Red Hat
Enterprise Linux

UKI Addons and extensions

safely extending UKIs kernel command line and initrd

Emanuele Giuseppe Esposito

February 03, 2024

Why this talk

- Extremely new stuff!

Why this talk

- Extremely new stuff!
- Not a lot of documentation

Why this talk

- Extremely new stuff!
- Not a lot of documentation
- UKIs are flexible, and security is not sacrificed

Why this talk

- Extremely new stuff!
- Not a lot of documentation
- UKIs are flexible, and security is not sacrificed
- Attempt to advertise UKIs and their features

Let’s first look at Vitaly’s slides…

https://fosdem.org/2024/events/attachments/fosdem-2024-2394-linux-on-a-confidential-vm-in-a-cloud-where-s-the-challenge-/slides/20266/slides_fosdem2024_vkuznets_k3pOduv.pdf

Vitaly's March 2023 presentation

So what is a Confidential VM?

Confidential VM provides protection from the host it runs on:

Vitaly’s March 2023 presentation

So what is a Confidential VM?

Confidential VM provides protection from the host it runs on:

- “Protection” means strong security boundary for all **data** in the VM. Malicious hypervisor or an actor having access (even privileged!) to the host should not be able to get access to the data.

Vitaly's March 2023 presentation

So what is a Confidential VM?

Confidential VM provides protection from the host it runs on:

- "Protection" means strong security boundary for all **data** in the VM. Malicious hypervisor or an actor having access (even privileged!) to the host should not be able to get access to the data.
- The host is still able to disrupt execution of the VM, e.g. it can stop it.

Vitaly's March 2023 presentation

So what is a Confidential VM?

Confidential VM provides protection from the host it runs on:

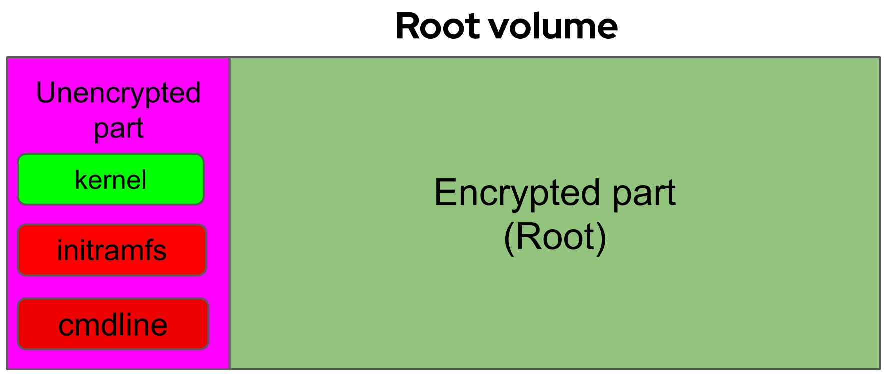
- "Protection" means strong security boundary for all **data** in the VM. Malicious hypervisor or an actor having access (even privileged!) to the host should not be able to get access to the data.
- The host is still able to disrupt execution of the VM, e.g. it can stop it.
- Hardware (AMD SEV-SNP, Intel TDX) is responsible for encrypting memory and CPU state.

Vitaly's March 2023 presentation

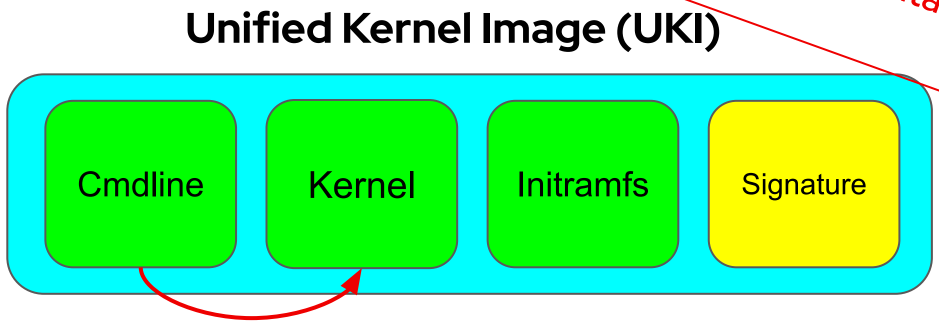
So what is a Confidential VM?

