Minefield: A Software-only Protection for SGX Enclaves against DVFS Attacks

USENIX Security 2022

12th August 2022

ARTIFACT EVALUATED
usenix ASSOCIATION
AVAILABLE

ARTIFACT EVALUATED
usenix ASSOCIATION
FUNCTIONAL

ARTIFACT EVALUATED
usenix ASSOCIATION
REPRODUCED

Andreas Kogler
Graz University of Technology

Daniel Gruss
Graz University of Technology

Michael Schwarz
CISPA Helmholtz Center for Information Security

Motivation

- **Undervolting**
  - Saves energy
  - Increases performance

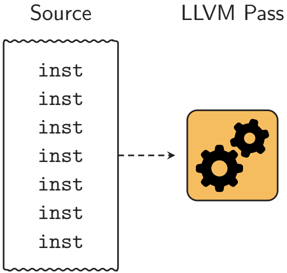
Motivation

- **Undervolting**
  - Saves energy
  - Increases performance
- **Attacks**
  - **SW**: Plundervolt, V0LTpwn, VoltJockey [3, 2, 4, 5]
  - **HW**: VoltPillager [1]

Motivation

- **Undervolting**
  - Saves energy
  - Increases performance
- **Attacks**
  - **SW**: Plundervolt, V0LTpwn, VoltJockey [3, 2, 4, 5]
  - **HW**: VoltPillager [1]
- **Intel's Mitigations**
  - **SW**: SGX $\oplus$ UV
  - **HW**: Fully Integrated Voltage Regulators (FIVRs)

High Level Idea

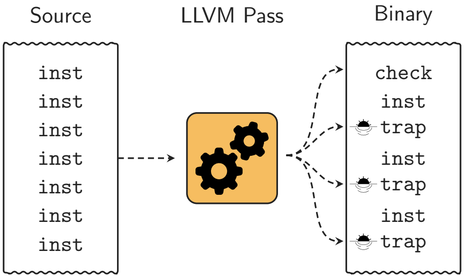


Source

LLVM Pass

```
inst
inst
inst
inst
inst
inst
inst
```

High Level Idea



Source

```
inst
inst
inst
inst
inst
inst
inst
```

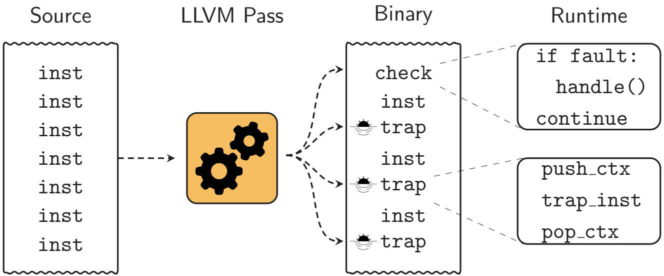
LLVM Pass

Binary

```
check
inst
trap
inst
trap
inst
trap
```

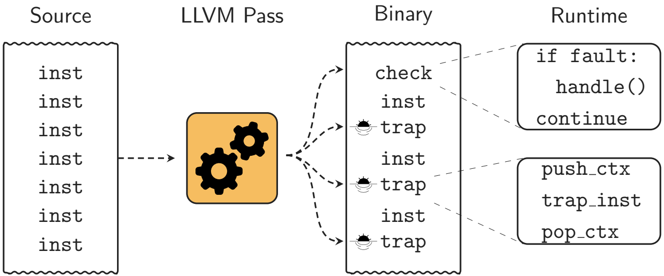
- **Place** trap instructions

High Level Idea



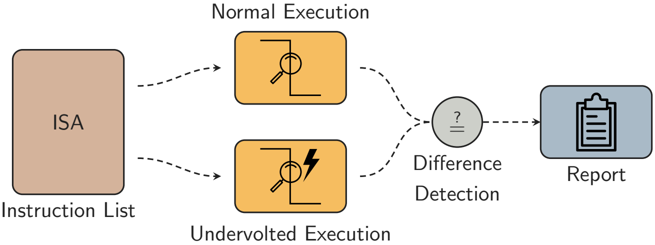
- **Place** trap instructions
- **Check** the results

