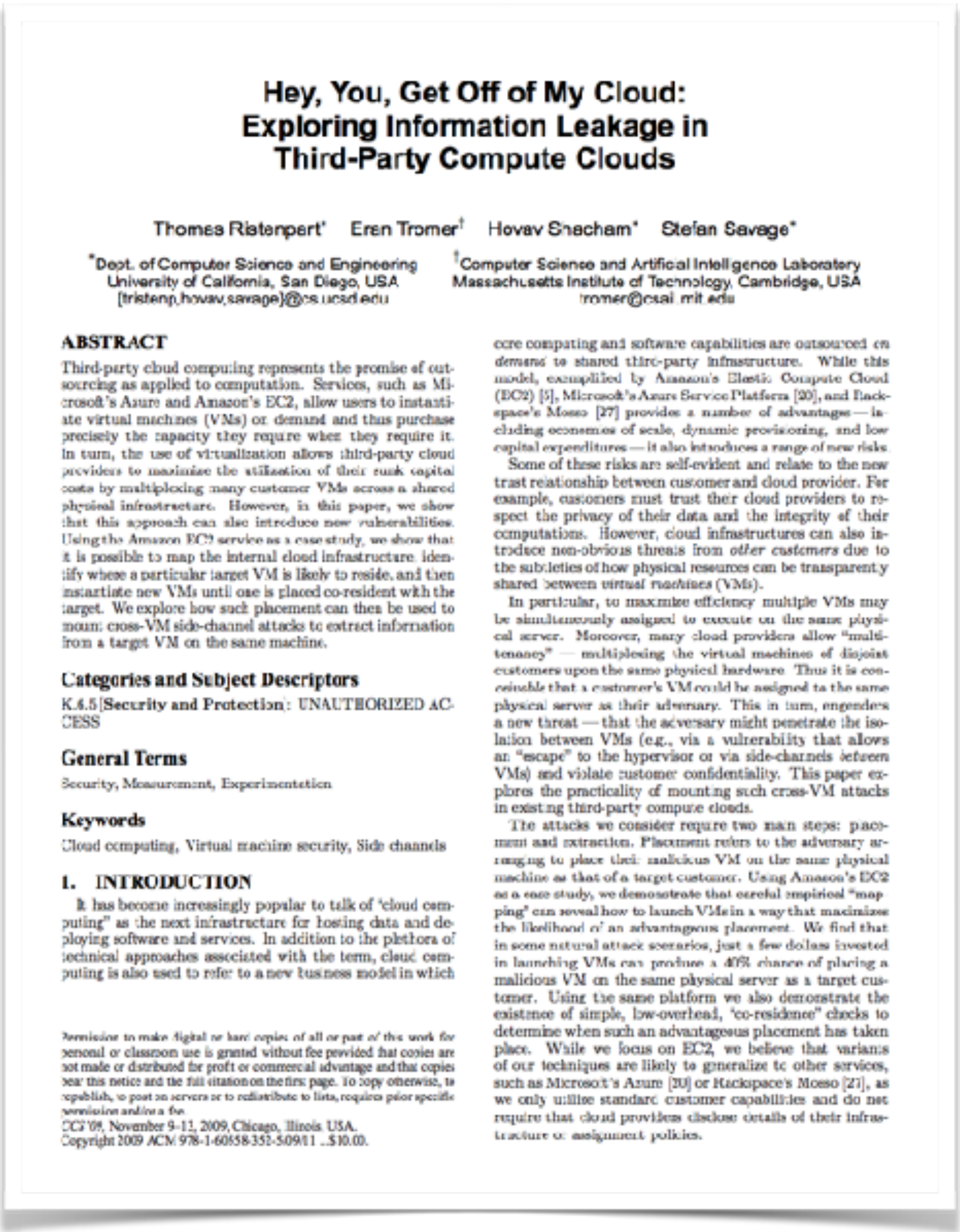
ISOLATION ATTACKS

GRAD SEC

OCT 03 2017

TODAY'S PAPERS



Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds

Thomas Ristenpart* Eran Tromer† Hovav Shacham* Stefan Savage*

*Dept. of Computer Science and Engineering
University of California, San Diego, USA
{tristenp,hovav,savage}@cs.ucsd.edu

†Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, USA
tromer@csail.mit.edu

ABSTRACT

Third-party cloud computing represents the promise of outsourcing as applied to computation. Services, such as Microsoft's Azure and Amazon's EC2, allow users to instantiate virtual machines (VMs) on demand and thus purchase precisely the capacity they require when they require it. In turn, the use of virtualization allows third-party cloud providers to maximize the utilization of their sunk capital costs by multiplexing many customer VMs across a shared physical infrastructure. However, in this paper, we show that this approach can also introduce new vulnerabilities. Using the Amazon EC2 service as a case study, we show that it is possible to map the internal cloud infrastructure, identify where a particular target VM is likely to reside, and then instantiate new VMs until one is placed co-resident with the target. We explore how such placement can then be used to mount cross-VM side-channel attacks to extract information from a target VM on the same machine.

Categories and Subject Descriptors

K.6.5 [Security and Protection]: UNAUTHORIZED ACCESS

General Terms

Security, Measurement, Experimentation

Keywords

Cloud computing, Virtual machine security, Side channels

1. INTRODUCTION

It has become increasingly popular to talk of "cloud computing" as the next infrastructure for hosting data and deploying software and services. In addition to the plethora of technical approaches associated with the term, cloud computing is also used to refer to a new business model in which core computing and software capabilities are outsourced *on demand* to shared third-party infrastructure. While this model, exemplified by Amazon's Elastic Compute Cloud (EC2) [5], Microsoft's Azure Service Platform [20], and Rackspace's Mosso [27] provides a number of advantages — including economies of scale, dynamic provisioning, and low capital expenditures — it also introduces a range of new risks.

Some of these risks are self-evident and relate to the new trust relationship between customer and cloud provider. For example, customers must trust their cloud providers to respect the privacy of their data and the integrity of their computations. However, cloud infrastructures can also introduce non-obvious threats from *other customers* due to the subtleties of how physical resources can be transparently shared between virtual machines (VMs).

In particular, to maximize efficiency multiple VMs may be simultaneously assigned to execute on the same physical server. Moreover, many cloud providers allow "multi-tenancy" — multiplexing the virtual machines of disjoint customers upon the same physical hardware. Thus it is conceivable that a customer's VM could be assigned to the same physical server as their adversary. This in turn, engenders a new threat — that the adversary might penetrate the isolation between VMs (e.g., via a vulnerability that allows an "escape" to the hypervisor or via side-channels between VMs) and violate customer confidentiality. This paper explores the practicality of mounting such cross-VM attacks in existing third-party compute clouds.

The attacks we consider require two main steps: placement and extraction. Placement refers to the adversary arranging to place their malicious VM on the same physical machine as that of a target customer. Using Amazon's EC2 as a case study, we demonstrate that careful empirical "mapping" can reveal how to launch VMs in a way that maximizes the likelihood of an advantageous placement. We find that in some natural attack scenarios, just a few dollars invested in launching VMs can produce a 40% chance of placing a malicious VM on the same physical server as a target customer. Using the same platform we also demonstrate the existence of simple, low-overhead, "co-residence" checks to determine when such an advantageous placement has taken place. While we focus on EC2, we believe that variants of our techniques are likely to generalize to other services, such as Microsoft's Azure [20] or Rackspace's Mosso [27], as we only utilize standard customer capabilities and do not require that cloud providers disclose details of their infrastructure or assignment policies.

