CLKSCREW

Exposing the Perils of Security-Oblivious Energy Management

Adrian Tang, Simha Sethumadhavan and Salvatore Stolfo
USENIX Security 2017

Presented by
Alokparna Bandyopadhyay
Fall 2018, Wayne State University

Overview

- Introduction
- DVFS
- The CLKSCREW Attack
- Attacking the ARM Trustzone
- Discussion
- Conclusion

Overview

- Introduction
- DVFS
- The CLKSCREW Attack
- Attacking the ARM Trustzone
- Discussion
- Conclusion

Introduction

- Energy management is crucial for modern devices
  - Reduce cost, increase battery life, enhance portability
  - Requires hardware-software interoperability to minimize energy consumption and maximize performance

- Security of modern systems is at a risk due to the lack of secure energy management
  - Security vulnerabilities attract malicious attackers

Introduction

- Attackers exploit *software interfaces* to Energy Management hardware
  - Does not require physical access to target devices
  - Does not need separate equipment
  - Bypass hardware security defense mechanisms

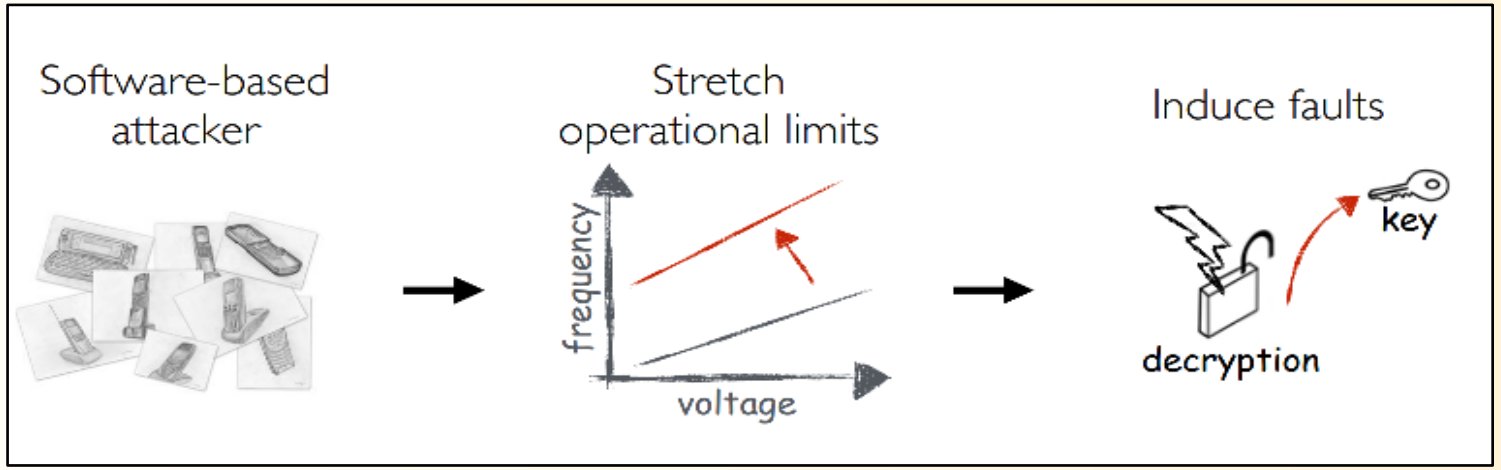


Image source: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang

Introducing the CLKSCREW Attack

- New type of software attack that exploits energy management
- Practical security attack which subverts the hardware-enforced isolation in ARM Trustzone
  - Extract the secret AES keys embedded within Trustzone
  - Load self-signed code into Trustzone
- Impacts hundreds of millions of devices which use the ARM Trustzone for secure computing
- Lessons for future energy management designs to be security-conscious

Overview

- Introduction
- DVFS
- The CLKSCREW Attack
- Attacking the ARM Trustzone
- Discussion
- Conclusion

