HCK-CI: Enabling CI for Windows guest paravirtualized drivers

Kostiantyn Kostiuk - Software engineer, Daynix

https://github.com/HCK-CI
Agenda

• virtio-win drivers
• Why do we need CI for virtio-win drivers?
• What is WHQL certification and what are HCK\HLK?
• How we can leverage WHQL certification for CI?
• What are the challenges with WHQL certification?
• Automating WHQL certification
• Examples of configuration and usage
VirtIO-Win

- Upstream: https://github.com/virtio-win/kvm-guest-drivers-windows/
- Drivers for the major virtio devices:
  - virtio-net
  - virtio-blk, virtio-scsi
  - virtio-balloon, virtio-serial, virtio-vsock, virtio-input, virtio-rng
  - virtio-gpu
Why do we need CI for virtio-win?

- Enable easy upstream contribution
- We want PRs to pass WHQL certification tests for each driver category
- Not easy to setup and execute
What are WHQL and HLK\HCK?

• Microsoft driver certification

• HLK and HCK are the test kits used for certifications
VM types

Controller

Test clients
- Device Under Test
- Support Device
Certification overview

Device certification

- Networking
- Storage
- USB
- Other categories

System certification

(SVVP for hypervisors)
Different kits

**HCK**
- Windows 7
- Windows 8
- Windows 8.1
- Windows Server 2008R2
- Windows Server 2012
- Windows Server 2012R2

**HLK**
- HLK1607
  - Windows 10, Windows Server 2016
- HLK1809
  - Windows 10, Windows Server 2019
- HLK2022
  - Windows Server 2022
Sample configuration - network

- **test bridge**
  - Device Under Test VM
  - Support Device VM
- **control bridge**
  - Controller VM
- **world connection**
Sample configuration - other devices

- DUT VM
- control bridge
- Controller VM
- world connection
What do we need?

- Run certification tests as part of CI process
- Can be used by individual developer
- The ultimate goal - to be used for actual certification
• Used by individual developer
• Orchestrates network and storage for the test
• Runs VMs on top of QEMU-KVM
What should be automated?

- Guest automation
- Guest communication layer
- Host Automation: VM and networking orchestration
- Test manager and reports
AutoHCK - architecture

- Engines
  - Guest communication layers
  - Setup managers
  - Result uploaders
- Auxiliary
• Driver installation (including test certificate)
• HCK\HLK studio control using MS API
  – Orchestration of HCK\HLK setup
  – Running tests
• Filter and playlist application
• Get back the results in a parsed human readable manner and test package
toolsHCK - examples

- createpool “NetKVM-pool”
- createproject “NetKVM-Win2022x64”
- createprojecttarget
  “PCI\VEN_1AF4&DEV_1000&SUBSYS_00011AF4&REV_00\3&267A616A&0&20” “NetKVM-Win2022x64” “CL2” “NetKVM-pool”
toolsHCK - examples

- queuetest “bbcc1b46-d0bf-46c8-85b4-2cd62df34a20” “PCI\VEN_1AF4&DEV_1000&SUBSYS_00011AF4&REV_00\3&267A616A&0&20” “NetKVM-Win2022x64” “CL2” “NetKVM-pool”

- listtestresults “bbcc1b46-d0bf-46c8-85b4-2cd62df34a20” “PCI\VEN_1AF4&DEV_1000&SUBSYS_00011AF4&REV_00\3&267A616A&0&20” “NetKVM-Win2022x64” “CL2” “NetKVM-pool”

- createprojectpackage “NetKVM-Win2022x64” [<package>]
• Communication between Controller VM and the test manager on the host using WinRM
• Communication between the Controller and Client VMs
• A gateway to toolsHCK
• Creates VMs images with corresponding software
• Orchestrates network and storage for the test
• Creates snapshots that will be used for the tests
• Runs VMs and monitors their status
Manages test configurations
Platforms, OSes, devices, HLK / custom test play lists, test black lists, HLK filter
Orchestrates test setup in HLK studio and run the tests
Monitors test process
• Result uploading
  – uploading results to DropBox
    • Parsed HLK\HCK results
    • Mini-dump files, if BSOD occurred during the tests’ run
    • Symbol and binary files for debugging
  – updates Github pull requests with CI status
• HLK test package
## Installation