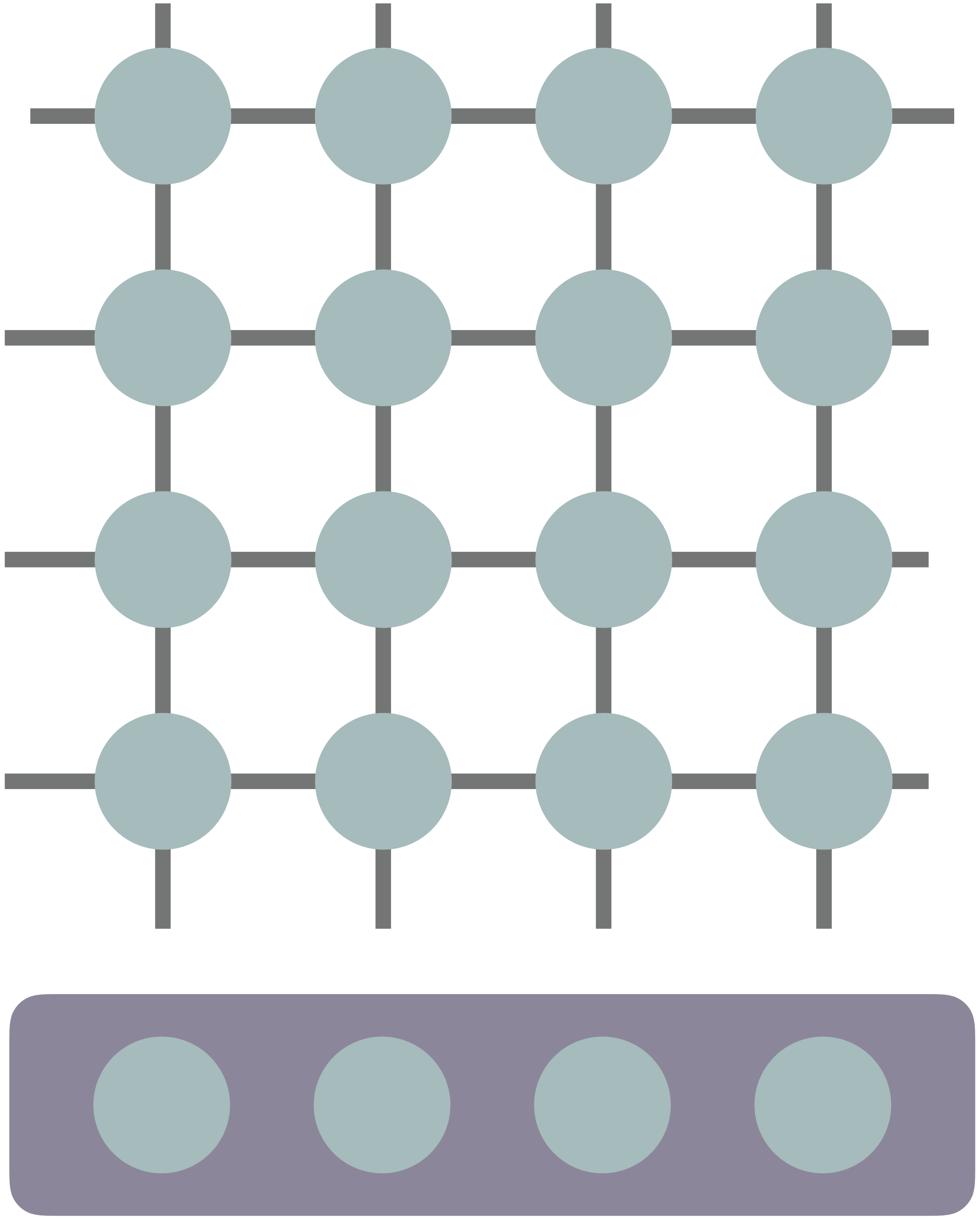
CCS'09, November 9–13, 2009, Chicago, Illinois, USA.

Exploiting the DRAM rowhammer bug to gain kernel privileges

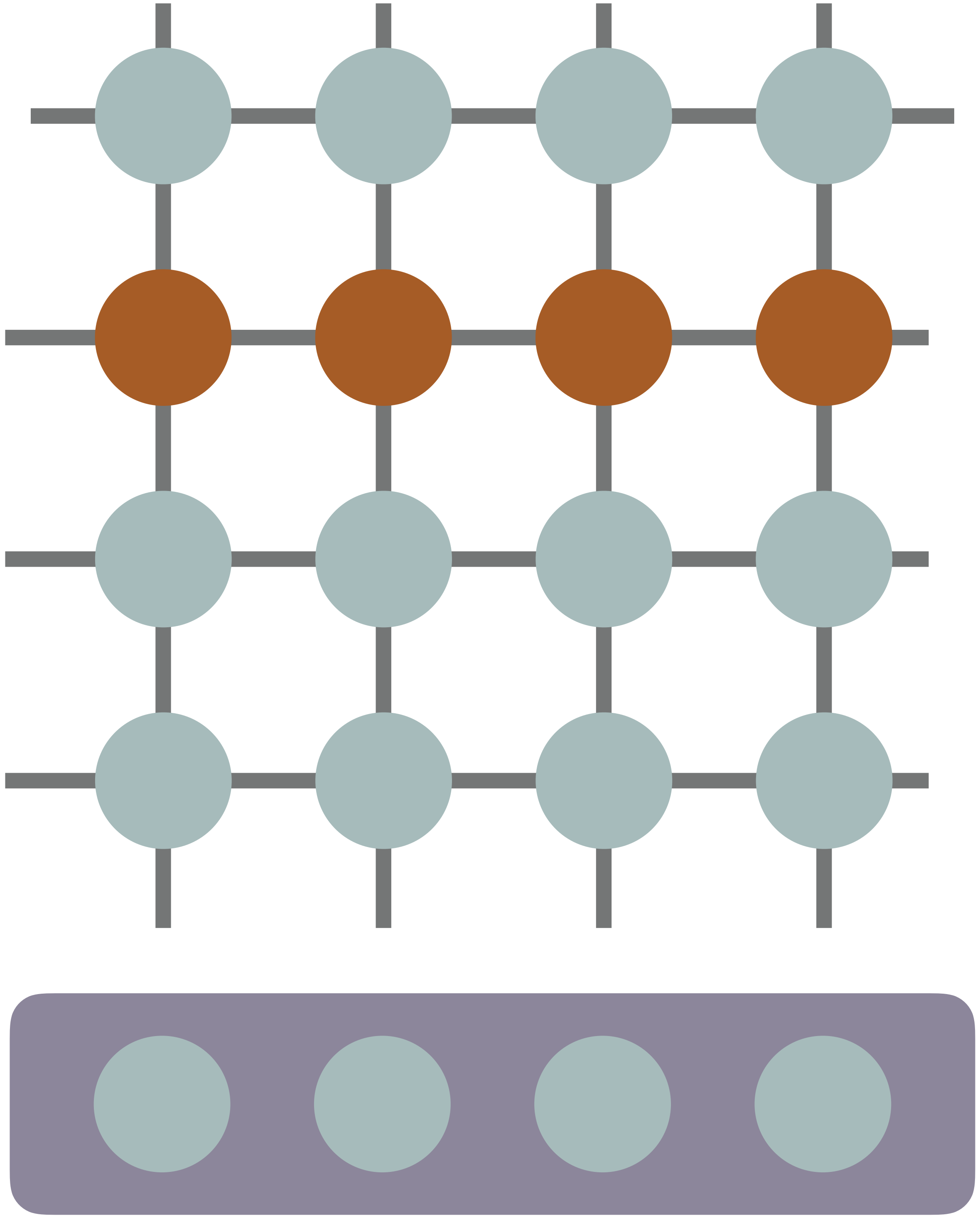
How to cause and exploit single bit errors