DVFS – Dynamic Voltage & Frequency Scaling

- Energy consumption ≈ Power x Time

  Power ∝ Frequency x Voltage
- DVFS : an energy management tool
  - Trades off between processing speed and energy savings
  - Focuses on efficiently optimizing both frequency and voltage based on runtime task demands
  - Requires *hardware and software components* across layers in the system stack
    - *Hardware support* : Voltage regulator & Frequency Regulator
    - *Software support* : Vendor specific regulator driver & OS-level power governor
    - Operating frequency and voltage can be configured via memory-mapped registers from software

Hardware & Software Support for DVFS

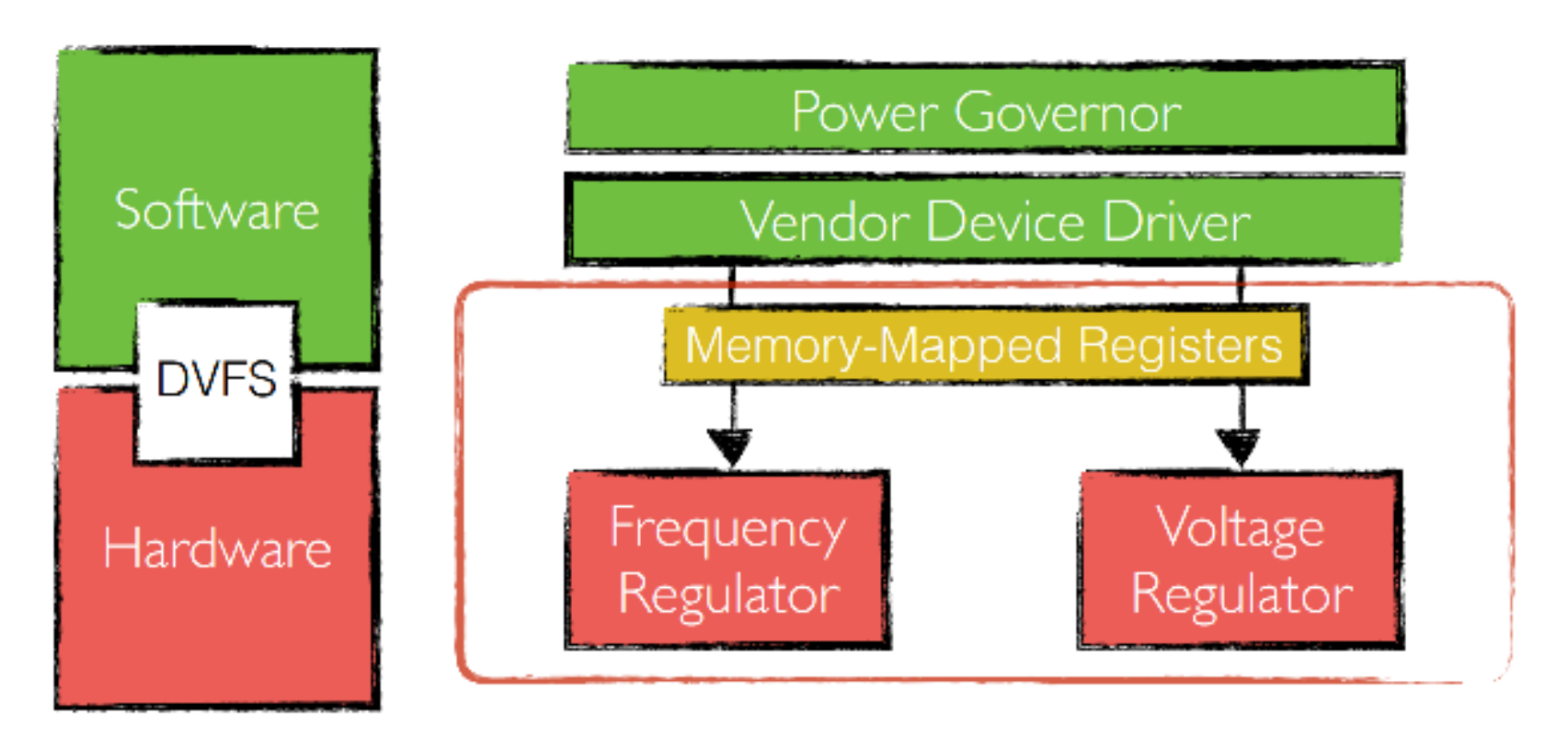
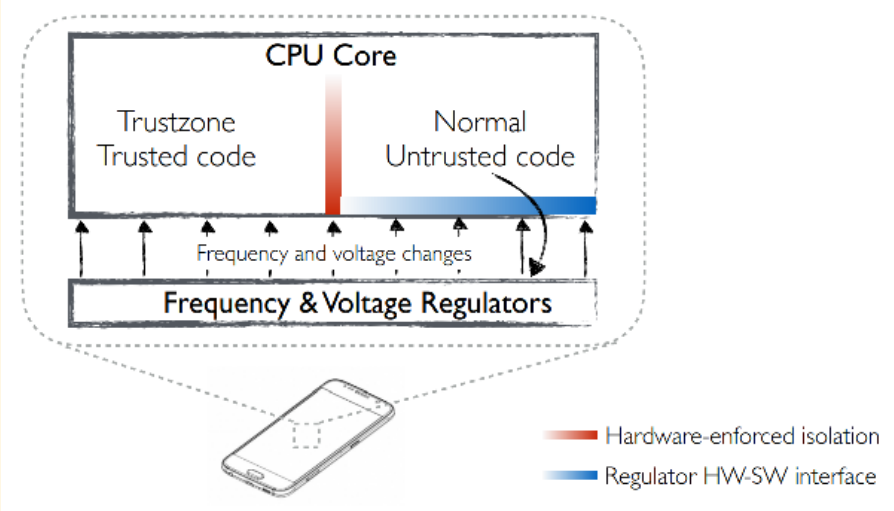


