Software Exploitation:
Hardware is the New Black

Cristiano Giuffrida
Vrije Universiteit Amsterdam

https://vusec.net
Bug-free Software: Expectations
Bug-free Software: Reality
Credits

- Erik Bosman
- Ben Gras
- Kaveh Razavi
- Victor van der Veen
- Andrei Tatar
- Lucian Cojocar
- Herbert Bos

VU Sec

https://vusec.net
Software Exploitation:

2014
Software Exploitation:

2014

Bugs, Bugs Everywhere!
Software Exploitation:

2014

Attacker Exploits Vulnerable Software
Software Exploitation:

2014

Attacker Owns App
Software Exploitation:

2014

Attacker Owns System
Software Exploitation: 2014

- Systems security problems caused by **bugs**
  - Software and configuration bugs
  - Weak security implementations

- **Impossible** to write software without bugs
  - However, we can mitigate their impact
  - Many defenses proposed by industry and academia
Defenses

Out of Control (S&P '14)
SROP (S&P '14)
Size Does Matter (USENIX Sec '14)
Allocation Oracles (USENIX Sec '16)
Thread Spraying (USENIX Sec '16)
Dedup est machina (S&P '16)
Flip Feng Shui (USENIX Sec '16)
Drammer (CCS'16)
AnC (NDSS'17)
VUzzer (NDSS'17)

Attacks

https://vusec.net
Software Exploitation:

2019

Exploits Difficult:
- Mitigations
- Verification
Software Exploitation:

2019

How to Find Memory R/W Primitives?
Software Exploitation:

2019

Memory R: Hw/Sw Side Channels
Software Exploitation:

2019

Memory W: Hardware Glitches
Software Exploitation:

2019

Memory R/W: Back to Reliable Exploits
Software Exploitation:

2019

Memory R/W: Back to Reliable Exploits
Software Exploitation: 2019

- Even if the software is perfect...
  - ...with no bugs, well-configured, and latest defenses
  - ...or even formally verified
  - ...it is still vulnerable!

- Attackers abuse properties of modern hw and sw for reliable exploitation

- We’ll look at some examples with different properties
EXAMPLE 1

Bug-free Exploitation in Browsers
Dedup Est Machina

- Published at IEEE S&P 2016
  - with Erik, Kaveh, Herbert
- Won Pwnie Award at Black HAT 2016

"Most Innovative Research"

- Complete exploit of Microsoft Edge browser on Windows 10 from malicious JavaScript
  - ...without relying on a single software bug
Dedup Est Machina

Memory deduplication
(software side channel)
Dedup Est Machina

Memory deduplication
(software side channel)
+
Rowhammer
(hardware glitch)
Dedup Est Machina

Memory deduplication
(software side channel)
+
Rowhammer
(hardware glitch)

Exploit MS Edge without software bugs
(from JavaScript)
Dedup Est Machina: Overview

Memory deduplication

- Leak randomized heap and code pointers
Dedup Est Machina: Overview

Memory deduplication
- Leak randomized heap and code pointers
Dedup Est Machina: Overview

Memory deduplication

- Leak randomized heap and code pointers
- Create a fake JavaScript object
Dedup Est Machina: Overview

Memory deduplication
- Leak randomized heap and code pointers
- Create a fake JavaScript object

\[ + \]

Rowhammer
- Create a reference to our fake object
Dedup Est Machina: Overview

Memory deduplication

- Leak randomized heap and code pointers
- Create a fake JavaScript object

Rowhammer

- Create a reference to our fake object
Dedup Est Machina: Overview

Memory deduplication

- Leak randomized heap and code pointers
- Create a fake JavaScript object

Rowhammer

- Create a reference to our fake object
Memory Deduplication

- A strategy to reduce physical memory usage
- Removes duplication in physical memory
- Common in virtualization environments