High Level Idea



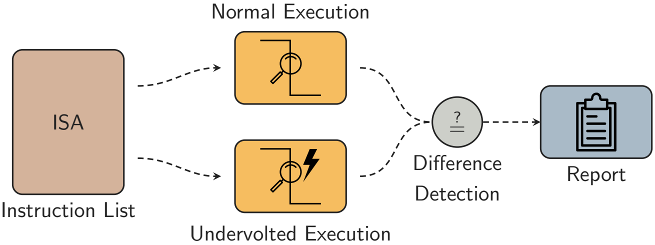
- **Place** trap instructions
- **Check** the results
- **Abort** if mismatch

Implementation - Trap Instruction



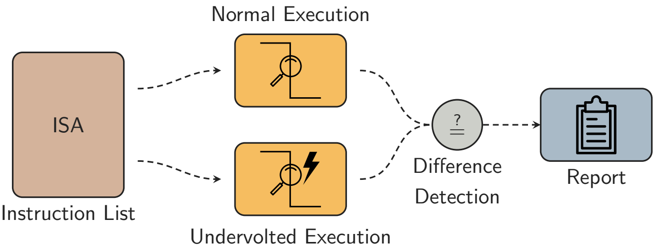
- **Analyze** instructions
  - CPUs
  - Cores
  - Voltages
  - Frequencies

Implementation - Trap Instruction



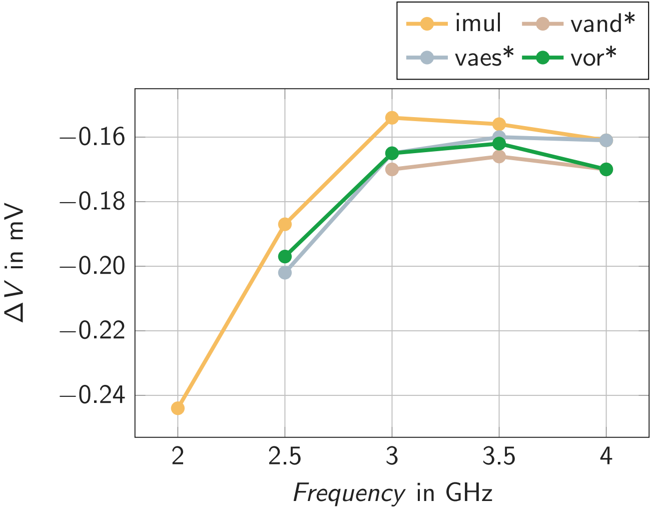
- **Analyze** instructions
  - CPUs
  - Cores
  - Voltages
  - Frequencies
- **Detect** faults

Implementation - Trap Instruction



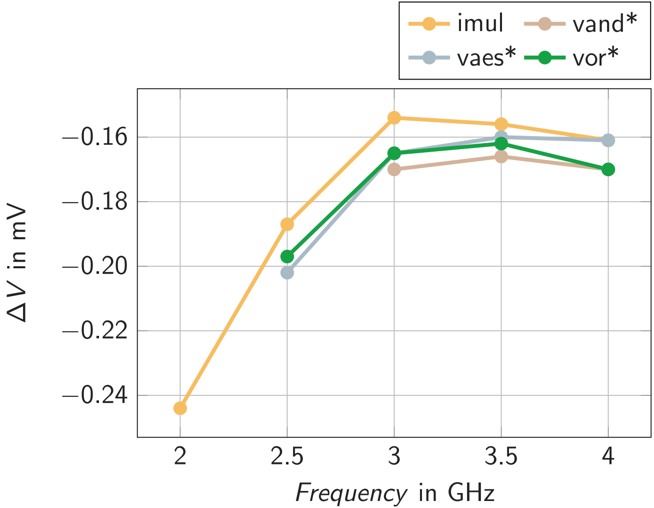
- **Analyze** instructions
  - CPUs
  - Cores
  - Voltages
  - Frequencies
- **Detect** faults
- **Restart** remote via PDU