Image source: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang

DVFS Security Vulnerability

- Hardware regulators do not impose any kind of safeguard limits to frequency/voltage changes
  - unfettered access to hardware regulators increases the risk of over exceeding frequency (*overclocking)* and under supplying voltage (*undervolting)* → Timing Fault!
- DVFS operates across the security boundaries of the ARM Trustzone
  - Remains unaware of the hardware enforced Trustzone isolation



Overview

- Introduction
- DVFS
- The CLKSCREW Attack
- Attacking the ARM Trustzone
- Discussion
- Conclusion

The CLKSCREW Attack

Objective

- Attack the Trustzone code execution using software-only control of the regulators
  - Take advantage of the DVFS security vulnerability
  - Stretch frequency and voltage beyond the operating limits of digital circuits to create erroneous computation → Timing Attack

Implementation platform

- CLKSCREW is implemented on a commodity ARMv7 phone, Nexus 6 to demonstrate the practicality of the security attack

Timing Attack

- For the overall circuit to propagate input → output, the minimum required clock cycle period, T_{clk}, is bounded by the following timing constraint:

$$T_{clk} \geq T_{FF} + T_{max\_path} + T_{setup} + K$$

- Violation of timing constraint:
  - Over-raising the frequency reduces the timing constraint T_{clk} – less time for data to propagate
  - Under-supplying the voltage increases the overall circuit propagation time, increases T_{max_path}
- Due to timing constraint violation during two consecutive clock signals, the output from the source flip-flop fails to latch properly in time as the input at the destination flip-flop, which continues to operate with stale data

CLKSCREW Challenges

- Regulator operating limits
  - Overclocking or undervolting attacks require the hardware to be configured far beyond its vendor-suggested operating range
- Self-containment within same device
  - Attack should not affect the execution of the attacking code which resides in the same device as the victim code
- Noisy complex OS environment with interrupts
  - Fault should be injected into the target code without affecting the nontargeted code
- Fault timing needs to be fine-tuned and precise
- Fine-grained timing resolution
  - Fault needs to be transient enough to occur during the intended region of victim code execution

CLKSCREW Solutions

- Regulator operating limits
  - No safeguard limits in the hardware regulators of DVFS to restrict the range of frequencies and voltages that can be configured
  - Reducing the operating voltage lowers the minimum required frequency needed to induce faults
- Self-containment within same device
  - Cores have different frequency regulators
  - Custom kernel driver is created to launch separate threads for the attack and victim code and to pin each of them to separate cores
  - Pinning the attack and victim code in separate cores allows each of them to execute in different frequency domains