Enabled by **default on Windows**
- Windows 8.1
- Windows 10
Memory Deduplication: Mechanics

physical memory

process A

process B
Memory Deduplication: Mechanics

physical memory

process A

process B
Memory Deduplication: Mechanics

physical memory

process A

process B
Memory Deduplication: Mechanics

physical memory

process A

process B
Memory Deduplication: Mechanics

physical memory

process A

process B
Memory Deduplication: Mechanics

physical memory

process A

process B
Memory Deduplication: The Problem

- Deduplicated memory is origin-agnostic
- Merges pages across security boundaries
- Attackers can use this as a side channel!
Memory Deduplication: Timing Side Channel

normal write
Memory Deduplication: Timing Side Channel

normal write

write
Memory Deduplication: Timing Side Channel

normal write

write

copy on write (due to deduplication)
Memory Deduplication: Timing Side Channel

normal write

write

copy on write (due to deduplication)

trap to kernel
Memory Deduplication: Timing Side Channel

- normal write
- copy on write (due to deduplication)
  - trap to kernel
  - copy whole page
Memory Deduplication: Timing Side Channel

normal write

write

copy on write (due to deduplication)

trap to kernel  copy whole page  update page tables
Memory Deduplication:
Timing Side Channel

- normal write
- copy on write (due to deduplication)
  - trap to kernel
  - copy whole page
  - update page tables
  - return from kernel
Memory Deduplication: Timing Side Channel

normal write

write

copy on write (due to deduplication)

* trap to kernel

copy whole page

update page tables

return from kernel

write
Memory Deduplication: Side-channel Leaks

Attacker can now leak 1 bit of information (directly from JavaScript and system-wide)

“Does the victim process have this page in memory?”
Memory Deduplication: Side-channel Leaks

- Very coarse-grained. Still interesting?
  - Is user logged into bank website X?

Bank of America

Wells Fargo

Citi

JPMorgan Chase
For software exploitation, 1 bit won’t really cut it (e.g., need to leak 64-bit pointers for MS Edge)

“Can we generalize this to leaking arbitrary data like randomized pointers or passwords?”
Dedup Est Machina: Challenges

Challenge 1:

The secret we want to leak does not span an entire memory page
Dedup Est Machina: Challenges

Turning a secret into a page

secret
Dedup Est Machina: Challenges

Turning a secret into a page

secret

known data

secret page
Challenge 2:
The secret to leak has too much entropy to leak it all at once
Dedup Est Machina: Challenges

Challenge 2:
The secret to leak has too much entropy to leak it all at once

Primitive #1
Primitive #2
Primitive #3
Dedup Est Machina: Primitives

Primitive #1: Alignment Probing

secret → known data

secret page
Dedup Est Machina: Primitives

Primitive #1: Alignment Probing

secret

known data

secret page
Dedup Est Machina: Overview

Memory deduplication

- Leak randomized heap and code pointers
Dedup Est Machina: Leaking Code Pointer (#1)

JIT Function Epilogue in MS Edge

secret

mov RCX,0x1c20   mov RAX, [code address]   jmp RAX   trap   trap   trap   trap   trap   trap   trap   trap   trap   trap   ...

known data
Dedup Est Machina: Leaking Code Pointer (#1)

JIT Function Epilogue in MS Edge

```
mov RCX, 0x1c20  mov RAX, [code address]  jmp RAX  trap  trap  trap
trap  trap  trap  trap  trap  trap
trap  trap  trap  trap  trap  trap
trap  trap  trap  trap  trap  trap
trap  trap  trap  trap  trap  trap
trap  trap  trap  trap  trap  trap
```
Dedup Est Machina: Leaking Code Pointer (#1)

JIT Function Epilogue in MS Edge
Dedup Est Machina: Leaking Code Pointer (#1)

JIT Function Epilogue in MS Edge
Dedup Est Machina: Overview

Memory deduplication
- Leak randomized heap and code pointers
Dedup Est Machina: Leaking Heap Pointer

- Heap pointers are word aligned
  - Alignment probing won’t cut it
- Time for the *Birthday HeapSpray* primitive!

“*How do we leak a heap pointer if we can only leak the secret all at once?”*
Dedup Est Machina: Birthday Paradox

- Only 23 people for a 50% same-birthday chance

- You compare everyone with everyone else
  → Any match suffices!
Birthday Heapspray

+1M, +1M, +1M, ...