Mark Seaborn and Thomas Dullien

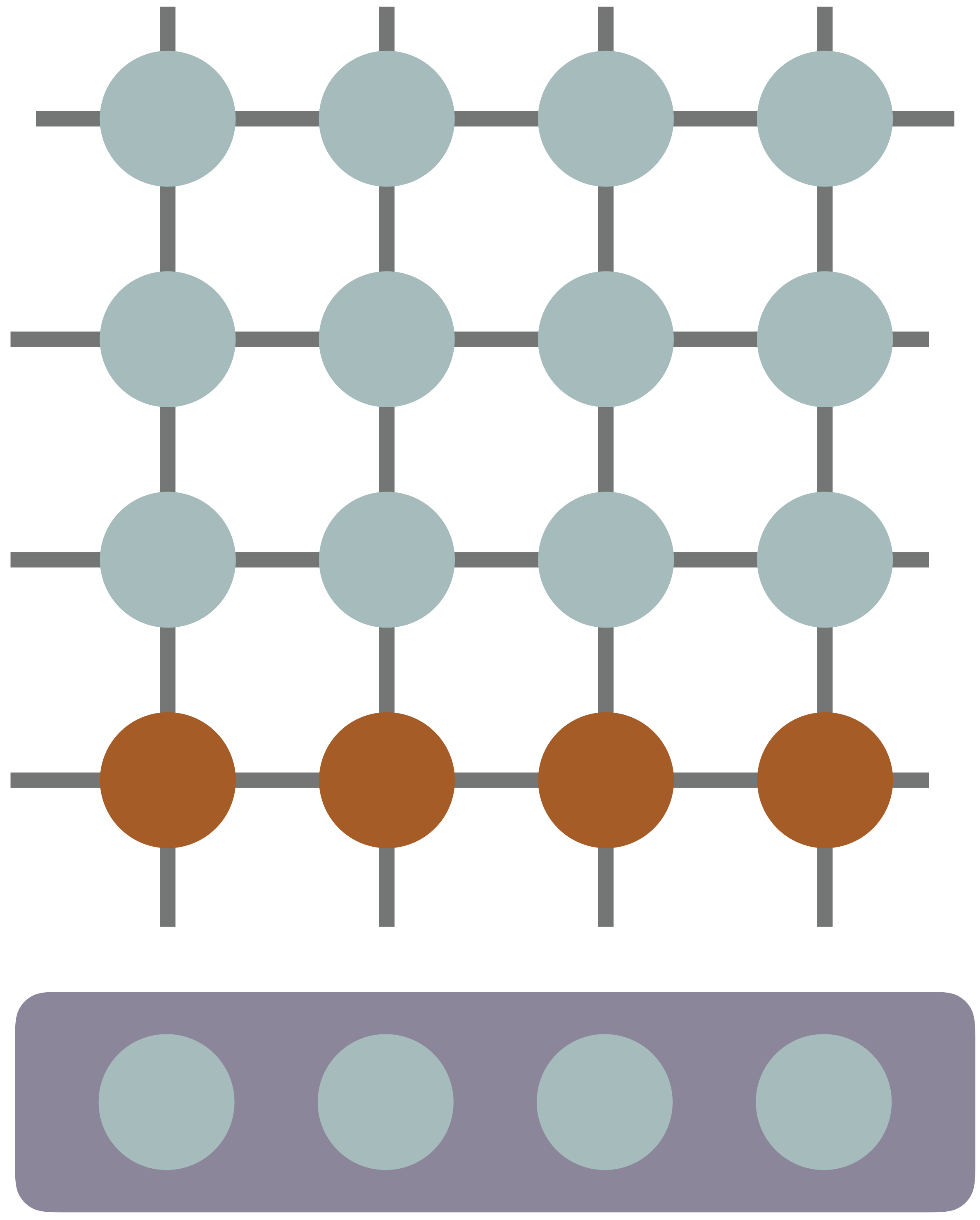
ROWHAMMER



ROWHAMMER

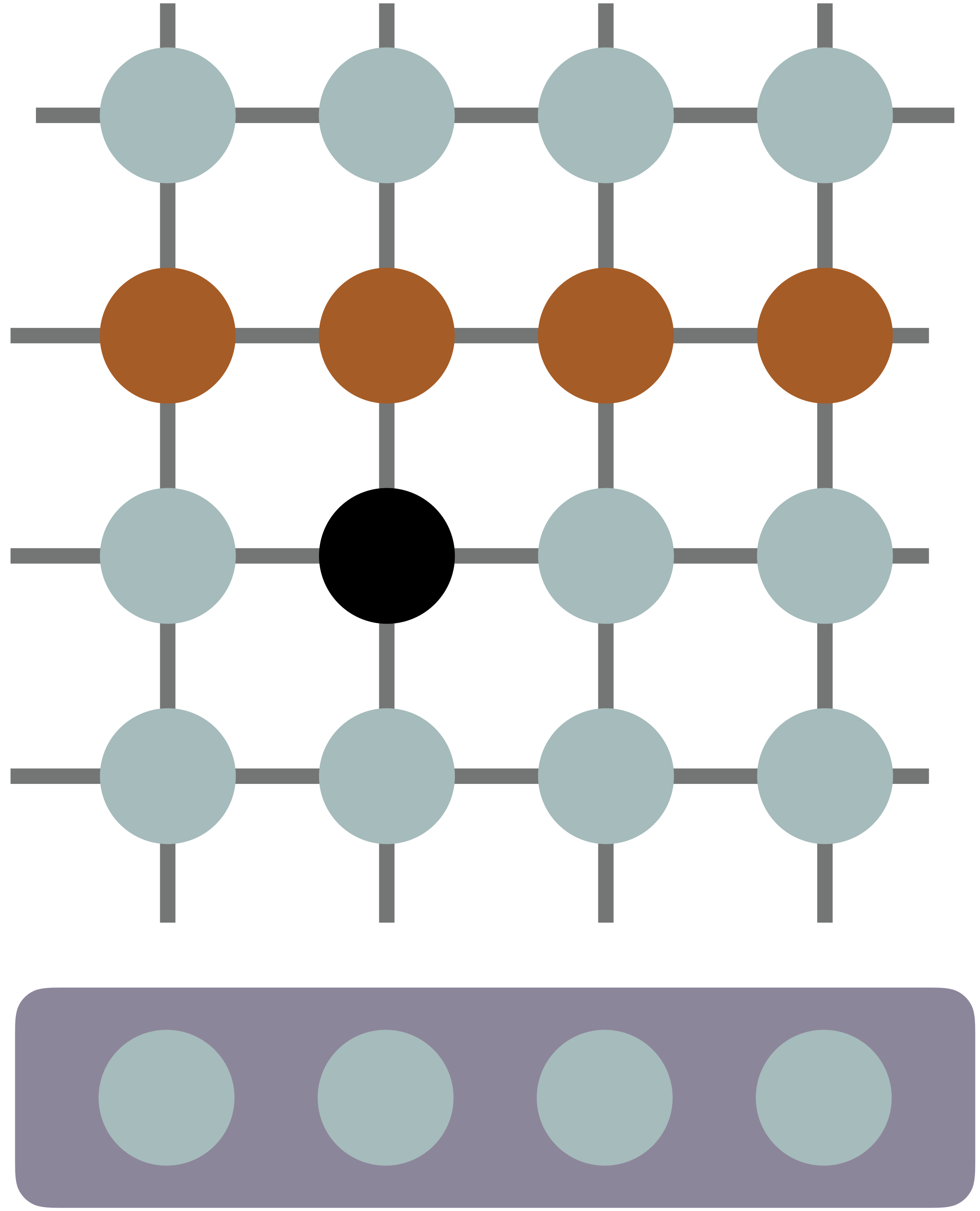


ROWHAMMER



ROWHAMMER

Bit Flip

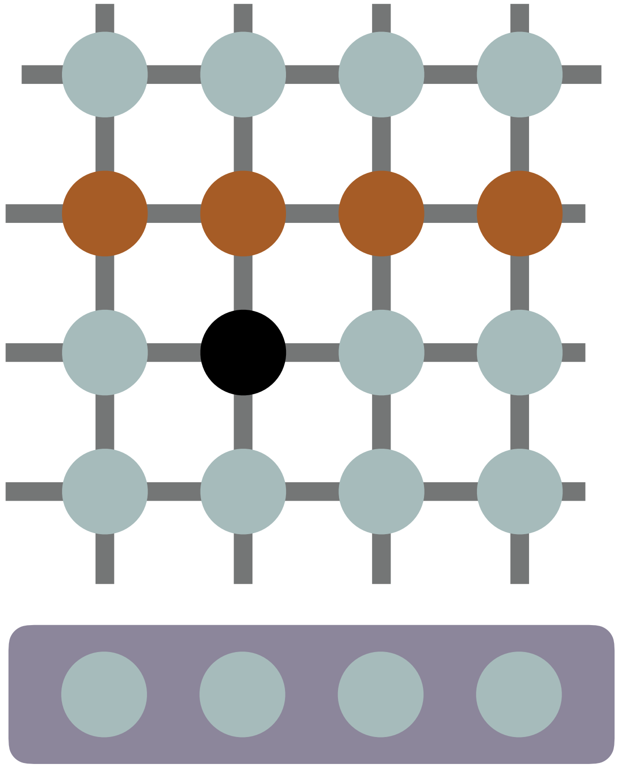


ROWHAMMER

A hardware glitch

Causes charge to leak in DRAM

DRAM row activations cause bit flips



ROWHAMMER + NACL

NaCl exploit

Safe instruction sequence:

```
andl $~31, %eax   // Truncate address to 32 bits
                  // and mask to be 32-byte-aligned.
addq %r15, %rax   // Add %r15, the sandbox base address.
jmp *%rax         // Indirect jump.
```

NaCl sandbox model:

- Prevent jumping into the middle of an x86 instruction
