What if software had no bugs?
2010

Security problems are caused by
  – Software bugs, and
  – Configuration bugs

Impossible to write software without bugs
Even if the software is perfect—and well-configured, it is still vulnerable!

What does that mean for formally verified systems?
Credits

Erik Bosman
Ben Gras
Kaveh Razavi
Victor van der Veen
Cristiano Giuffrida

https://vusec.net
BinArmor (USENIX ATC '12)
ASLR3 (USENIX Sec '12)
ShrinkWrap (ACSAC '15)
StackArmor (NDSS '15)
PathArmor (CCS '15)
TypeArmor (S&P '16)
MvArmor (DSN '16)
CodeArmor (EuroS&P'16)
APM (USENIX Sec '16)
VTPin (ACSAC '16)
TypeSan (CCS '16)

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)

Defenses

Attacks

https://vusec.net
Software Exploitation:

2010
Software Exploitation:

2010

Bugs, Bugs Everywhere!
Software Exploitation:

2010

Attacker Exploits Vulnerable Software
Software Exploitation:

2010

Attacker Owns Application
Software Exploitation:

2010

Attacker Owns System
Software Exploitation: 2010

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
Software Exploitation:

2016

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

2016

Memory R: Hw/Sw Side Channels
Software Exploitation:

2016

Memory W: Hardware Glitches
Software Exploitation:

2016

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

2016

Memory R/W: Back to Reliable Exploits
Even if the software is perfect...
...with no bugs, well-configured, and latest defenses
...it is still vulnerable!

Attackers abuse properties of modern hw and sw for reliable exploitation

We’ll look at 2 examples (browsers, clouds) with 2 properties (dedup, Rowhammer)
EXAMPLE 1

Bug-free Exploitation in Browsers
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)
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)
Memory deduplication
Leak randomized heap and code pointers
Dedup Est Machina: Overview

Memory deduplication
Leak randomized heap and code pointers

JavaScript Array

+0.0
+3.141592
42.
1
NaN

chakra.dll
Dedup Est Machina: Overview

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

+ Rowhammer
Create a reference to our fake object
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

Now also 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**
  - 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

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?
Memory Deduplication: Side-channel Leaks

Very coarse-grained. Still interesting?
Is user running software X?

Skype not running
Memory Deduplication: Side-channel Leaks

Very **coarse-grained**. Still interesting?

Is user running software X?

Skype running
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?”
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
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
Dedup Est Machina: Primitives

Primitive #1: Alignment Probing
Memory deduplication
Leak randomized heap and code pointers
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

```
mov RCX, 0x1c20
mov RAX, [code address]
```
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
Memory deduplication
Leak randomized heap and code pointers

JavaScript Array
+0.0
+3.141592
42.
1
NaN

chakra.dll
Heap pointers are word aligned
Alignment probing won’t cut it, same for primitive #2

Time for primitive #3!

“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!
Dedup Est Machina: Birthday Paradox
Dedup Est Machina: Birthday Paradox
Primitive #3: Birthday Heapspray

physical memory

attacker memory

victim memory
Primitive #3: Birthday Heapspray

physical memory

attacker memory

victim memory
Primitive #3: Birthday Heapspray

physical memory

attacker memory

victim memory
Primitive #3: Birthday Heapspray

Physical memory

Attacker memory

Victim memory
Primitive #3: Birthday Heapspray

physical memory

attacker memory

victim memory
Primitive #3: Birthday Heapspray

physical memory

attacker memory

victim memory
Primitive #3: Birthday Heapspray
Primitive #3: Birthday Heapspray

physical memory

attacker memory

victim memory
<table>
<thead>
<tr>
<th>Physical Memory</th>
<th>Attacker Memory</th>
<th>Victim Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Physical Memory Grid" /></td>
<td><img src="image2" alt="Attacker Memory Grid" /></td>
<td><img src="image3" alt="Victim Memory Grid" /></td>
</tr>
</tbody>
</table>

**Primitive #3:** Birthday Heapspray
Primitive #3: Birthday Heapspray

physical memory

attacker memory

victim memory
Primitive #3: Birthday Heapspray

physical memory

attacker memory

victim memory
Dedup Est Machina: Leaking Heap Pointer (#3)

Creating Secret Pages

1M Aligned objects
Creating Secret Pages

Dedup Est Machina: Leaking Heap Pointer (#3)

array data

1M Aligned objects
Dedup Est Machina: Leaking Heap Pointer (#3)

Creating Secret Pages

page
page
page
page

1M Aligned objects
Dedup Est Machina: Leaking Heap Pointer (#3)

Creating Secret Pages
Creating Secret Pages

Dedup Est Machina: Leaking Heap Pointer (#3)
Creating Probe Pages
Dedup Est Machina: Leaking Heap Pointer (#3)

Creating Probe Pages

guessed aligned addresses, 128M apart

???
???
???
???
???
???
???