Implementation - Trap Instruction



- **1258** Instructions
- **5/26** CPUs/Cores
- **71** Faultable

Implementation - Trap Instruction



- **1258** Instructions
- **5/26** CPUs/Cores
- **71** Faultable

✓ **92.1%** → `imul`

✓ **6.4%** → `vorpd`

✓ **1.5%** → `aesenc`

Implementation - Integration

```
imul $11, input(%rip), %rax

cmp  %rax, limit(%rip)



ja   .L1
```

- **LLVM** compiler extension
  - Inserts checks
  - Inserts alternating traps
  - Saves context

Implementation - Integration

```
cmp   %r12, %r13
jne  __abort
imul __factor(%rip), %r12
```

```
imul $11, input(%rip), %rax
```

```
cmp  %rax, limit(%rip)
```

```
ja   .L1
```

- **LLVM** compiler extension
  - Inserts checks
  - Inserts alternating traps
  - Saves context

Implementation - Integration

```
cmp    %r12, %r13
jne   __abort
imul __factor(%rip), %r12
imul $11, input(%rip), %rax
imul __factor(%rip), %r13
cmp  %rax, limit(%rip)





ja   .L1
```

- **LLVM** compiler extension
    - Inserts checks
    - Inserts alternating traps
    - Saves context

Implementation - Integration

```
cmp   %r12, %r13
jne  __abort
imul __factor(%rip), %r12
imul $11, input(%rip), %rax
imul __factor(%rip), %r13
cmp  %rax, limit(%rip)
pushf
imul __factor(%rip), %r13
imul __factor(%rip), %r12
popf
ja   .L1
```

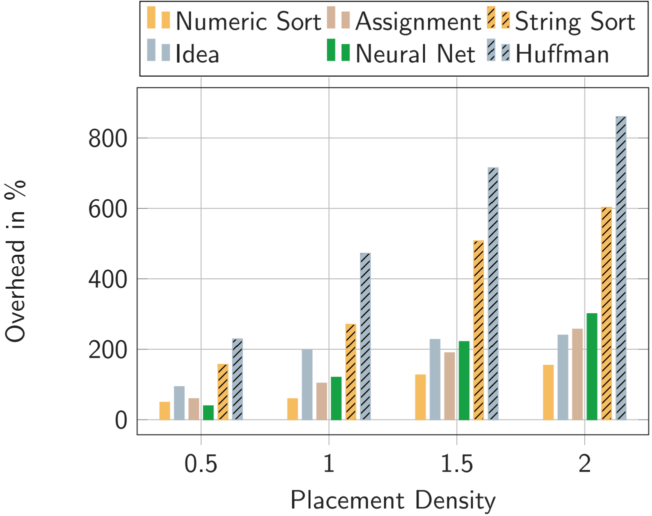
- **LLVM** compiler extension
  - Inserts checks
  - Inserts alternating traps
  - Saves context

Implementation - Integration

```
cmp   %r12, %r13
jne  __abort
imul __factor(%rip), %r12
imul $11, input(%rip), %rax
imul __factor(%rip), %r13
cmp  %rax, limit(%rip)
pushf
imul __factor(%rip), %r13
imul __factor(%rip), %r12
popf
ja   .L1
```

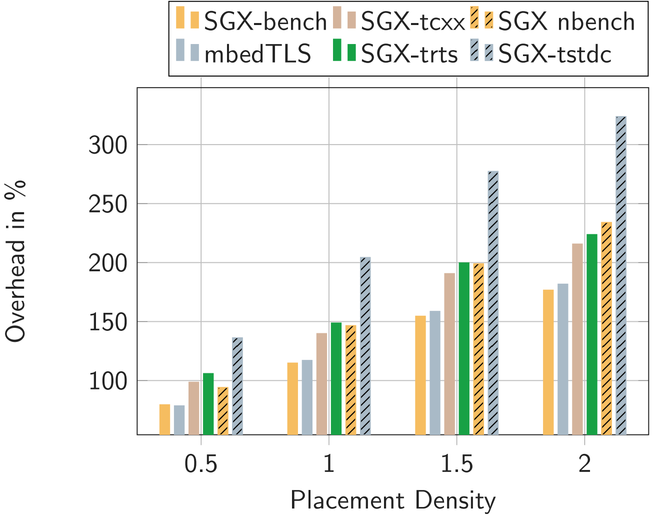
- **LLVM** compiler extension
  - Inserts checks
  - Inserts alternating traps
  - Saves context
- **SGX-SDK** support

