  - Execution behavior and branch prediction uncertainties
- T-SGX
  - Attacks based on monitoring flags are possible
- Déjà vu
  - Page monitoring attacks still effective
  - Only works on AEX based attacks
- Bottom line
  - Transaction based defenses usually come with high overload and low utilization, and need isolation

Analysis

- Other methods
  - Attack detection methods ⇨ Unreliable
  - Shuffling memory ⇨ expensive
  - ORAM (make addresses input-independent) ⇨ expensive
- The perfect solution?
  - Remove all branches and conditional code
  - Pin data/code to cache and TLB
  - CAT+SGX
  - Non-shared non-cached secure memory element
- Most of defenses either work on page table based attacks or cache-based attacks
- ORAM performance optimization
- Page monitoring attacks



Attack/Defense	P+P	Basic PF	F+R	F+F	DRAMA	PM
Cloak	✓	✗	✓	✓	✗	✗
T-SGX	✗	✓	✗	✗	✓	✗
Deja Vu	✗	✓	✗	✗	✓	✗
Shinde et al.	✗	✓	✗	✗	✓	✓
SGX-Shield	✗	✓	✗	✗	✓	✗
ZeroTrace	✓	✓	✓	✓	✓	✓
DR. SGX	✓	✗	✓	✓	✗	✗

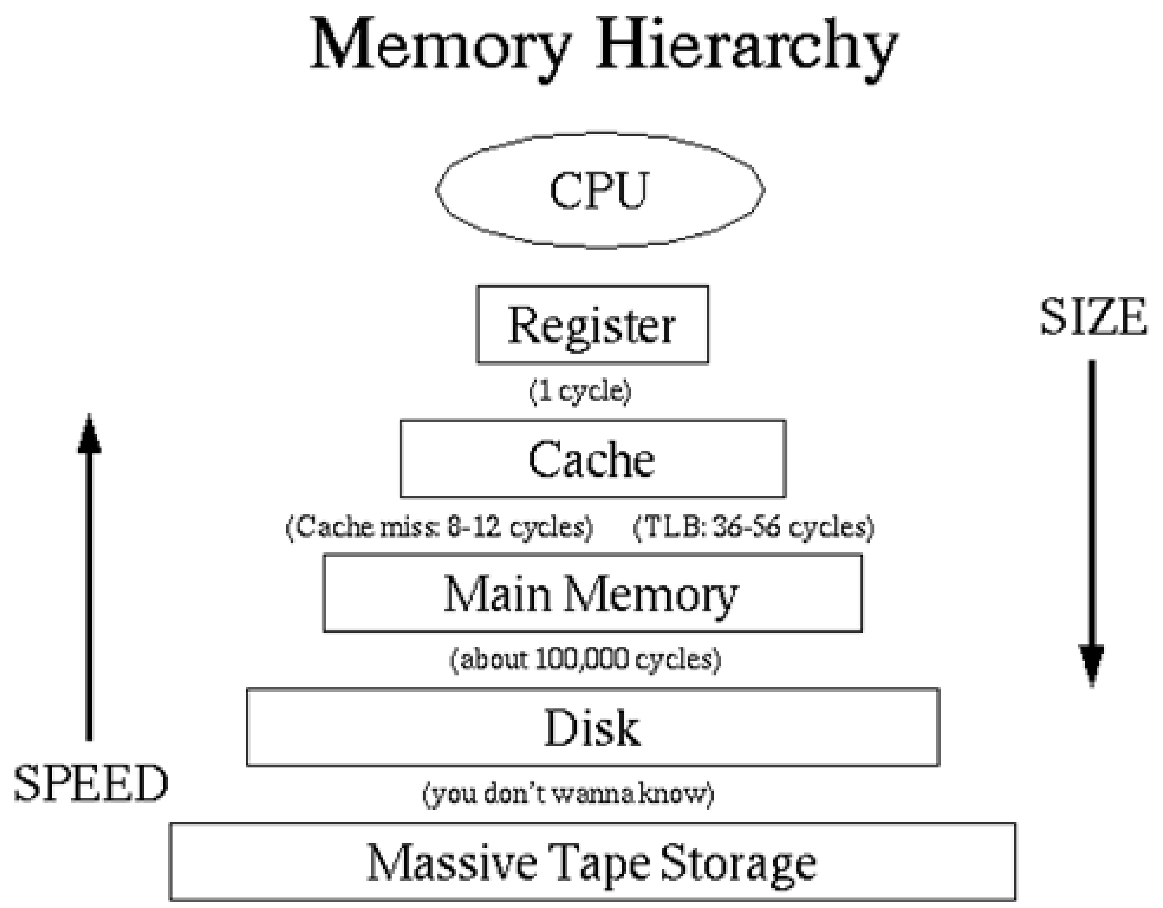
Conclusion

- We studied multiple attacks and defenses
- Still no robust defense
- Attack are emerging
- Open to research