???

typed array data
Creating Probe Pages

guessed aligned addresses, 128M apart

Dedup Est Machina: Leaking Heap Pointer (#3)
Birthday Heapspray

Dedup Est Machina:
Leaking Heap Pointer (#3)
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)

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

probe pages (guessed addresses)

+128M, +128M, +128M, ...
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
Memory deduplication
Leak randomized heap and code pointers
Create a fake JavaScript object

Rowhammer
Create a reference to our fake object
Dedup Est Machina: Creating a Fake Object

Fake JavaScript Uint8Array

JavaScript Array

JavaScript Array
Dedup Est Machina: Creating a Fake Object

Fake JavaScript Uint8Array

array header ? ? array header

array data

JavaScript Array JavaScript Array
Dedup Est Machina: Creating a Fake Object

Pointer Pivoting

JavaScript Array  Array header

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

Rowhammer

rows

DDR memory
Dedup Est Machina: Referencing the Fake Object

Rowhammer

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

Double-sided Rowhammer

DDR memory
Dedup Est Machina: Referencing the Fake Object

Double-sided Rowhammer

<table>
<thead>
<tr>
<th>physical memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Dedup Est Machina: Referencing the Fake Object

Double-sided Rowhammer

physical memory
Double-sided Rowhammer

physical memory
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

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?
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
We shared our MS Edge exploit with Microsoft and they addressed it in MS-16-093, July 18th (CVE-2016-3272) by temporarily disabling memory deduplication on Windows 10.

Disable it on legacy systems (Powershell): 

> Disable-MMAgent -PageCombining
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 exploits 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

Backing memory

Target sensitive memory page in victim VM’s 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: Probabilistic Rowhammering

Double-sided Rowhammer

physical memory
Flip Feng Shui: Probabilistic Rowhammering

Seaborn’s Attack

physical memory

sprayed page tables
Step 1:

The attacker needs to find a vulnerable physical page to flip bits at a given sensitive offset.
<table>
<thead>
<tr>
<th>Physical Memory</th>
<th>Attacker Memory</th>
<th>Victim Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Physical Memory Grid" /></td>
<td><img src="image2.png" alt="Attacker Memory Grid" /></td>
<td><img src="image3.png" alt="Victim Memory Grid" /></td>
</tr>
</tbody>
</table>
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
Flip Feng Shui: Mechanics

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

physical memory

attacker memory

victim memory
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
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

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMUensMjWvw+d4SLKCVCcP0MR3n2PsSohXBrow/qOcUXB8NFH1bWXUORC/uSPnAnWH1QYeUIp5UNnkBXWpDGgjmWTbrUfA4tqW1BBwji4qIUWcBGqlldBUvqWbZ86/NY2fsKltLDkkleFhcJmNFXnYkRs3J21BGS7JdUnDd9ue0x2Nk/aSp2GODzAXwDPhwQNw4LQ8/xZTkn5DjqIAAXBpa+qaqTMdKNItOi/IVLoR/7BqgVs1t3tbgZmew4IsmUFQMCwKdxBk5TxAgAjCmwmh+gRt0/tb6tDKzvVCNcHc4968VPXJYK2+Hr/RdYloYSLoIV/DQcTlyyYzhUV5v test@source
```
Flip Feng Shui: OpenSSH Attack

Virtualization Host

Linux/KVM

Victim

Attacker

Backing memory
Flip Feng Shui: OpenSSH Attack

Virtualization Host
Linux/KVM
Victim
Attacker

Attempt SSH connection

Backing memory
Flip Feng Shui: OpenSSH Attack

Virtualization Host
Linux/KVM

Victim
Attacker

Check authorized_keys

Attempt SSH connection

Backing memory
Flip Feng Shui: OpenSSH Attack

Virtualization Host

Linux/KVM

Craft victim page content in vulnerable template

Victim
Attacker

Backing memory
Flip Feng Shui: OpenSSH Attack

Virtualization Host
Linux/KVM

Victim
Attacker

Dedup moves the victim page to the vulnerable template

Backing memory
Flip Feng Shui: OpenSSH Attack

Virtualization Host

Linux/KVM

Victim
Attacker

Backing memory

Hammer time!
Flip Feng Shui: OpenSSH Attack

Changes are reflected in the victim page

Virtualization Host
Linux/KVM

Victim
Attacker

Hammer time!

Backing memory
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!

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMUensMjWvw+d4SLKCVCcP0MR
3n2PsSohXBroW/qOcUXB8NFHlbWXUORC/uSPnAnWH1QYeIP5UNnkBXWpDGgjm
WTbrUfa4tqW1BBwjii4qIUWCBGq1ldBUvqWsWhZ86/NY2fsKLtLDkk1eFhcJmN
FXnYkRs3J21BGs7JdUnDd9ue0x2Nk/aSp2GODzAxwDPhwQNw4LQ8/xZTkn5Djq
IAAXBpa+qaqTMdKNItoi/ILoR/7BqgVs1t3tbg2mew4IsmUFQMCwKdxBk5TxA
agAjCmwmh+gRto/tb6tDKzvVCNhC4968VPXJYK2+Hr/RdYloYSLoTV/DQcTly
yYzhUV5v test@source
```
Flip Feng Shui: OpenSSH Attack