Evaluation - Performance



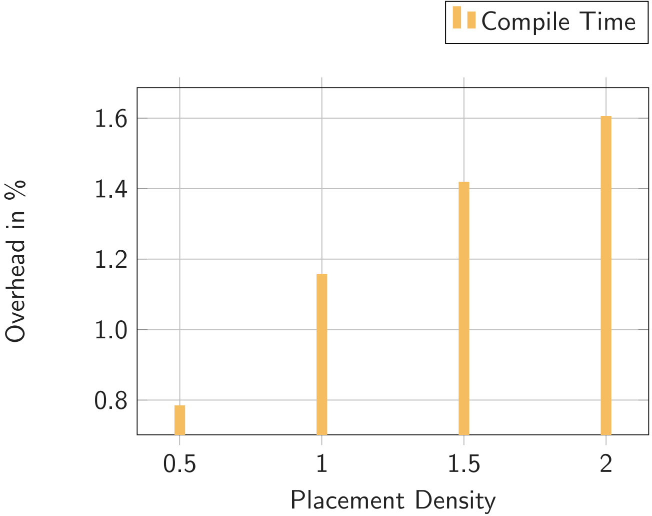
- **Runtime**

Evaluation - Performance



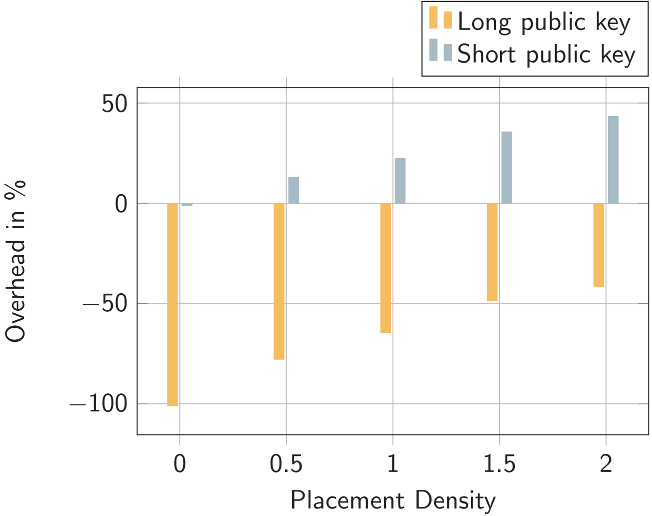
- **Runtime**
- **Size**

Evaluation - Performance



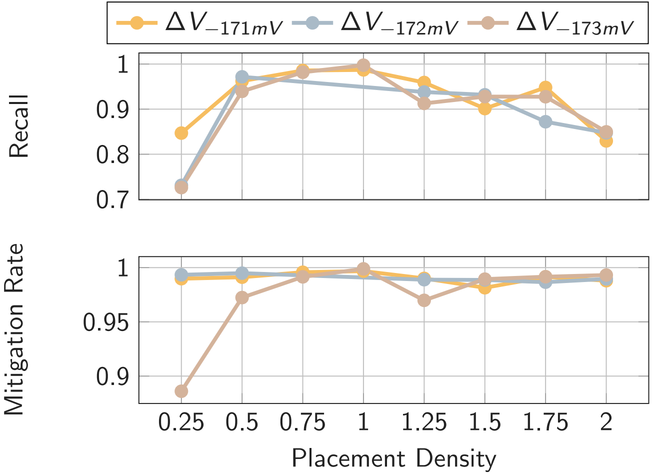
- **Runtime**
- **Size**
- **Compile time**