- Indirect jumps can only target 32-byte-aligned addresses

ROWHAMMER + NACL

NaCl exploit

Bit flips make instruction sequence unsafe:

```
andl $~31, %eax    // Truncate address to 32 bits
                   // and mask to be 32-byte-aligned.
addq %r15, %rax    // Add %r15, the sandbox base address.
jmp *%rax          // Indirect jump.
```

e.g. %eax → %ecx

- Allows jumping to a non-32-byte-aligned address

ROWHAMMER + NACL

Hiding unsafe code in NaCl

Existing technique for exploiting non-bundle-aligned jump:

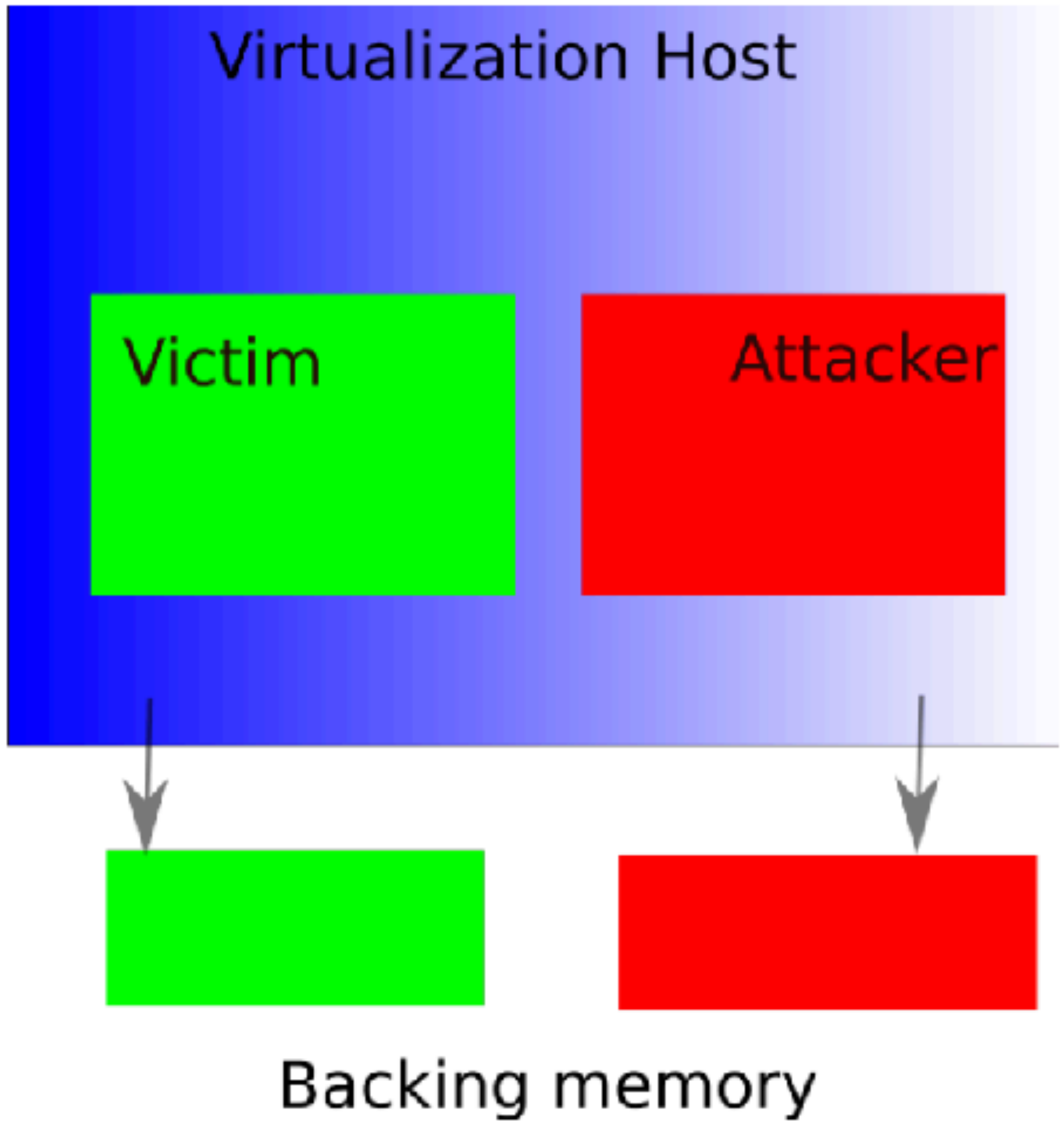
```
20ea0: 48 b8 0f 05 eb 0c f4 f4 f4 f4
           movabs $0xf4f4f4f40ceb050f, %rax
```

This conceals:

