Want strong isolation?
Just reset your processor.

How we can build more secure systems by applying
the age-old wisdom of “turning it off and on again”

Anish Athalye, Adam Belay, Frans Kaashoek, Robert Morris, Nickolai Zeldovich
Security devices are increasingly common

Smartphone apps

Custom hardware
They are getting better

- SMS 2FA
- Smartphone TOTP
- Hardware TOTP
- Smartphone U2F
- Hardware U2F
- Hardware Cryptocurrency wallet

Smaller TCB
Paradigm shift

2FA: More secure login on PC

Transaction approval, Removes PC from TCB
Can we make the PC secure instead?

• Endless bugs:
  • Application bugs
  • OS bugs (kernels > 20M LOC)
  • Micro-architectural CPU bugs (Spectre, Meltdown, Foreshadow, Zombieload)
  • Hardware bugs (Rowhammer, RAMBleed)
Transaction approval on simple devices
Remove PC from TCB

Untrusted PC

TX

Sign(TX)

Trusted hardware
(has private key)
Remove PC from TCB
Not just cryptocurrencies

Before: confirm on PC

After: must confirm on device, which signs transaction
Not just cryptocurrencies

Web Authentication:
An API for accessing Public Key Credentials
Level 1
W3C Recommendation, 4 March 2019

§ 10.2. Simple Transaction Authorization Extension (txAuthSimple)

This extension allows for a simple form of transaction authorization. A Relying Party can specify a prompt string, intended for display on a trusted device on the authenticator.
Transaction approval everywhere!
Transaction approval everywhere!
Sharing results in isolation bugs

• Some past wallet bugs
  • Bad argument validation in syscalls
  • Bad configuration of MPU

[Riscure @ Black Hat 2018; Ledger Blog; Trezor Blog]
Security through physical separation
Simulating physical separation
Reset-based design

Syscalls:
- `exit()`
- `exit_state(state)`

Application core
- Runs third-party code; has no persistent state

Management core
- Manages persistent state; never runs third-party code
What needs to be reset?

![Diagram showing components connected to CPU and RAM]

- Buttons
- Display
- USB
- UART
- Flash
- Application core
- Management core
Purging state in a CPU, attempt #1

[Diagram of a CPU showing connections and cut power highlighted]
Purging state in a CPU, attempt #2

reset!
What does reset mean?

3.3 Reset

Upon reset, a hart’s privilege mode is set to M. The mstatus fields MIE and MPRV are reset to 0, and the VM field is reset to Mbare. The pc is set to an implementation-defined reset vector. The mcause register is set to a value indicating the cause of the reset. All other hart state is undefined.
Purging state in a CPU, attempt #3

```
reset!

mov r0, #0
mov r1, #0
mov r2, #0
...
```
Architectural and micro-architectural state

Examples of architectural state vs micro-architectural state
Minimizing complexity

Simpler processors have less micro-architectural state
Purging state in a CPU, final attempt

These instructions affect micro-architectural state

\[
\begin{align*}
& \text{mov } r0, \#0 \\
& \text{mov } r1, \#0 \\
& \text{mov } r2, \#0 \\
& \text{+ ...} \\
& \text{// do things that end up} \\
& \text{// clearing internal state} \\
& \text{...}
\end{align*}
\]
How do we know that reset is correct?

Arbitrary state \[\rightarrow\] Reset / purge \[\rightarrow\] Purged state
How do we know that reset is correct?

State (secret = 0) → Reset / purge → (Same) purged state

State (secret = 1)
Tool: Satisfiability Modulo Theories (SMT)

\[(x \text{ AND } y) \text{ OR } (\text{NOT } z)\]  
\[
\Rightarrow \text{ SAT: } \{x = \text{False}, y = \text{False}, z = \text{False}\}
\]

\[(x \text{ AND } y) \text{ AND } ((\text{NOT } x) \text{ AND } z)\]  
\[
\Rightarrow \text{ UNSAT}
\]

SAT solvers

SMT: SAT on steroids

\[
x: \text{Int}, y: \text{Int}
\]  
\[
x + y < 1 \text{ AND } x + 1 = 3 \text{ AND } y > 0
\]  
\[
\Rightarrow \text{ UNSAT}
\]

\[
x: \text{BitVec}(8)
\]  
\[
x > 0 \text{ AND } x + 1 < 0
\]  
\[
\Rightarrow \text{ SAT: } \{x = 127\}\]
SMT solvers as theorem provers

Theorem: \( \forall x, P(x) \)

Mechanical translation to SMT formula: strip foralls, negate proposition

\[ \neg (P(x)) \]

SAT \( \Rightarrow \) theorem is false

A counterexample to our theorem: an \( x \) where \( \neg (P(x)) \)

UNSAT \( \Rightarrow \) theorem is proven

A proof that our theorem holds: because there is no \( x \) that makes \( \neg (P(x)) \) true, \( P(x) \) must hold for all \( x \)
SMT solvers as theorem provers

Theorem: forall x y : Real, min(x, y) <= (x + y)/2 <= max(x, y)