References

[1] Ferdinand Brasser, Srdjan Capkun, Alexandra Dmitrienko, Tommaso Frassetto, Kari Kostiainen, Urs Mu¨ller, and Ahmad-Reza Sadeghi. 2017. DR. SGX: Hardening SGX Enclaves against Cache Attacks with Data Location Randomization. arXiv preprint arXiv:1709.09917 (2017).
[2] Ferdinand Brasser, Urs Mu¨ller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and AhmadReza Sadeghi. 2017. Software grand exposure: SGX cache attacks are practical. arXiv preprint arXiv:1702.07521 (2017), 33.
[3] Sanchuan Chen, Xiaokuan Zhang, Michael K Reiter, and Yinqian Zhang. 2017. Detecting privileged side-channel attacks in shielded execution with D´ej´a Vu. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. ACM, 7–18.
[4] Oded Goldreich. 1987. Towards a theory of software protection and simulation by oblivious RAMs. In Proceedings of the nineteenth annual ACM symposium on Theory of computing. ACM, 182–194.
[5] Oded Goldreich and Rafail Ostrovsky. 1996. Software protection and simulation on oblivious RAMs. Journal of the ACM (JACM) 43, 3 (1996), 431–473.
[6] Daniel Gruss, Julian Lettner, Felix Schuster, Olya Ohrimenko, Istvan Haller, and Manuel Costa. 2017. Strong and efficient cache side-channel protection using hardware transactional memory. In USENIX Security Symposium.
[7] Daniel Gruss, Cl´ementine Maurice, Klaus Wagner, and Stefan Mangard. 2016. Flush+ Flush: a fast and stealthy cache attack. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 279–299.
[8] Mehmet Kayaalp, Nael Abu-Ghazaleh, Dmitry Ponomarev, and Aamer Jaleel. 2016. A high-resolution side-channel attack on last-level cache. In Proceedings of the 53rd Annual Design Automation Conference. ACM, 72.
[9] Rafail Ostrovsky. 1990. Efficient computation on oblivious RAMs. In Proceedings of the twenty-second annual ACM symposium on Theory of computing. ACM, 514–523.
[10] Dag Arne Osvik, Adi Shamir, and Eran Tromer. 2006. Cache attacks and countermeasures: the case of AES. In Cryptographers' Track at the RSA Conference. Springer, 1–20.
[11] Peter Pessl, Daniel Gruss, Cl´ementine Maurice, Michael Schwarz, and Stefan Mangard. 2016. DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks.. In USENIX Security Symposium. 565–581.
[12] Sajin Sasy, Sergey Gorbunov, and Christopher Fletcher. 2017. ZeroTrace: Oblivious memory primitives from Intel SGX. In Symposium on Network and Distributed System Security (NDSS).
[13] Jaebaek Seo, Byounyoung Lee, Seongmin Kim, Ming-Wei Shih, Insik Shin, Dongsu Han, and Taesoo Kim. 2017. SGX-Shield: Enabling address space layout randomization for SGX programs. In Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA.
[14] Ming-Wei Shih, Sangho Lee, Taesoo Kim, and Marcus Peinado. 2017. T-SGX: Eradicating controlledchannel attacks against enclave programs. In Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA.
[15] Shweta Shinde, Zheng Leong Chua, Viswesh Narayanan, and Prateek Saxena. 2015. Preventing your faults from telling your secrets: Defenses against pigeonhole attacks. arXiv preprint arXiv:1506.04832 (2015).
[16] Eran Tromer, Dag Arne Osvik, and Adi Shamir. 2010. Efficient cache attacks on AES, and countermeasures. Journal of Cryptology 23, 1 (2010), 37–71.
[17] Jo Van Bulck, Nico Weichbrodt, Ru¨diger Kapitza, Frank Piessens, and Raoul Strackx. 2017. Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In Proceedings of the 26th USENIX Security Symposium. USENIX Association.
[18] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, Xiaofeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A. Gunter. 2017. Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX. 2421–2434.
[19] Yuval Yarom and Katrina Falkner. 2014. FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack.. In USENIX Security Symposium

Introduction



Defenses

- SGX-Shield
  - Address space layout randomization
- ORAM+SGX
- Data shuffling

Introduction

Intel TSX

- Introduced with Haswell
- Critical sections management
- Monitoring serialization
- Restricted Transactional Memory
  - New Instruction set interface

Attacks

Points of attack

- TLB: shared between enclave and non-enclave programs ⇨ with hyperthreading enabled it creates side-channels
- Page faults: visible to OS
- TLB flushes: on context switch
- Page table entry flags
- Enclave memory address beginning and offset are known
- Enclave exits (AEX)

Analysis

- SGX-Shield
  - No live randomization ⇨ observe and monitor random patterns
- Shinde et al.
  - Cache and TLB based attacks possible