+128M,
+128M,
+128M,
...
Dedup Est Machina: Leaking Heap Pointer (#3)

Birthday Heapspray

secret pages (allocated addresses)
Dedup Est Machina: Leaking Heap Pointer (#3)

Birthday Heapspray

secret pages (allocated addresses)

probe pages (guessed addresses)
Dedup Est Machina: Leaking Heap Pointer (#3)

Birthday Heapspray

secret pages (allocated addresses)

probe pages (guessed addresses)
Dedup Est Machina: Overview

Memory deduplication

- Leak randomized heap and code pointers
- Create a fake JavaScript object
Dedup Est Machina: Creating a Fake Object

Fake JavaScript Uint8Array
Dedup Est Machina: Creating a Fake Object

Fake JavaScript Uint8Array

array header

JavaScript Array
Dedup Est Machina: Creating a Fake Object

Fake JavaScript Uint8Array

array header

JavaScript Array

array header

JavaScript Array

array data
Memory deduplication

- Leak randomized heap and code pointers
- Create a fake JavaScript object

Rowhammer

- Create a reference to our fake object
Dedup Est Machina: Referencing the Fake Object

Rowhammer

rows

DDR memory
Dedup Est Machina: Referencing the Fake Object

Rowhammer

DDR memory

row activation
Dedup Est Machina: Referencing the Fake Object

Rowhammer

DDR memory

row activation
Dedup Est Machina: Referencing the Fake Object

Rowhammer

DDR memory

row activation
Dedup Est Machina: Referencing the Fake Object

Double-sided Rowhammer

row activation

row activation

DDR memory
Dedup Est Machina: Referencing the Fake Object

Double-sided Rowhammer

physical memory
Dedup Est Machina: Referencing the Fake Object

Double-sided Rowhammer

physical memory
Dedup Est Machina: Referencing the Fake Object

Double-sided Rowhammer

physical memory
Dedup Est Machina: Referencing the Fake Object

Pointer Pivoting

JavaScript Array  JavaScript Array
Dedup Est Machina: Referencing the Fake Object

Pointer Pivoting

array header

array header

array data

JavaScript Array

JavaScript Array
Dedup Est Machina: Referencing the Fake Object

Pointer Pivoting

array header ? ? array header array data

JavaScript Array JavaScript Array
Dedup Est Machina:

Can One Attack the Full System?
Dedup Est Machina: System-wide Exploitation

- Deduplication is enabled system-wide
- We can leak secrets from other processes
- Say arbitrarily long passwords
  - E.g., 30-byte password hashes in nginx
- System-wide Rowhammer is more involved
  - We don’t “own” other processes’ physical memory
- We’ll look at this in our next example
Dedup Est Machina: Impact

- We shared our MS Edge exploit with Microsoft. They addressed this issue in MS-16-093 (CVE-2016-3272) by disabling memory deduplication on Windows 10
- Disable it on legacy systems (Powershell)

  > Disable-MM Agent - PageCombining

- [https://vusec.net/projects/dedup-est-machina](https://vusec.net/projects/dedup-est-machina)
EXAMPLE 2

Bug-free Exploitation in Clouds
Flip Feng Shui

- Published at USENIX Security 2016
  - with Ben, Kaveh, Erik, Herbert, and Bart (KU Leuven)
- Much media attention

"Flip Feng Shui" Security Now! #576
An incredibly righteous and sublime hack:
Weaponizing the RowHammer attack

- System-wide exploit in public KVM clouds
  - ...without relying on a single software bug
Flip Feng Shui: Overview

Rowhammer
(hardware glitch)
Flip Feng Shui: Overview

Rowhammer
(hardware glitch)
+
Memory deduplication
(physical memory massaging primitive)
Flip Feng Shui: Overview

Rowhammer
(hardware glitch)
+
Memory deduplication
(physical memory massaging primitive)

Cross-VM compromise in public Linux/KVM clouds without software bugs
Flip Feng Shui: Attacker’s Goals

Virtualization Host

Linux/KVM

Victim

Attacker

KSM: cross-VM memory deduplication

Backing memory
Flip Feng Shui: Attacker’s Goals

Virtualization Host

Linux/KVM

Victim

Attacker

Target sensitive memory page in victim VM’s memory

Backing memory
Flip Feng Shui: Attacker’s Goals

Virtualization Host
Linux/KVM

Victim
Attacker

Corrupt sensitive page to subvert victim VM

Backing memory
Flip Feng Shui: Mechanics

Step 1:

