Spectre/Meltdown at eCG: Rebooting 80k cores

Bruno Bompastor
Adrian Joian
Cloud Reliability Team
2018
10 Brands In Multiple Countries

NL/DE Datacenters
Spectre/Meltdown

- Meltdown: melts security boundaries which are normally enforced by the hardware

- Spectre: exploits speculative execution on modern cpus

- A malicious program can exploit Meltdown and Spectre to get hold of secrets stored in the memory of other running programs

- Spectre is harder to exploit than Meltdown, but it is also harder to mitigate

- Source: https://meltdownattack.com/
Assessment

In the Assessment phase we determined a set of packages that we needed to update.

Linux Kernel:
- Applies mitigations to speculative execution by exposing three system calls: Page Table Isolation (pti), Indirect Branch Restricted Speculation (ibrs) and Indirect Branch Prediction Barriers (ibpb)
  - https://access.redhat.com/articles/3311301

Qemu-kvm-ev:
- Patches to KVM that expose the new CPUID bits and MSRs to the virtual machines
  (https://www.qemu.org/2018/01/04/spectre/)

BIOS:
- Several microcode updates were provided by Intel but it was not clear if indeed would totally fix the vulnerability, and if it would cover all CPU versions
- BIOS was the last requirement to mitigate Spectre/Meltdown. Released on 24 Feb 2018.
Cloud Images Vulnerabilities Patches

We have rebuilt all our cloud images with the patched kernel.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>eCG fix released</th>
<th>upstream fix released</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centos7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ubuntu Bionic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ubuntu Xenial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ubuntu Trusty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debian Stretch</td>
<td></td>
<td>partially fixed</td>
</tr>
<tr>
<td>Debian Jessie</td>
<td></td>
<td>partially fixed</td>
</tr>
</tbody>
</table>
When Spectre/Meltdown vulnerabilities were unveiled it was clear that we needed to automate the process.

For that we decided to use **Ansible** as our primary tool.

Ansible has a great way to organize a group of tasks that achieve a common goal - **Ansible Roles**

- **Openstack roles**: e.g. enable-nova-compute, restore-reason-nova-compute, start-vms, stop-vms, start-vrouter-services

- **Hardware roles**: e.g. reset-idrac, restart-compute

- **Update roles**: e.g. update-os, upgrade-bios

- **Meltdown-specter-checker role**
Meltdown-specter-checker Role

- name: Check patched BIOS version

- name: Check if we have correct version of kernel installed

- name: Check if we have correct version of qemu installed on computes

- name: Get checker from repo

- name: Run the checker on the host
  shell: sh /tmp/spectre-meltdown-checker.sh --variant 1 --variant 3 --batch
  become: True
  register: result_check
- debug: msg="{{ result_check.stdout_lines }}"

Final step runs an open source script that identifies Spectre/Meltdown vulnerabilities: https://github.com/speed47/spectre-meltdown-checker
Meltdown-specter-checker Role Output

```
$ ansible-playbook -i "host.example.com," maintenance-playbooks/meltdown-specter/compute/meltdown-specter-checker.yaml

PLAY [Check if servers are vulnerable to meltdown/specter] ........................................................................
TASK [setup] ...........................................................................................................................................
  ok: [host.example.com]

TASK [Check if servers are vulnerable to meltdown/specter] ..................................................................
TASK [maintenance/meltdown-specter-checker : Check patched BIOS version] ...........................................
  ok: [host.example.com]

TASK [maintenance/meltdown-specter-checker : Check if we have correct version of kernel installed] .......
  ok: [host.example.com] => (item=kernel-3.10.0-693.21.1.el7.x86_64)

TASK [maintenance/meltdown-specter-checker : Check if we have correct version of qemu installed on computes]
  ok: [host.example.com] => (item=qemu-kvm-ov-2.9.0-16.el7_4.14.1.x86_64)

TASK [maintenance/meltdown-specter-checker : Get checker from repo] ..............................................
  ok: [host.example.com]

TASK [maintenance/meltdown-specter-checker : Run the checker on the host] ........................................
  changed: [host.example.com]

TASK [maintenance/meltdown-specter-checker : debug] ...........................................................................
  ok: [host.example.com] => {
    "msg": {
      "CVE-2017-5753: OK (Mitigation: Load fences)",
      "CVE-2017-5754: OK (Mitigation: PTI)"
    }
  }

PLAY RECAP ..............................................................................................................................................
host.example.com : ok=7    changed=1    unreachable=0    failed=0
```
Meltdown-specter-patching Playbook