### Server requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>12 x vCPUs (minimal)</td>
</tr>
<tr>
<td>Memory</td>
<td>14 GB RAM (minimal)</td>
</tr>
<tr>
<td>Storage</td>
<td>512 GB (SSD is preferred)</td>
</tr>
<tr>
<td>Host OS</td>
<td>Linux based</td>
</tr>
<tr>
<td>Internet connection</td>
<td>Recommended</td>
</tr>
</tbody>
</table>
Installation

• Project source:
  – AutoHCK
  – toolsHCK
  – HLK-Setup-Scripts
  – extra-software
Configuration

• JSON – Format
• Global configuration
• Setup managers
• Engines
Configuration

- **Global configuration**
  - **config.json**
  - **drivers.json**
  - **svvp.json**

```json
{
  "iso_path": "/home/hck-ci/HCK-CI/iso",
  "extra_software": "/home/hck-ci/HCK-CI/extra-software",
  "workspace_path": "/home/hck-ci/HCK-CI/workspace",
  "ip_segment": "192.168.0.",
  "id_range": [2, 90],
  "winrm_port": "5985",
  "repository": "virtio-win/kvm-guest-drivers-windows",
  "toolshck_path": "/toolsHCK.ps1",
  "studio_username": "Administrator",
  "studio_password": "******",
  "result_uploaders": ["dropbox"],
  "test_engine": "hcktest",
  "install_engine": "hckinstall",
  "setupmanager": "qemuhck",
  "time_out": "9"
}
```
Configuration

- Setup managers
  - qemu_machine.json
  - physhck.json
Configuration

- Engines
  - hlkinstall.json
- HLK Install
  - studio_platform.json
  - kit.json
  - iso.json
- HLK Test
  - <platform>.json

```json
{
  "name": "Win2022x64",
  "kit": "HLK2022",
  "st_image": "HLK2022.qcow2",
  "clients": {
    "c1": {
      "name": "CL1",
      "cpus": "4",
      "memory": "4G",
      "winrm_port": "4002",
      "image": "HLK2022-C1-Win2022x64.qcow2"
    },
    "c2": {
      "name": "CL2",
      "cpus": "4",
      "memory": "4G",
      "winrm_port": "4003",
      "image": "HLK2022-C2-Win2022x64.qcow2"
    }
  }
}
```
Optional software - Sentry

Sentry

Issues

All Unresolved  For Review  Ignored  Saved Searches

Search: is:unresolved

@project.close
- Ruby
  1wk ago | 3wk old

Errno: EACCES lib/project.rb in init_workspace
Permission denied @ dir_s._mkdir - /data/user/auto_heck/workspace/hcktest
- Ruby
  1wk ago | 1wk old

Showing 2 of 2 Issues

Sort by: Last Seen

Graph: 24h | Md
Events: 1
Users: 0
Assignee: 

THE LINUX FOUNDATION
Image installation

```
ruby ./bin/auto_hck install -p Win10_2004x86
```
Running sample test

```
ruby ./bin/auto_hck test
  --platform Win2019x64 --drivers ivshmem
  --driver-path "${HOME}/workspace/ivshmem-Win2019x64-build"
```
Running sample test