NOT [ min(x, y) <= (x + y)/2 <= max(x, y) ]

```python
>>> from z3 import *
>>> x = Real('x'); y = Real('y'); s = Solver()
>>> Min = z3.If(x < y, x, y)
>>> Max = z3.If(x < y, y, x)
>>> avg = (x + y)/2
>>> theorem = And(Min <= avg, avg <= Max)
>>> s.add(Not(theorem))
>>> s.check()
unsat
```
Reset: theorem statement

forall s s': CPU state,  
purge(s) = purge(s')
Combinatorial circuits

Full adder

S

A
B
C_{in}

C_{out}

T_c

Carry-block
Combinatorial circuits

module full_adder(input a, input b, input c_in, output s, output c_out);

    wire xor_in;
    assign xor_in = a ^ b;
    assign s = xor_in ^ c_in;
    wire and_in;
    assign and_in = a & b;
    assign c_out = (xor_in & c_in) | and_in;

endmodule
Combinatorial circuits

```python
def full_adder(a, b, c_in):
    xor_in = Xor(a, b)
    and_in = And(a, b)
    s = Xor(xor_in, c_in)
    c_out = Or(And(xor_in, c_in), and_in)
    return (s, c_out)
```

```python
input1 = (Bool('a1'), Bool('b1'), Bool('c_in1'))
input2 = (Bool('a2'), Bool('b2'), Bool('c_in2'))

solver = Solver()
solver.add(input1[0] == input2[1])
solver.add(input1[1] == input2[0])
solver.add(input1[2] == input2[2])

def all_eq(els, e2s):
    return And(*[el == e2 for el, e2 in zip(els, e2s)])

solver.add(Not(all_eq(*input1), full_adder(*input1), full_adder(*input2)))
solver.check() # => unsat
```

Preconditions:
a/b are swapped
c_in is the same

Negated theorem statement:
result of adder is different

Verified!

Prove: \( a + b = b + a \) (when c_in is the same)
module counter(input clk, input reset, input en, output [7:0] count);

    reg [7:0] count;

    always @(posedge clk) begin
        if (reset) begin
            count <= 0;
        end
        else if (en) begin
            count <= count + 1;
        end
    end
endmodule
Proof (attempt): as long as it’s not reset, count doesn’t decrease

\[ s_1 = \text{Function('s1', counter_s)}() \]
\[ s_2 = \text{Function('s2', counter_s)}() \]

\[ s = \text{Solver()} \]
\[ s.\text{add(counter_t(s1, s2))} \]
\[ s.\text{add(Not(reset(s1)))} \]
\[ s.\text{add(count(s2) < count(s1))} \]

Preconditions: s1 steps to s2, counter is not reset

Negated theorem statement: count decreases

\[ s.\text{check()} \] # => sat
\[ m = s.\text{model()} \]
\[ m.\text{evaluate(s1)} \]

# => counter(False, True, 127)

Concrete counterexample!
Converting CPU implementation to SMT

Verilog implementation:
Gate-level design of CPU

Python / Z3 SMT model:
Describes 1 cycle of CPU execution
Symbolic simulation of the CPU
Proving reset correct

Any possible initial states

Must converge to the same final state

s₀ → reset → s₁ → step → s₂ → step → s₃ → step → s₄ → step → ... → sₙ

s₀' → reset → s₁' → step → s₂' → step → s₃' → step → s₄' → step → ... → sₙ'

sₙ → sₙ'

sₙ → sₙ'
Interactive development of reset code

“Concrete counterexample: state \(<X>\) is not cleared”
Demo (reset verification)
cycle 102
  substrate cpuregs_30 is not provably reset
    (step took 2.7s)

cycle 103
  substrate cpuregs_30 is not provably reset
    (step took 2.7s)

cycle 104
  substrate mem_wdata is not provably reset
    (step took 1.7s)

cycle 105
  substrate mem_wdata is not provably reset
    (step took 1.8s)

cycle 106
  substrate mem_wdata is not provably reset
    (step took 1.7s)

cycle 107
  substrate mem_wdata is not provably reset
    (step took 1.7s)

cycle 108
  ^C  substrate mem_wdata is not provably reset
    (step took 4.4s)

cycle 109
  substrate mem_wdata is not provably reset
    (step took 9.4s)

cycle 110
  ^CException ignored in: <function AstRef.__del__ at 0x104a26b70>

Traceback (most recent call last):
  File "/usr/local/lib/python3.7/site-packages/z3/z3.py", line 305, in __del__
    Z3_dec_ref(self.ctx.ref(), self.as_ast)
  File "/usr/local/lib/python3.7/site-packages/z3/z3.py", line 504, in as_ast
    return Z3_sort_to_ast(self.ctx_ref(), self.ast)
  File "/usr/local/lib/python3.7/site-packages/z3/z3core.py", line 2475, in Z3_sort_to_ast
    r = elems.f(a0, a1)
ctypes.ArgumentError: argument 2: <class 'KeyboardInterrupt'>:
  ^C  substrate mem_wdata is not provably reset
    (step took 2.2s)

cycle 111
  ^C
Demo (hardware)
Welcome to NamedManager, a PHP web-based DNS management interface for managing DNS zones.
Conclusion

"Turning it off and on again" (with a little more details) can be used as a primitive for achieving isolation.

• **Users**: Start using transaction authorization devices

• **Developers**:
  • Support factoring out approval decisions (see WebAuthn)
  • Use verification as a tool to improve security

Code: [git.io/shiva](https://git.io/shiva)