The attacker needs to find a vulnerable physical page to flip bits at a given sensitive offset
Flip Feng Shui: Templating

physical memory

attacker memory

victim memory
Flip Feng Shui: Templating

physical memory

attacker memory

victim memory
Flip Feng Shui: Templating

physical memory

attacker memory

victim memory
Flip Feng Shui: Templating

physical memory

attacker memory

victim memory
Flip Feng Shui: Templating

physical memory

attacker memory

victim memory
Flip Feng Shui:
Mechanics

Step 2:

The attacker needs to force the system to map the victim page into the vulnerable template
Flip Feng Shui: Physical Memory Massaging

physical memory

attacker memory

victim memory
Flip Feng Shui: Physical Memory Massaging

physical memory

attacker memory

victim memory
Flip Feng Shui: Physical Memory Massaging

physical memory

attacker memory

victim memory
Flip Feng Shui: Physical Memory Massaging

physical memory

attacker memory

victim memory
Step 3:

The attacker needs to flip the bit at the sensitive offset in the vulnerable template.
Flip Feng Shui: Exploitation

- Physical memory
- Attacker memory
- Victim memory
Flip Feng Shui: Exploitation

physical memory

attacker memory

victim memory
**Flip Feng Shui:**
Exploitation

<table>
<thead>
<tr>
<th>Physical Memory</th>
<th>Attacker Memory</th>
<th>Victim Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Flip Feng Shui: Finding a Victim Page

The attacker wants a *victim page*:

- containing security-sensitive data
  - Corruption should result in cross-VM compromise
- with predictable content
  - For memory deduplication to map it into attacker VM
- with ideally many sensitive offsets
  - Easier to find useful templates
Flip Feng Shui: Finding a Victim Page

How about **public cryptographic keys**?

- Public keys are not secret, thus predictable
- Arbitrary corruption weakens their security
Flip Feng Shui: OpenSSH Attack

How about **public cryptographic keys**?
- Public keys are not secret, thus predictable
- Arbitrary corruption weakens their security

Target OpenSSH’s `~/.ssh/authorized_keys` to SSH to victim VM and login as administrator

```bash
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQBDMUensMjWvw+d4SLKCVcP0MR3n2PsSohXBRow/qOcUXB8NFH1bWXUORC/uSPnAnWH1QYeuIP5UNnkBXWpDGgjmWTbrUfA4tqWllBBwjii4qIUWcBGqlldBUqvWsWbZ86/NY2fsKLtLDkkleFhcJmNFXnYkRs3J21BGs7JdUnd9ue0x2Nk/aSp2GODzAXwDPhwQNW4LQ8/xZTkml5DjqIAXpqa+aqTMDKNItoi/TVLoR/7BqgVslt3tbg2mew4IsmUFQMCwKdxBk5TxAgAjCmwmh+gRt0/tb6tDKzvVCNcHc4968VPXJYK2+Hr/RdYoYSLoIV/DQcTIyYZhUV5v test@source
```
Flip Feng Shui: OpenSSH Attack
Flip Feng Shui: OpenSSH Attack

Virtualization Host

Linux/KVM

Victim

Attacker

Backing memory

Attempt SSH connection
Check `authorized_keys`

Virtualization Host

Linux/KVM

Victim

Attacker

Attempt SSH connection

Backing memory
Flip Feng Shui: OpenSSH Attack

Virtualization Host

Linux/KVM

Victim

Attacker

Backing memory

Craft victim page content in vulnerable template
Flip Feng Shui: OpenSSH Attack

Virtualization Host

Linux/KVM

Victim

Attacker

Dedup moves the victim page to the vulnerable template

Backling memory
Flip Feng Shui: OpenSSH Attack

Virtualization Host
Linux/KVM

Victim
Attacker

It’s Hammer time!

Backing memory
Flip Feng Shui: OpenSSH Attack

Changes are reflected in the victim page

Virtualization Host

Linux/KVM

Victim

Attacker

It’s Hammer time!

