An insight into the life cycle of testing critical security updates supporting large scale security infrastructure

Vittalkumar Mirajkar
PNSQC 2021
Time crunched HotFix Test cycle

Keeping large infra secured needs
• Environment updated with regular patching / applying Hotfix [HF’s].
• The patch applied need to be reliable and seamless.

How are these HF’s tested ?
• HotFix [HF] release testing are time crunched but have to maintain test confidence of the highest order.
• However, HF for HF is reality.
The problem we wanted to address

Test and certify critical HF’s for daily release readiness for

- Multiple products
- Each product having multiple version per product active
- On all supported OS’s
Breaking down the problem

- Product set $P_n = \{P_1, P_2, \ldots, P_{12}\}$
  (Where ‘$n$’ is unique product, each having ‘$m$’ active version in field.)
- Product version($V_m$) = $V_m = \{V_1, V_2, \ldots, V_{10}\}$

Limiting production active versions to only 10,
- $P_nV_m = \{P_1V_1, P_1V_2, \ldots, P_2V_m, P_2V_1, P_2V_2, \ldots, P_2V_m, \ldots, P_{10}V_1, P_{10}V_2, \ldots, P_{10}V_m\}$

$P_nV_m$ is to be tested on supported OS’s
- $OS_n = \{OS_1, OS_2, OS_3, \ldots, OS_n\}$
Number of OS to test on

OS Set  (Taking Windows OS only)
•  $OS_n = \{OS_1, OS_2, OS_3, \ldots, OS_n\}$  (Ex: Win 7 till Win 11)

Each OSn has
•  Multiple versions (Vm)  (Ex: Win 10 has had 12+ releases)
•  Each version having Multiple flavors (Fx)  (Ex: Home, Pro, Business etc)
Our theoretical test set

- Unique Product \( P_n \), where \( n = 1 \) to 12
- Each version \( P_nV_m \), where \( m = 1 \) to 10
- Unique OS \( O_sn \), where \( n = 1 \) to 15
- OS Version released \( O_snV_m \), where \( m = 1 \) to 10
- OS Flavors active per version \( O_snV_mF_x \), where \( x = 1 \) to 4
- Total number of combinations, theoretically possible is
  - \( P_nV_mO_snV_mF_x = 12 \times 10 \times 15 \times 10 \times 4 = 72000 \)

Note
- Not all OS’s releases have same number of released field versions (examples: OS1 may have had 2 releases, whereas OS2 may have had 5 releases.)
- Not all Products have same version of releases (example: P1 had 1 to 10 patch releases, where P2 had 1 to 8)
Test set Data Sanitization

Plotting $PnVm \text{ vs } OSnVmFx$ (Max theoretical test set)

Rules for Sanitization:

- Not all product versions are supported by all OS’s
- Latest product version may not support outdated OS’s
- Installation count of older product version can be insignificant low to make a impact
Building a manageable test set

Rules for picking relevant data:

- Product Installation telemetry
- Product install counts from Product Team / Product Managers
- OS Market share
- Apply 80-20 rule
The test bed needs to

CI/CD test bed which is

- High availability
- Has 2\textsuperscript{nd} level redundancy setup
- A status monitor dashboard
- Automated Build release to test readiness infrastructure
Going beyond $2(N+1)$ redundancy

- 2 Geo Location.
- Per location: Primary & Secondary images
- Back up image
- Auto failsafe and switch over
- It’s a $2(N+N+N)$ redundancy
The execution flow

Execution flow for Patch test to release cycle

Controller Ready for Test run

Infrastructure POST and Environment remediation

Poll for Build availability for test

Trigger Test and Monitor

Trigger Re-run if required

Publish result to Controller

System Clean up and ready for next run
Step 1: Heartbeat System

- Controller
- System Heartbeat Service
- WebServer
- Email Notification Service
- SMTP Gateway
- Dashboard
- Vmware Test Machine Cluster