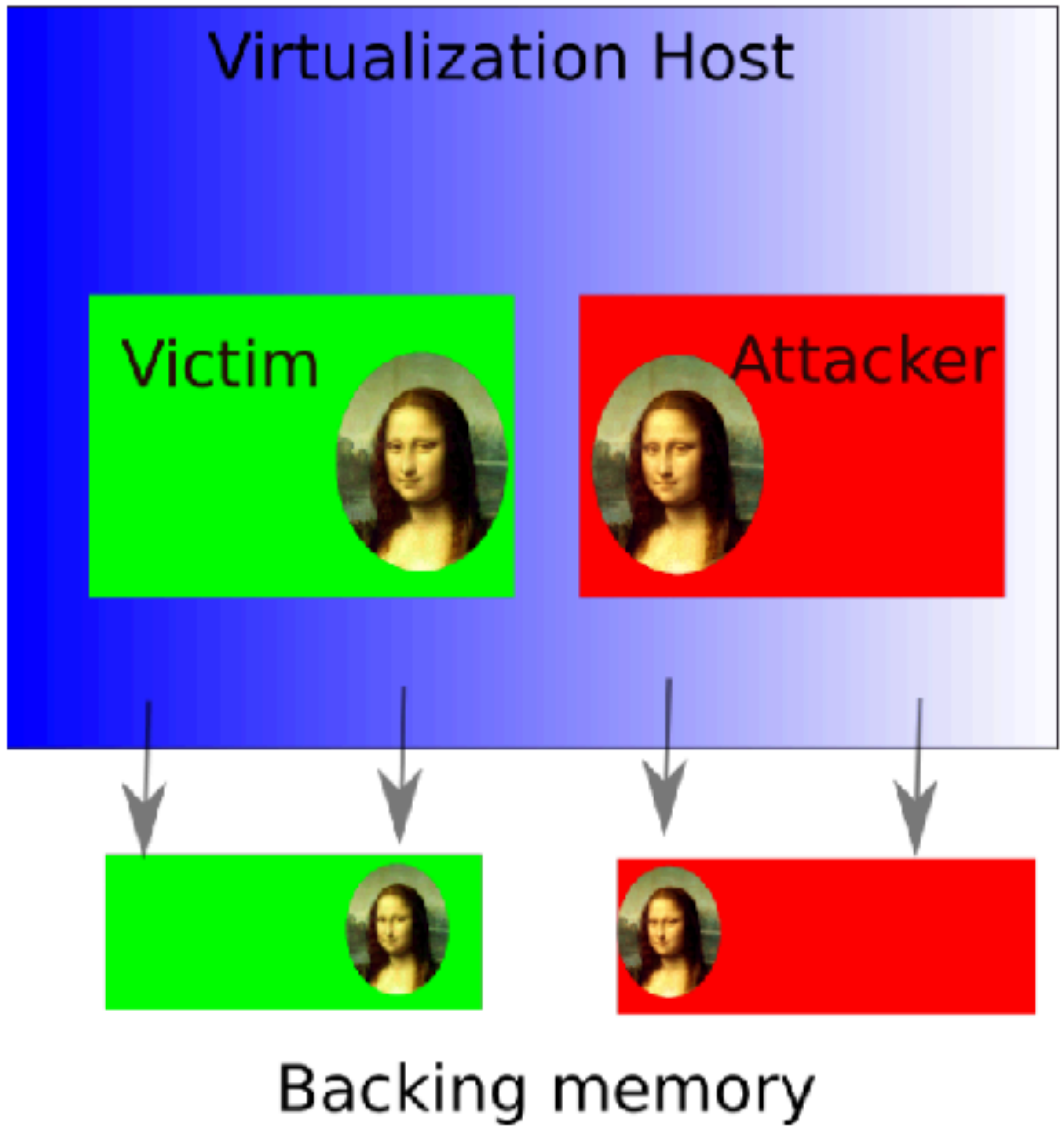
```
20ea2: 0f 05    syscall
20ea4: eb 0c    jmp ...   // Jump to next hidden instr
20ea6: f4       hlt       // Padding
```