Pre-tasks:
- name: 'disable compute node in monitoring'
- name: 'disable puppet'
- name: 'disable compute node in OpenStack'
- name: 'stop instances'
- name: 'zfs umount /var/lib/nova'
- name: 'Check files on /var/lib/nova'
- name: 'Check directories on /var/lib/nova'
- name: 'reset iDRAC'
- name: 'getting current bios version'

Update-tasks:
- name: 'upgrade BIOS'
- name: 'update operating system'

Post-tasks:
- name: 'reboot compute nodes'
- name: 'Check if servers are vulnerable to meltdown/specter'
- name: 'zfs mount /var/lib/nova'
- name: 'start vrouter services'
- name: 'run puppet'
- name: 'start canaries'
- name: Resolve all checks
- name: 'enable compute node in monitoring'
- name: 'start vms'
- name: 'enable compute node in OpenStack'
Services Restarted

- vRouter agent: is a contrail component that takes packets from VMs and forwards them to their destinations (manages the flows)

- Canary: small instance created in every hypervisor to provide monitoring and testing

- ZFS file system used to host virtual machines was unmounted and mounted (safety precaution)
Saving Compute Nodes and VMs State

- Need to disable compute nodes and shutdown VMs during maintenances
- No way to recover previous disabled reasons from API
- VMs started according to saved state
- Information should be stored in service accessible to all operators
BIOS upgrade

- Most error-prone operation in the maintenance

- Fixed most of the time by restarting out of band (OOB) system (e.g. iDRAC)

- As last resort, BIOS upgrade needed to be done manually
Hardware Failures

- Very often hardware fails after upgrade maintenance

- BIOS corrupted, no network, cpu/memory errors

- There is always risk when restarting compute nodes
Testing

- Selected platforms (group of users) tested the patched hypervisors
- We decided not to patch our full infrastructure as fast as we can
- We choose to deploy new infrastructure with these patches available wherever possible
- At the same time, we were keeping an eye on the community whenever load results were announced publicly
AVI LBaaS automation

- A Service engine is the distributed load balancer offered by Avi Networks
- Need to migrate all SEs
- Automated with AVI Ansible SDK and Python
- Started with one zone per week and ramped up to two zones on the last week
- The whole region was a success and gave us experience on automation
- Four zones from April to July
- Two patches in between
- Started with one zone per day
- Finished with one rack per day
Contrail SDN and AVI LBaaS Patch

- Contrail uses the IF-MAP protocol to distribute configuration information from the Configuration Nodes to the Control nodes

- Apply patch to avoid throwing exceptions when some link configuration already exists

- Issue with how the AVI service engines sets up the cluster interface

- AVI created a patch to fix old and new SEs creation
Performance DUS1
Performance AMS1

Hypervisor Aggregate CPU Stats

Hypervisor CPU Load
Maintenance Strategies

- Started with one zone per week
- A rack per day seems a good compromise between velocity and impact for platforms
- Notify which VMs are affected by a rack maintenance (needs automation)
- Communication on all the steps we are taking during the maintenance windows
What we have learned

- Ansible is a great tool for infrastructure automation
- Do not rush on updating as soon as the vulnerability is discovered
- Restart your whole infrastructure often to catch bugs/issues
- Scoping maintenances works best to reduce impact
Questions?