Heartbeat System:
- Scheduled cron job called after rollback 1 complete (1)
- Write request (2)
- Trigger email (3)
- Fetch data (4)
- Notify Failure

Step 2: Build copy system

- Controller
- Build Copy Service
- Build Server
- HTTP Long Polling

Build Copy System:
- Scheduled cron Job called after Heartbeat check
Step 3: Run Test

- Controller
- Dashboard
- E-mail client
- SMTP Gateway
- Run Test Service
- HTTP Requests
- VM Infrastructure (test bed)
- Result db
- Trigger email (3)
- Fetch data (4)
- Notify Failures

Step 4: Rollback system

- Controller
- Rollback Service
- HTTP Requests
- VM Infrastructure
- Rollback db
The Test execution controller

Test controller is built on an Active-Passive model to ensure high availability.

Active Passive Configuration
(Eg Load balancer)
The Hardware

Processing Power

Dell Blade server (Dell PowerEdge 2021) with 16 CPU rig

Storage

Dell-EMC XtremeIO (XTREMIO_X2 2020) two brick solution

VMware vCenter
Controller Submodules

The controller dashboard

- Front end which visualizes an easy-to-understand current flow of execution.
- Calls out current run state and failure (if any) and steps taken to remediate it

One of the important steps in this chain is Power on Self-Test (POST)

<table>
<thead>
<tr>
<th>Failure Points</th>
<th>Remediation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network connection</td>
<td>Network card reset followed by reboot</td>
</tr>
<tr>
<td>CPU spikes</td>
<td>Poll back followed by reboot</td>
</tr>
<tr>
<td>Hard disk availability</td>
<td>Release Hard disk using VMware api's</td>
</tr>
</tbody>
</table>
What we built

• A CI/CD setup, which takes
  • One test run : 3hrs
  • Rollback : 30mins
  • Heartbeat Check : 30mins
  • One End to End cycle : 4hrs

• Product set tested
  • Products supported : 7
  • Total Test VM’s : 250
  • OS Supported
    • Win 7 to win 10
    • Win 2k8 to 2k19
    • MacOS
    • RedHat Linux
## Test Results

### Mid run status from dashboard

<table>
<thead>
<tr>
<th>MOVE VMs</th>
<th>UPD</th>
<th>ODS</th>
<th>OFFICE</th>
<th>STATUS</th>
<th>RBS</th>
<th>Disk_Space(MB)</th>
<th>Warnings/Alerts</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012X64-MOVE49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012X64-MOVE491</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10RS264-MOVE461</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W8IP64-MOVE46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10RLX64-MOVE47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2K16-MOVE47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W8IP64-MOVE47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012RST-MOVE45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10RSX64-MOVE48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2K19864-MOVE481</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10RSX66-MOVE481</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Test completion

<table>
<thead>
<tr>
<th>MAC Machines</th>
<th>UPD</th>
<th>ODS</th>
<th>OAS</th>
<th>STATUS</th>
<th>RBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC_OS_X_MOJAVE_CORP2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC_OS_X_BIGSUR_CORP-MEDIUMDAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC_OS_X_HIGHSIERRA_CORP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC_OS_X_CATALINA_CORP-MEDIUMDAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC_OS_X_CATALINA_CONS_NE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC_OS_X_MOJAVE_MEDIUMDAT-MEDIUMDAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC_OS_X_MOJAVE_CONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC_OS_X_BIGSUR_CONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Challenges

Environment upkeeping challenges

• Relevant Product versions for testing
• Patching OS’s
• Keeping hardware running and building redundancy
Take always

Some of the areas where Rapid test-to-release model of testing can be used are,

- For quicker certification of HF for release readiness.
- For releasing minor fixes quickly to field rather than waiting for monthly release cycles.
- Incremental improvement fixes can be tested and released to production as and when feature is ready and not wait for major release cycles.
Q & A

Thank You