<table>
<thead>
<tr>
<th>Project</th>
<th>Selection</th>
<th>Tests</th>
<th>Results</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Selected</td>
<td>View Details</td>
<td>Load Playlist</td>
<td>Save Selected As Playlist</td>
<td>View By</td>
</tr>
<tr>
<td>Status</td>
<td>Test Name</td>
<td>Type</td>
<td>Length</td>
<td>Target</td>
</tr>
<tr>
<td>☑</td>
<td>DF - PNP Rebalance Request N</td>
<td>I/O</td>
<td>03m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>☑</td>
<td>DF - PNP Rebalance Request N</td>
<td>I/O</td>
<td>08m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>☑</td>
<td>DF - PNP Remove Device Test</td>
<td>I/O</td>
<td>03m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>PNP Remove Device Test</td>
<td>I/O</td>
<td>08m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>☑</td>
<td>DF - PNP Stop (Rebalance) Dev</td>
<td>I/O</td>
<td>03m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>PNP Stop (Rebalance) Dev</td>
<td>I/O</td>
<td>08m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>☑</td>
<td>DF - PNP Surprise Remove Dev</td>
<td>I/O</td>
<td>03m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>PNP Surprise Remove Dev</td>
<td>I/O</td>
<td>08m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Reboot restart with IO before</td>
<td>I/O</td>
<td>04m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Reboot restart with IO during</td>
<td>I/O</td>
<td>10m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>☑</td>
<td>DF - Reboot Restart with IO Dev</td>
<td>I/O</td>
<td>05h 00m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Reinstall with IO before</td>
<td>I/O</td>
<td>10m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>☑</td>
<td>DF - Reinstall with IO during</td>
<td>I/O</td>
<td>01h 30m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Simple - stress test with</td>
<td>I/O</td>
<td>02m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Sleep and PNP (disable an</td>
<td>I/O</td>
<td>05m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Sleep and PNP (disable an</td>
<td>I/O</td>
<td>45m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Sleep with IO Before and</td>
<td>I/O</td>
<td>04m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Sleep with IO During (Dev</td>
<td>I/O</td>
<td>05m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Sleep with IO During (Reli</td>
<td>I/O</td>
<td>45m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>HyperVisor Code Integrity</td>
<td>I/O</td>
<td>10m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Operate in Server Core Test</td>
<td>I/O</td>
<td>05m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Static Tools Logo Test</td>
<td>I/O</td>
<td>05m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>TDI filters and LSs are not allo</td>
<td>I/O</td>
<td>01m</td>
<td>VSHEM Device</td>
</tr>
<tr>
<td>DF</td>
<td>Wdf - Check Kmnd Function Tal</td>
<td>I/O</td>
<td>01m</td>
<td>VSHEM Device</td>
</tr>
</tbody>
</table>

DF - Reboot Restart with IO During (Reliability)

Run Test 22

Run Test 24
Running sample test
Projects using HCK-CI

virtio-win CI

OpenVPN CI

Some checks haven’t completed yet
4 pending and 2 successful checks

- HCK-CI/NetKVM-Win10_1703x64 Pending — Running tests (15/58): 0 tests f... Details
- HCK-CI/NetKVM-Win10_1703x86 Pending — Running tests (10/56): 0 tests f... Details
- HCK-CI/NetKVM-Win10_1809x64 Pending — Running tests (35/58): 4 tests f... Details
- HCK-CI/NetKVM-Win81x64 Pending — Running tests (14/79): 0 tests failed Details

This branch has no conflicts with the base branch when rebasing
Rebase and merge can be performed automatically.

Some checks haven’t completed yet
1 pending and 2 successful checks

- HCK-CI/openvpn-Win2019x64 Pending — Running tests (58/58): 2 tests failed Details
- HCK-CI — Build finished. Details
- continuous-integration/appveyor/pr — AppVeyor build succeeded Details

This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.
Current Status

- Running as upstream CI for several projects
- Modular architecture allowing adding support for other hypervisors or bare metal support (vDPA on Windows)
- Support for automating SVVP (hypervisor) certification tests
- Integration with GitHub (modularity allowing integration with other services)
- Integration with DropBox to upload the results
- Parsing the results so they can be viewed by people without installation of HLK\HCK Studio
Future

- Adding support for other hypervisors
- Adding support for additional results’ storage
Links – source code

- HCK-CI Framework source:
  - https://github.com/HCK-CI/AutoHCK
  - https://github.com/HCK-CI/toolsHCK
  - https://github.com/HCK-CI/HLK-Setup-Scripts
  - https://github.com/HCK-CI/extra-software
- virtio-win drivers source code (upstream CI):
  - https://github.com/virtio-win/kvm-guest-drivers-windows
- OpenVPN TAP-Windows driver source code (upstream CI):
  - https://github.com/OpenVPN/tap-windows6
Links – related information

• Windows Unattended Setup:
  – https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/automate-windows-setup

• HLK Test Reference
• KVM Forum 2012 WHQL Process for Windows Drivers and What the Community Can Learn From It - Yan Vugenfirer, Daynix:

• HLK Test Reference
  – https://www.slideshare.net/YanVugenfirer/virt-hck-kvmforum2013