Insert ROP-like gadgets in your own code

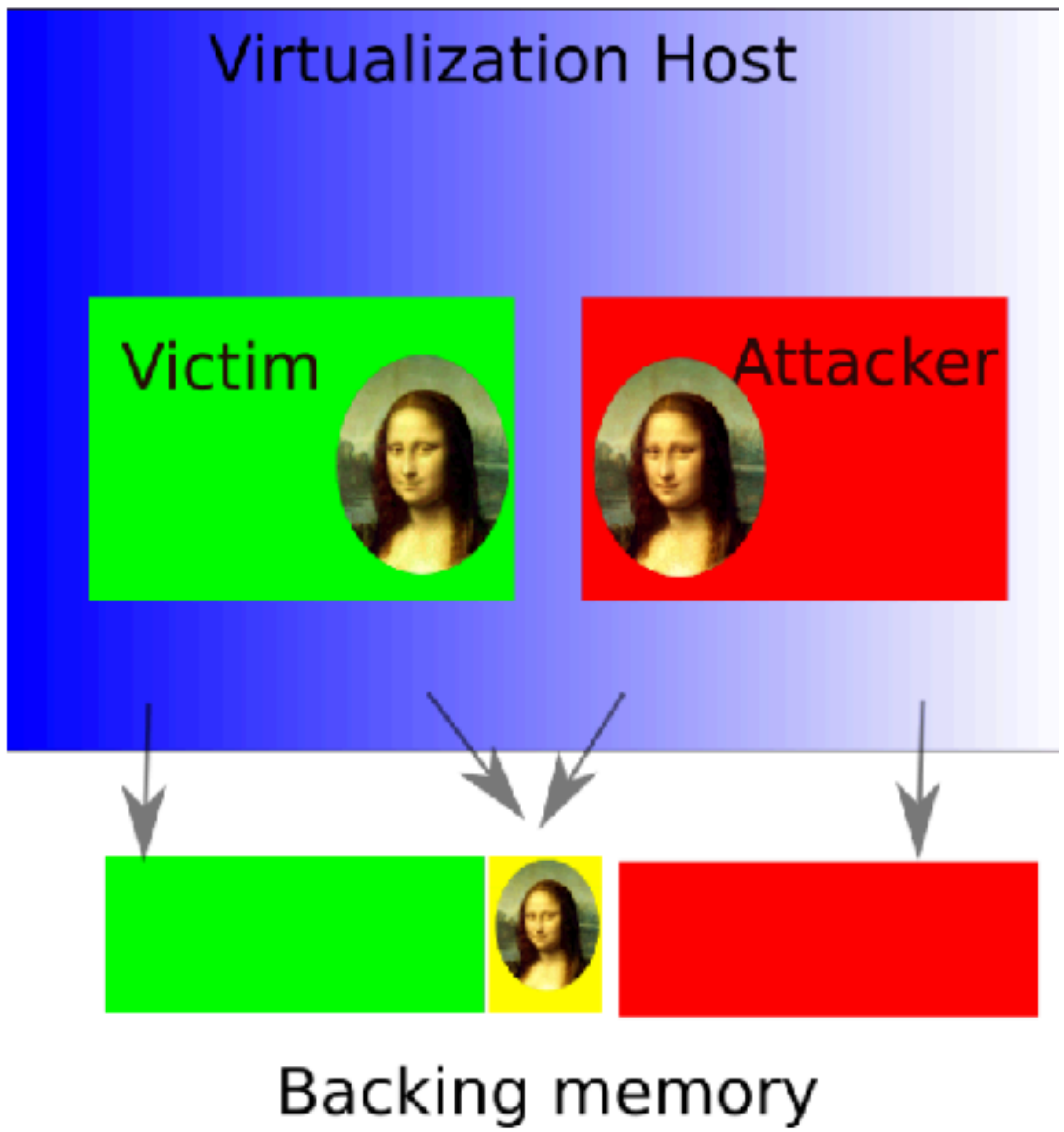
ROWHAMMER + MEMORY DEDUPLICATION



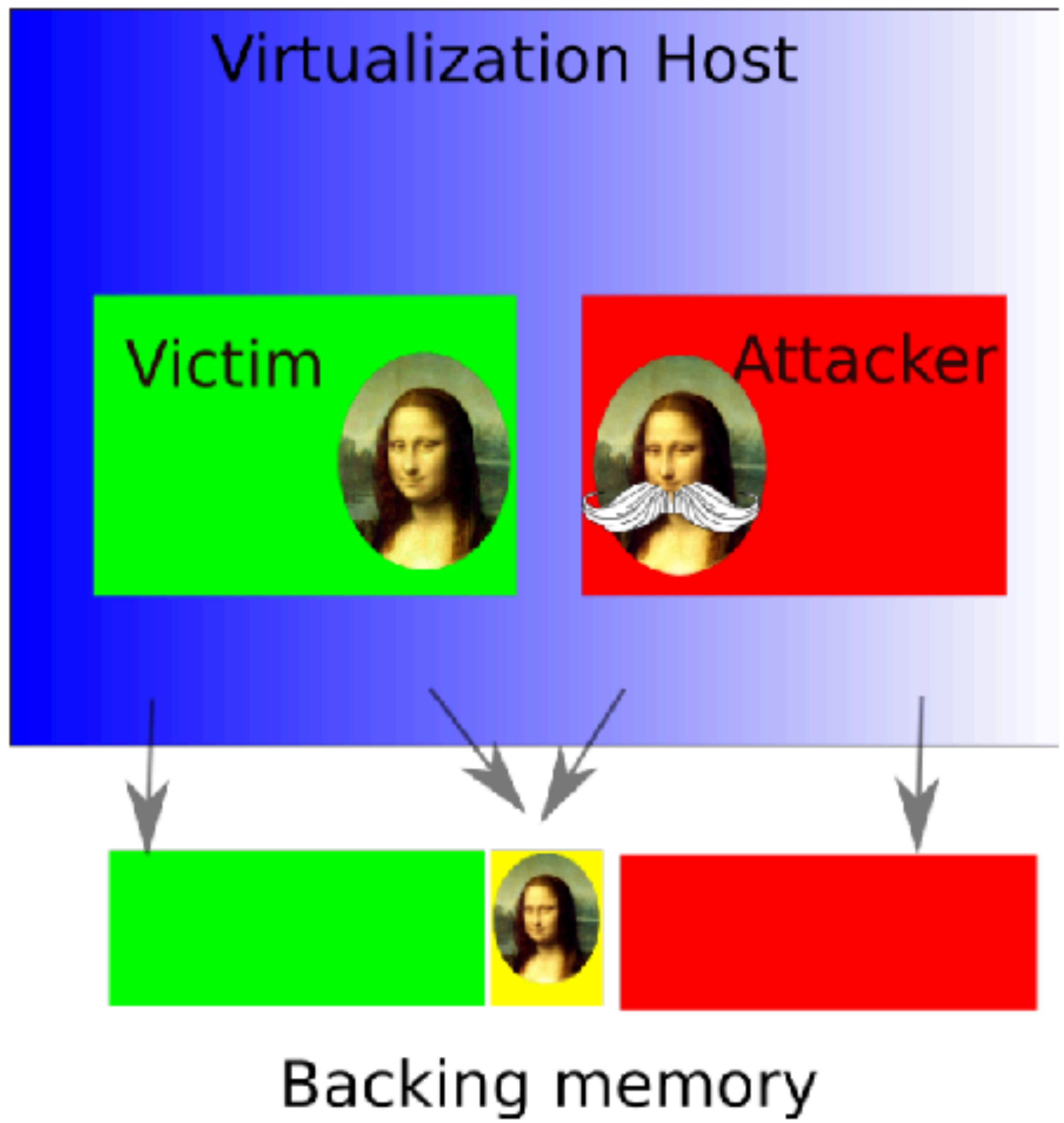
ROWHAMMER + MEMORY DEDUPLICATION



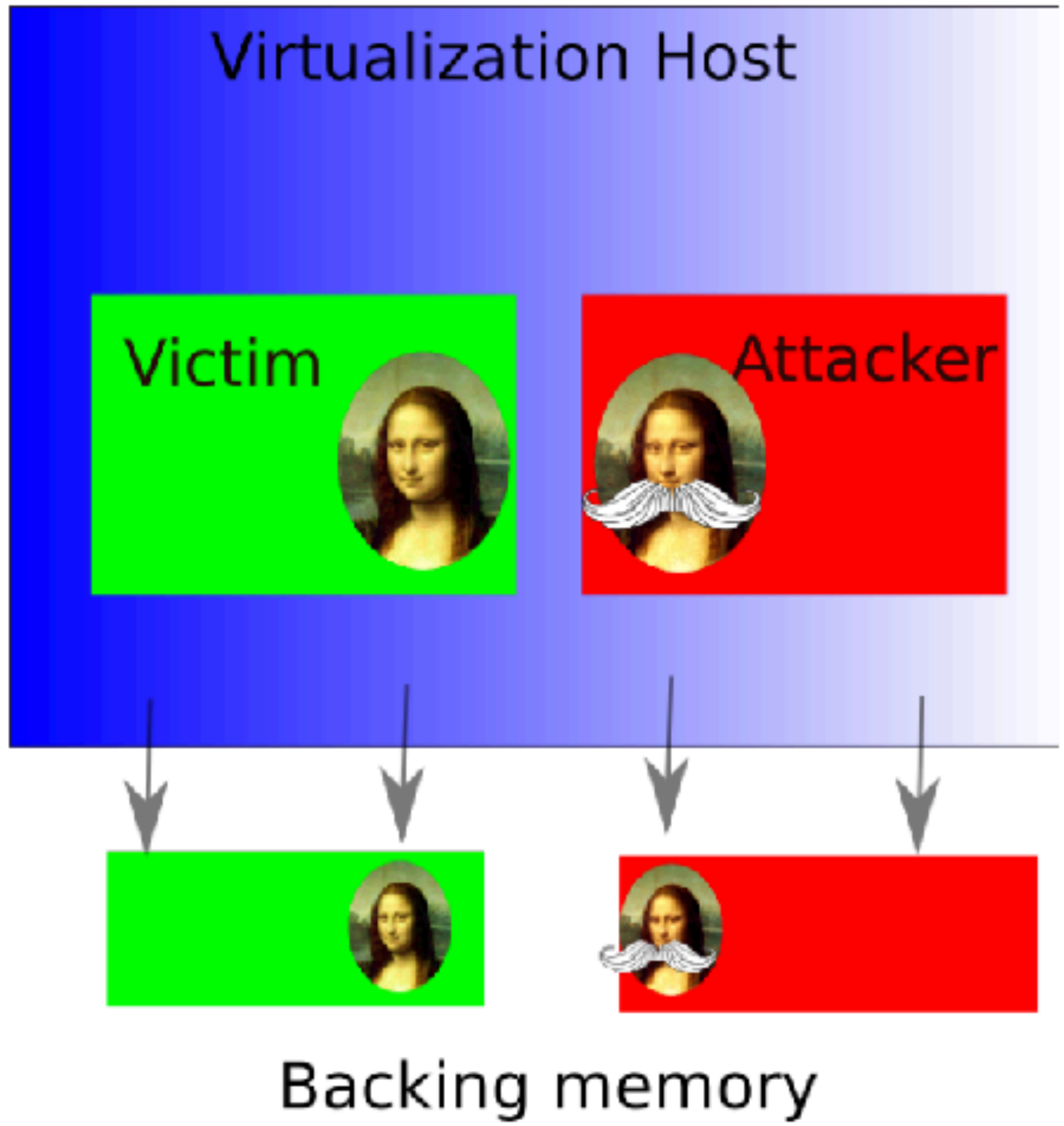
ROWHAMMER + MEMORY DEDUPLICATION



ROWHAMMER + MEMORY DEDUPLICATION

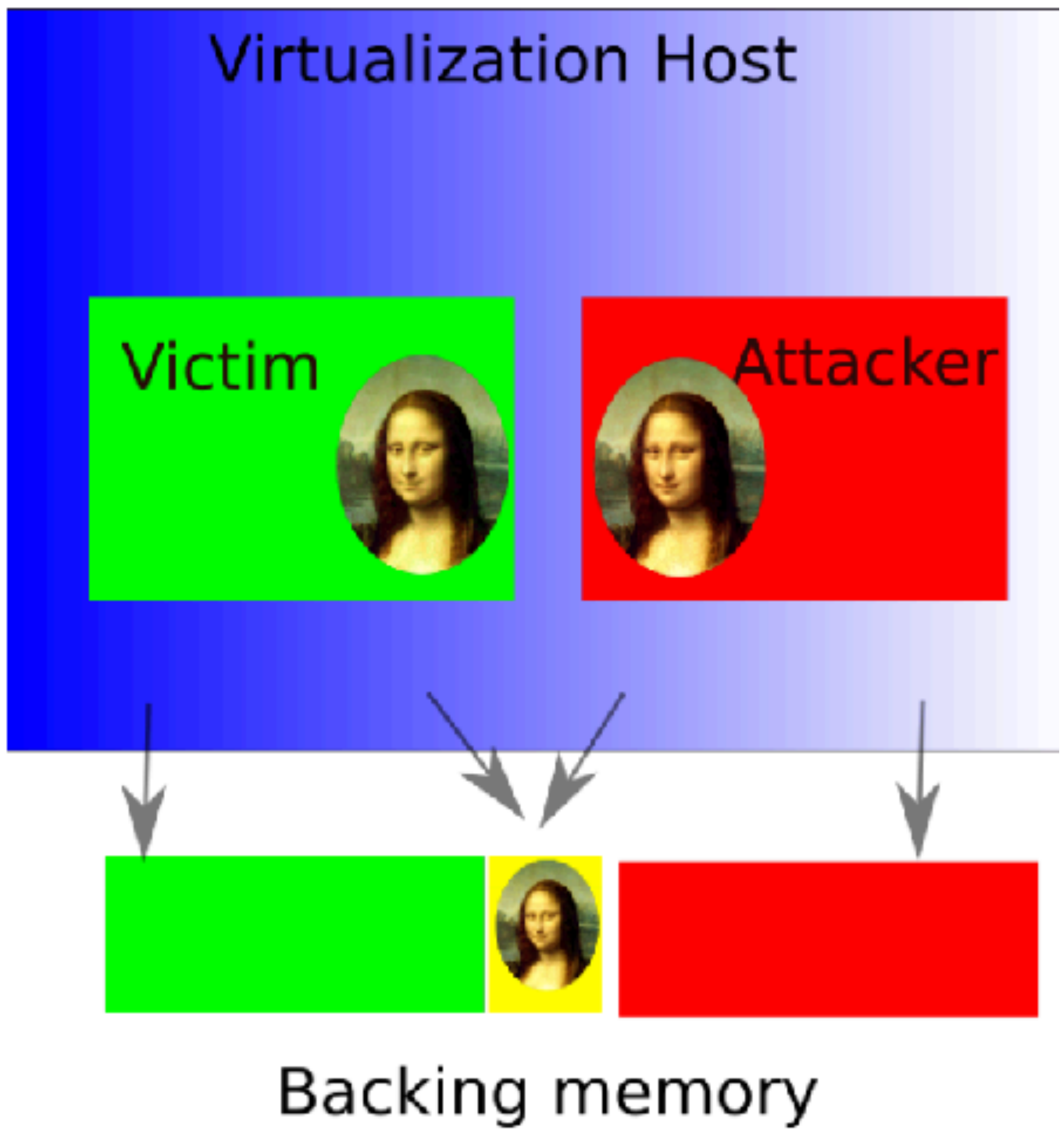


ROWHAMMER + MEMORY DEDUPLICATION

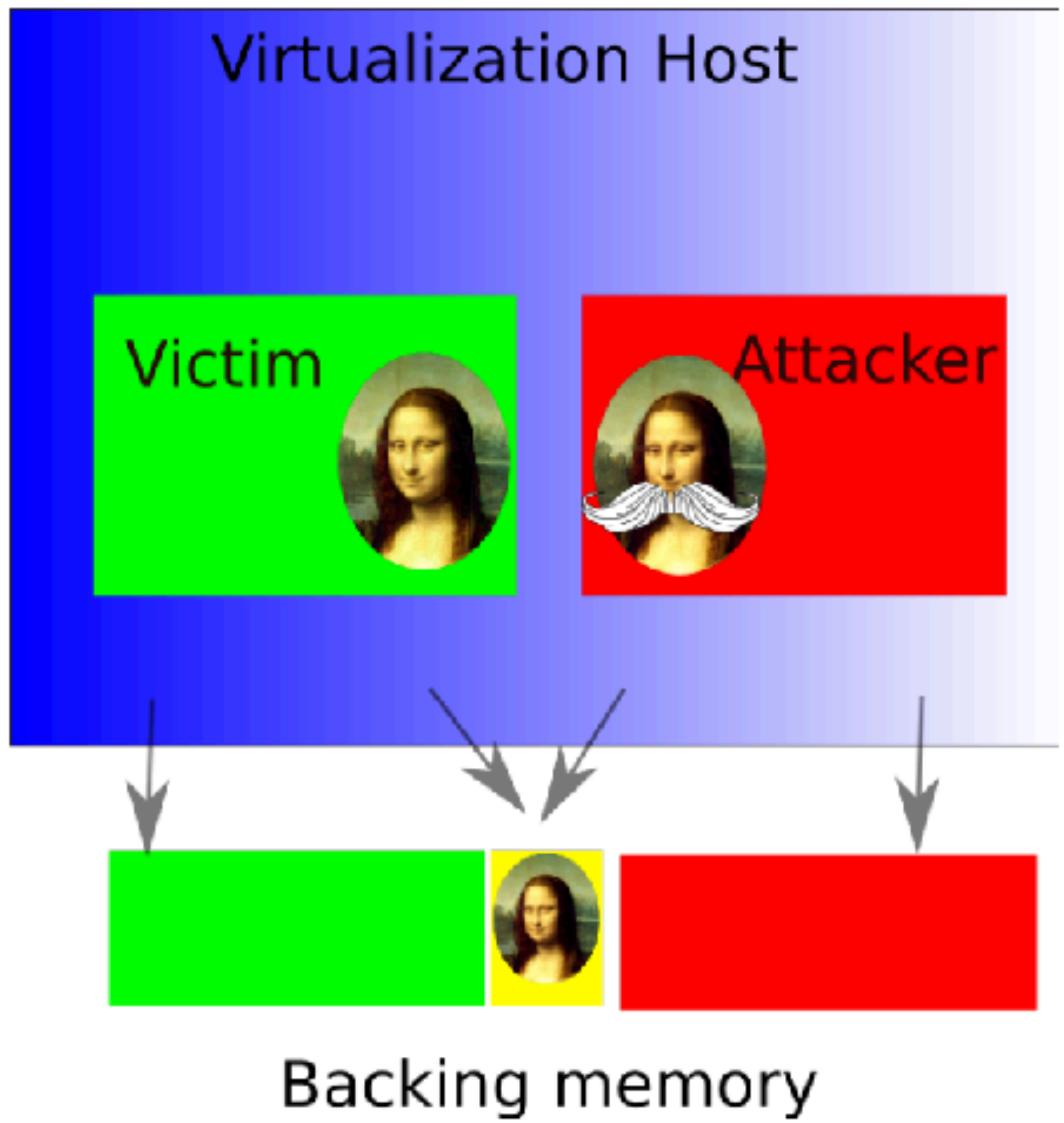


Copy-on-write (COW) ensures isolation

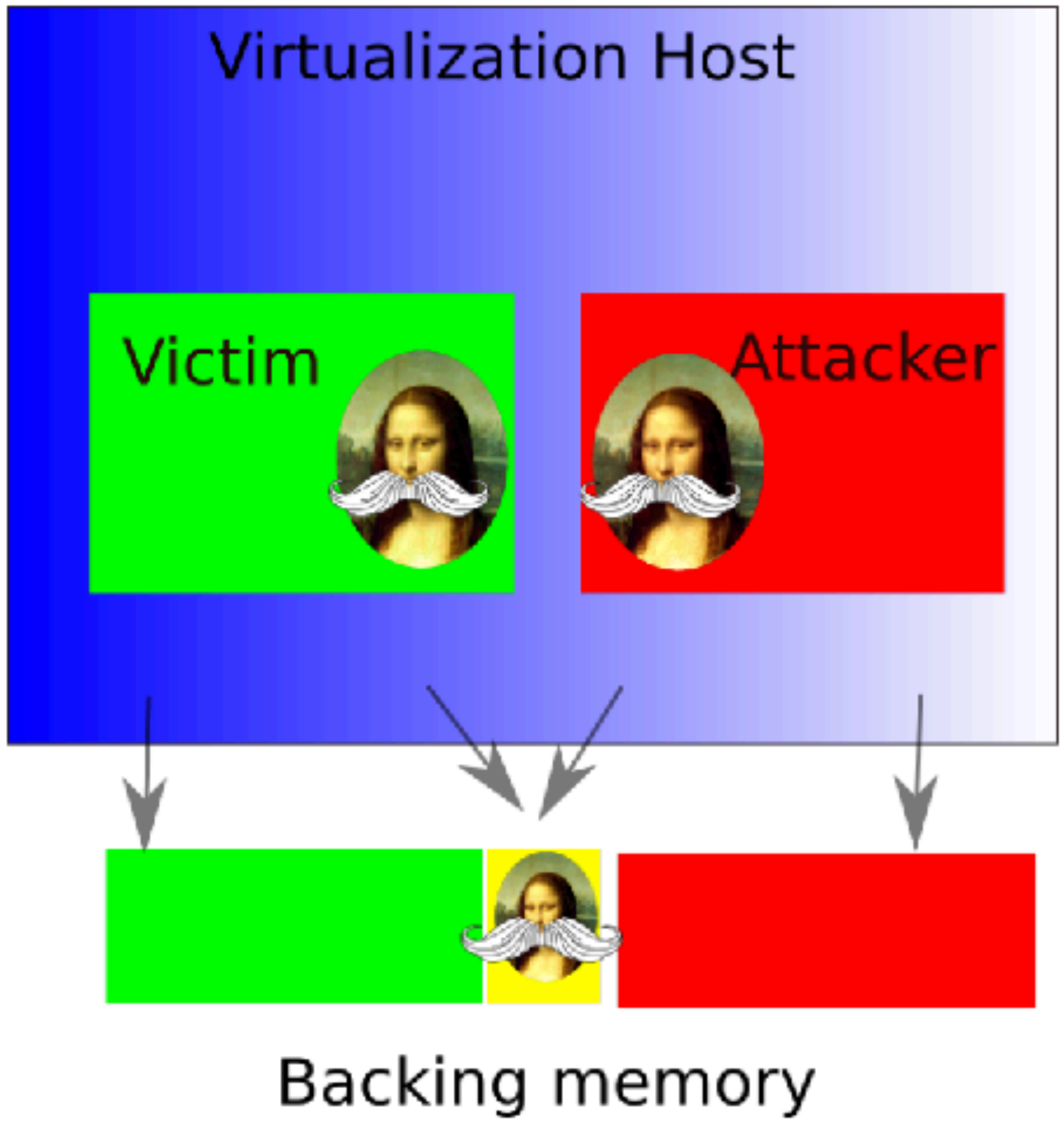
ROWHAMMER + MEMORY DEDUPLICATION



ROWHAMMER + MEMORY DEDUPLICATION



ROWHAMMER + MEMORY DEDUPLICATION



Rowhammer breaks COW

FLIP FENG SHUI

Flip Feng Shui: Hammering a Needle in the Software Stack

Kaveh Razavi* Vrije Universiteit Amsterdam
Ben Gras* Vrije Universiteit Amsterdam
Erik Bosman Vrije Universiteit Amsterdam
Bart Preneel Katholieke Universiteit Leuven
Cristiano Giuffrida Vrije Universiteit Amsterdam
Herbert Bos Vrije Universiteit Amsterdam

* Equal contribution joint first authors

Abstract

We introduce Flip Feng Shui (FFS), a new exploitation vector which allows an attacker to induce bit flips over *arbitrary* physical memory in a *fully controlled* way. FFS relies on hardware bugs to induce bit flips over memory and on the ability to surgically control the physical memory layout to corrupt attacker targeted data anywhere in the software stack. We show FFS is possible today with very few constraints on the target data, by implementing an instance using the *Rowhammer bug* and *memory deduplication* (an OS feature widely deployed in production). Memory deduplication allows an attacker to reverse-map any physical page into a virtual page she owns as long as the page's contents are known. Rowhammer, in turn, allows an attacker to flip bits in controlled (initially unknown) locations in the target page.