CDF

Attack time (mins)
“What if we don’t know the public key(s) of the administrator?”
Flip Feng Shui: apt-get Attack

Wait for apt-get update on Ubuntu or Debian victim VM

Virtualization Host
Linux/KVM

Victim
Attacker

Backing memory
Flip Feng Shui: 
apt-get Attack

Check sources.list

debian.org
ubuntu.com
...

Virtualization Host
Linux/KVM

Victim

Attacker

Wait for apt-get update on Ubuntu or Debian victim VM

Backing memory
Flip Feng Shui: apt-get Attack

Virtualization Host

Linux/KVM

Corrupt URLs in sources.list

Victim
Attacker

Backing memory
With a bit flip in a mirror domain name...
The victim VM installs our own packages from:

ubun\textbf{v}u.com
\textbf{uc}untu.com

\ldots

(which we own)

But fortunately, the packages are signed!
Wait...
Flip Feng Shui: apt-get Attack

We can:

Flip a bit in trusted.gpg
  where apt-get’s trusted package public keys are stored
Generate the new corresponding private key
  Again, we can do this in minutes
Sign our own packages
  Say from ubunvu.com
Install & run anything we want in the victim VM
Flip Feng Shui: Impact

Notified:

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

NCSC did all the hard work, thanks!

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

“Can we just disable memory deduplication and buy better DRAM?”

Yes, you really should, but...
Mitigations

No dedup?

Need another memory massaging primitive
E.g., just exploit predictable memory reuse patterns in common page allocators

Basic approach:

- Fill physical memory with attacker-allocated pages
- Find a vulnerable template
- Release corresponding physical page to allocator
- Trigger allocation of victim page
- The allocator has only 1 option to fulfill the allocation
Better DRAM?
Not so fast
  Rowhammer exploits fundamental DRAM properties
Discovered on DDR3, still there on DDR4
  Despite targeted countermeasures
Originally on x86, we found flips on ARM
  See our upcoming *Drammer* CCS’16 paper
ECC memory is not a panacea
  Not cheap/widespread, can’t fix all bit flips
Mitigations

No dedup and no Rowhammer?
Other primitives will come along

Expect:
More hw/sw properties you didn’t know about
More side channels
More hardware glitches
A radical change in the way we think about sys security and “reasonable” threat models
Flip Feng Shui:

Is Physics Part of Your Threat Model Yet?
[Aug 4, 12:00] **Microsoft**: “Thanks to our mitigation improvements, since releasing Edge one year ago, there have been no zero day exploits targeting Edge”
[Aug 4, 12:00] Microsoft: “Thanks to our mitigation improvements, since releasing Edge one year ago, there have been no zero day exploits targeting Edge”

[Aug 4, 17:00] VUSec: “Dedup Est Machina: One can exploit the latest Microsoft Edge with all the defenses up, even in absence of software/configuration bugs”
Rethinking Systems
Security

Formally verified systems

Microsoft Research
@MSFRResearch

Feel better. Hacker-proof code has been confirmed. quantamagazine.org/20160920-forma ... via @KSHartnett
Formally verified systems

Microsoft Research
@MSFTResearch

Feel better. Hacker-proof code has been confirmed. quantamagazine.org/20160920-forma ...
via @KSHartnett

[Aug 10] VUSec: “Flip Feng Shui: Reliable exploitation of bug-free software systems”
What’s Next?

Start worrying about emerging new threats
Think about new security defenses
Don’t forget the past
  E.g., Anomaly detection for Rowhammer attacks
But also:
  Randomization
  Isolation
...

(now applied to physical memory)
Conclusion

Software security defenses are getting better
But hw and sw are getting extremely complex
Potentially huge unexplored attack surface
Attackers can subvert even “perfect” software
Beyond side channels (but they play a role)

https://vusec.net