Backing memory
Flip Feng Shui: OpenSSH Attack

A bit flip in a **public RSA key**...
- Results in a weak key one can factorize
- Easy to reconstruct the new private key
- We do this in minutes and login to the VM!

```bash
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMUensMjWvw+d4SLKCVcP0MR3n2PsSohXBroW/qOcUXB8NFHlbWXUORC/uSPnAnWH1QYeuIP5UNnkBXWpDGgjmWTbrUfA4tgW1BBwji4qIUIWCBGql1dBVqWsWbZ86/NY2fsKLTtLDkk1eFhcJmNFXnYkRs3J21BGS7JdUnDd9ue0x2Nk/aSp2G0DzAXwDphwQNw4LQ8/xZTk5DjqIAAXBpa+qaqTMdKNItOi/IVLoR/7BqgVslt3tbqzmeaw4IsmUFQMCwKdxBk5TxAagAjCmwnh+gRt0/tb6tDkzvVCNhC4968VPXJYK2+Hr/RdYIoYSLoIV/DQcTIyyYzhUV5v test@source
```
Flip Feng Shui: Impact

- Notified:
  - Red Hat, Oracle, Xen, VMware, Debian, Ubuntu, OpenSSH, GnuPG, hosting companies

- GnuPG “included hw bit flips in their threat model”

- https://vusec.net/projects/flip-feng-shui
“Can we just disable memory deduplication and buy better DRAM?”

Yes, you really should, but...
Mitigations

No dedup and want phys mem massaging?

- Enter **Drammer** [CCS ‘16]:
  - Deterministic RH root exploit on ARM/Android
  - User-level DMA mem to flip bits and massage mem

  “Android Security Reward”

- More recently:
  - JS: **Glitch** [S&P ‘18]
  - Remote: **Throwhammer**

- [https://vusec.net/projects/drammer](https://vusec.net/projects/drammer)
Mitigations

No dedup and want phys mem massaging?

- **Enter Drammer [CCS ‘16]:**
  - Deterministic RH root exploit on ARM/Android
  - User-level DMA mem to flip bits and massage mem

- **Physical memory massaging approach:**
Mitigations

No dedup and want phys mem massaging?

- Enter **Drammer** [CCS ‘16]:
  - Deterministic RH root exploit on ARM/Android
  - User-level DMA mem to flip bits and massage mem

- Physical memory massaging approach:

  Exhaust all physical memory
Mitigations

No dedup and want phys mem massaging?

- Enter Drammer [CCS ‘16]:
  - Deterministic RH root exploit on ARM/Android
  - User-level DMA mem to flip bits and massage mem

- Physical memory massaging approach:
  - Release vulnerable page
Mitigations

No dedup and want phys mem massaging?

- Enter Drammer [CCS ‘16]:
  - Deterministic RH root exploit on ARM/Android
  - User-level DMA mem to flip bits and massage mem

- Physical memory massaging approach:

  Trigger allocation page table page
Mitigations

No dedup and want to leak pointers?

- **Enter AnC [NDSS ‘17]:**
  - Leak heap and code pointers from JS in seconds
  - Cache attack on the MMU

- 4 CVEs (Intel, AMD, ARM)
- Tested on 22 microarchs
- Apple: WebKit mitigation

- [https://vusec.net/projects/anc](https://vusec.net/projects/anc)
Mitigations

No Rowhammer?

- Not so fast
  - Rowhammer exploits fundamental DRAM properties
- Originally on x86, we found flips on ARM
  - See *Drammer* paper
- Discovered on DDR3, still there on DDR4
  - Despite targeted hardware countermeasures
...What About ECC?
Mitigations

ECC memory?

● Enter ECCploit [S&P ‘19]:
  ○ Reversed commodity (SECDED-like) ECC functions
  ○ First reliable ECC-aware Rowhammer exploit

● Approach:
  ○ Side-channel corrections
  ○ Induce individual bit flips via Rowhammer
  ○ Combine flips to bypass reversed ECC functions

● https://vusec.net/projects/eccploit
Mitigations

No dedup and no Rowhammer?
- Other primitives will come along (e.g., AnC)

Expect:
- More hw/sw properties you didn’t know about
- More side channels (Meltdown, Spectre…)
- More hardware glitches (SSD bit flips…)
- A radical change in the way we think about sys security and “reasonable” threat models
Mitigations

Recipe moving forward:

- More “open” hardware
- Systems-level solutions (hw, sw, hw+sw)

Some examples:

- VUSion [SOSP’17]
  - Secure, constant-time memory deduplication
- ZebRAM [OSDI’18]
  - OS design for comprehensive Rowhammer protection
Conclusion

- Software security is getting better
- But hw and sw are getting extremely complex
- Potentially huge unexplored attack surface
- Attackers can subvert even “perfect” software
- Need a deep understanding of hw/sw stack

VUVec

https://vusec.net