We show FFS is extremely powerful: a malicious VM in a practical cloud setting can gain unauthorized access to a co-hosted victim VM running OpenSSH. Using FFS, we exemplify end-to-end attacks breaking OpenSSH public-key authentication, and forging GPG signatures from trusted keys, thereby compromising the Ubuntu/Debian update mechanism. We conclude by discussing mitigations and future directions for FFS attacks.

1 Introduction

The demand for high-performance and low-cost computing translates to increasing complexity in hardware and software. On the hardware side, the semiconductor industry packs more and more transistors into chips that serve as a foundation for our modern computing infrastructure. On the software side, modern operating systems are packed with complex features to support efficient resource management in cloud and other performance-sensitive settings.

Both trends come at the price of reliability and, inevitably, security. On the hardware side, components are increasingly prone to failures. For example, a large fraction of the DRAM chips produced in recent years are prone to bit flips [34, 51], and hardware errors in CPUs are expected to become mainstream in the near future [10, 16, 37, 53]. On the software side, widespread features such as memory or storage deduplication may serve as side channels for attackers [8, 12, 31]. Recent work analyzes some of the security implications of both trends, but so far the attacks that abuse these hardware/software features have been fairly limited—probabilistic privilege escalation [51], in-browser exploitation [12, 30], and selective information disclosure [8, 12, 31].