CLKSCREW Solutions

- Noisy complex OS environment with interrupts
  - Disable OS interrupts during the entire victim code execution to prevent context switching
- Fault timing needs to be fine-tuned and precise
- Fine-grained timing resolution
  - High-precision timing loops in attack architecture
  - Execution timing of Trustzone code can be profiled with hardware cycle counters that are accessible outside of Trustzone

CLKSCREW Attack Steps

- *Goal* is to induce a fault in a subset of an entire victim thread execution
- *Attack Steps*:
  1. Clearing residual states
     - Before attacking the victim code it must be ensured that the caches do not have any residual data from non-victim code before each fault injection attempt
  2. Profiling for a timing anchor
     - Victim code execution is typically a subset of the entire victim thread execution
     - Need to identify a timing anchor – a consistent point of execution just before the target code to be faulted

CLKSCREW Attack Steps

- *Attack Steps (cont.)*:
  3. Pre-fault timing delay
     - Need to finetune the exact delivery timing of the fault
     - Configure the attack thread to spin-loop with a predetermined number of no-op operation loops F_{pdelay} before inducing the actual fault
  4. Delivering the fault
     - Attack thread raises the frequency of the victim core from F_{volt} to F_{freq_hi}, keep that frequency for F_{dur} loops and then restore the frequency to F_{freq_lo}

Parameter	Description
F_{volt}	Base operating voltage
F_{pdelay}	Number of loops to delay/wait before the fault
F_{freq_hi}	Target value to raise the frequency *to* for the fault
F_{freq_lo}	Base value to raise the frequency *from* for the fault
F_{dur}	Duration of the fault in terms of number of loops

Overview

- Introduction
- DVFS
- The CLKSCREW Attack
- Attacking the ARM Trustzone
- Discussion
- Conclusion

Attacking the ARM Trustzone

1. Confidentiality Attack
   - Infer secret AES key stored within Trustzone
2. Integrity Attack
   - Load self-signed app into Trustzone

Confidentiality Attack

- AES Keys stored within Trustzone can be inferred from outside Trustzone by lower privileged code
- Attacker code influences the computation of higher-privileged code using the energy management timing attack
- Induce a fault during the AES encryption procedure to generate faulty ciphertext
- Execute a differential fault analysis between the original and faulty ciphertexts → Infer the AES encryption key!

Integrity Attack

- RSA – Primary public-key cryptography used for authenticating the loading of firmware images into the isolated execution environment of ARM Trustzone
- Vendor specific firmware are subject to regular updates
  - Update files consist of the updated code, a signature protecting the hash of the code, and a certificate chain
  - Trusted Execution Environment (TEE) authenticates the certificate chain and verifies the integrity of the code updates before loading these signed code updates
- CLKSCREW loads self-signed apps into the Trustzone violating firmware integrity

Overview

- Introduction
- DVFS
- The CLKSCREW Attack
- Attacking the ARM Trustzone
- Discussion
- Conclusion

Discussion

Attack Applicability to Other Platforms

- Energy management mechanisms in the industry is trending towards fine-grained, heterogeneous designs that separate voltage/frequency domains for the cores
- Security vulnerabilities in energy management mechanisms of Intel devices, ARMv8 devices and Cloud computing providers like Amazon AWS, etc. need to be explored

Discussion

- *Hardware-Level Defences*
  - Operating limits in hardware
    - Hard limits can be enforced within the regulators in the form of additional limit-checking logic
  - Separate cross-boundary regulators
    - Maintain different power domains across security boundaries when the isolated environment is active
- *Software-Level Defences*
  - Randomization of the runtime execution of the code to be protected
  - Code execution redundancy and checks
    - Compiling code with checksum integrity verification
    - Executing sensitive code multiple times

Overview

- Introduction
- DVFS
- The CLKSCREW Attack
- Attacking the ARM Trustzone
- Discussion
- Conclusion

Conclusion

- CLKSCREW exposes the perils of security oblivious energy management systems
- Not a hardware or software bug
- Fundamental design flaw in energy management mechanisms
- Does not need physical access to the target devices
- Exploits the energy management software interfaces
- Future energy management designs must take security into consideration

THANK YOU