Evaluation - Performance



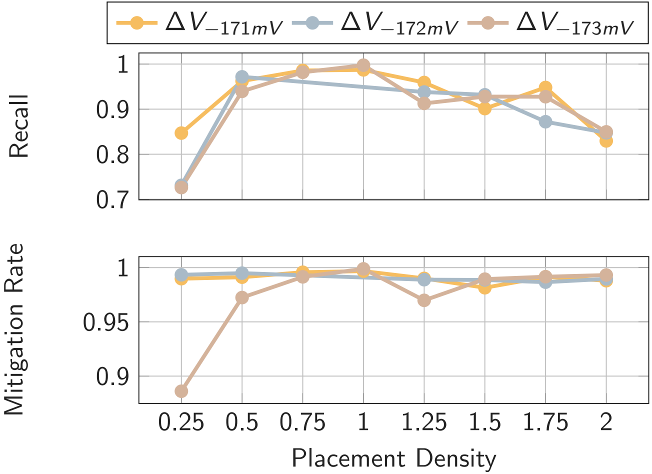
- **Runtime**
- **Size**
- **Compile time**
- **MbedTLS**

Evaluation - Detection



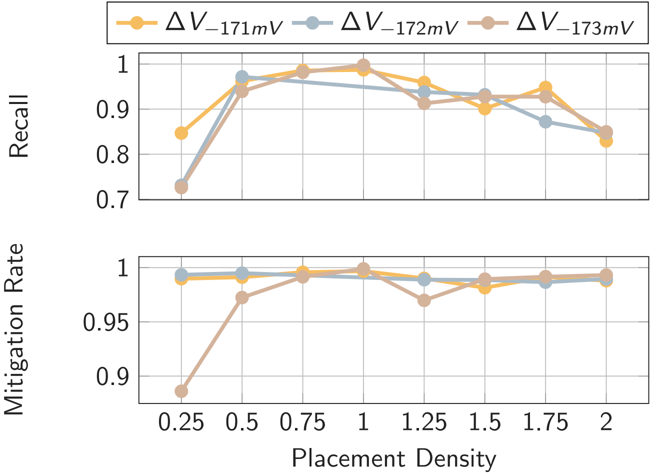
- **Plundervolt** PoC
  - Worst case

Evaluation - Detection



- **Plundervolt** PoC
  - Worst case
- Recall

Evaluation - Detection



- **Plundervolt** PoC
  - Worst case
- Recall
- Mitigation rate

Conclusion

- **Open Source** `https://github.com/IAIK/minefield`

Conclusion

- **Open Source** `https://github.com/IAIK/minefield`
- **Passed** artifact evaluation

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Conclusion

- **Open Source** `https://github.com/IAIK/minefield`
- **Passed** artifact evaluation

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

- **More** details
  - Exact faulting points
  - Faulting masks
  - Compiler details
  - ...

Conclusion

- **Open Source** `https://github.com/IAIK/minefield`
- **Passed** artifact evaluation

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

- **More** details
  - Exact faulting poin
  - Faulting mask
  - Compiler
  - ...

Read the Paper

References i

- Zitai Chen, Georgios Vasilakis, Kit Murdock, Edward Dean, David Oswald, and Flavio D Garcia. VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface. In: USENIX Security Symposium. 2020.
- Zijo Kenjar, Tommaso Frassetto, David Gens, Michael Franz, and Ahmad-Reza Sadeghi. V0LTpwn: Attacking x86 Processor Integrity from Software. In: USENIX Security Symposium. 2020.
- Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. Plundervolt: Software-based Fault Injection Attacks against Intel SGX. In: S&P. 2020.
- Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. VoltJockey: Breaching TrustZone by Software-Controlled Voltage Manipulation over Multi-core Frequencies. In: CCS. 2019.
- Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. VoltJockey: Breaking SGX by Software-Controlled Voltage-Induced Hardware Faults. In: AsianHOST. 2019.

Additional Slides