In this paper, we show that an attacker abusing modern hardware/software properties can mount much more sophisticated and powerful attacks than previously believed possible. We describe Flip Feng Shui (FFS), a new exploitation vector that allows an attacker to induce bit flips over *arbitrary* physical memory in a *fully controlled way*. FFS relies on two underlying primitives: (i) the ability to induce bit flips in controlled (but not predetermined) physical memory pages; (ii) the ability to control the physical memory layout to reverse-map a target physical page into a virtual memory address under attacker control. While we believe the general vector will be increasingly common and relevant in the future, we show that an instance of FFS, which we term dFFS (i.e., deduplication-based FFS), can already be implemented on today's hardware/software platforms with very few constraints. In particular, we show that by abusing Linux' memory deduplication system (KSM) [6] which is very popular in production clouds [8], and the widespread Rowhammer DRAM bug [34], an attacker can *reliably* flip a single bit in *any* physical page in the software stack with known contents.

Despite the complete absence of software vulnerabilities, we show that a practical Flip Feng Shui attack can have devastating consequences in a common cloud setting. An attacker controlling a cloud VM can abuse

USENIX Association 25th USENIX Security Symposium 1

Exploits memory deduplication + Rowhammer to attack a co-resident VM

See the demo starting at about 17:00

https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/razavi