Confidential VM provides protection from the host it runs on:

- "Protection" means strong security boundary for all **data** in the VM. Malicious hypervisor or an actor having access (even privileged!) to the host should not be able to get access to the data.
- The host is still able to disrupt execution of the VM, e.g. it can stop it.
- Hardware (AMD SEV-SNP, Intel TDX) is responsible for encrypting memory and CPU state.
- Storage encryption is necessary for security and must be done by the guest OS.

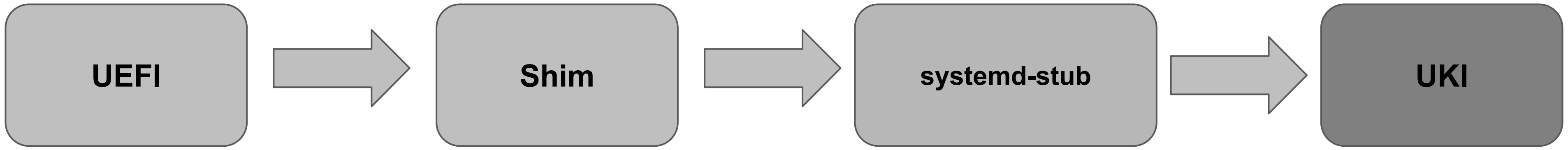


- While kernel binary is signed by Red Hat, initramfs and kernel command line are locally produced and are not signed.
- Locally produced initramfs/cmdline have unpredictable measurements.



- A single binary (UEFI application) produced **and signed** in Red Hat build system.
- The base for building UKI is systemd-stub.
- Contains vmlinuz, initramfs, and cmdline as PE sections.

Booting UKI



Vitaly's March 2023 presentation

Kernel cmdline is now immutable

- Systemd GPT auto generator (link) must be used instead of "root="
- "Limited" customization is still required:
  - "crashkernel=" like options
  - debugging, tuning options
- A mechanism to have more than one cmdline in the UKI was requested (link).
- An additional "allowlist" of options which are allowed for customization is needed.
  - E.g. the basic "root=", "init=",... can't be allowed

Requirements for UKI kernel cmdline

Requirements for UKI kernel cmdline

- It cannot be static (one fits all),
  - Production and debug options
  - Some options like “crashkernel” cannot be hardcoded and are os-dependent.

Requirements for UKI kernel cmdline

- It cannot be static (one fits all),
  - Production and debug options
  - Some options like “crashkernel” cannot be hardcoded and are os-dependent.
- Secure
  - Whoever modifies the cmdline is authenticated.
  - By default, nobody.

Red Hat

Requirements for UKI kernel cmdline

- It cannot be static (one fits all),
  - Production and debug options
  - Some options like “crashkernel” cannot be hardcoded and are os-dependent.
- Secure
  - Whoever modifies the cmdline is authenticated.
  - By default, nobody.
- Easily extensible
  - No need from RH to ship a new UKI every time cmdline changes, or have multiple UKIs with multiple cmdline

Adding kernel cmdline to an UKI

1. The embedded UKI .cmdline section

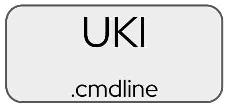
1. The embedded UKI .cmdline section

- .cmdline PE section inside the UKI, created when the UKI is generated

UKI
.cmdline

1. The embedded UKI .cmdline section

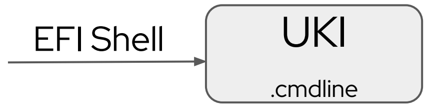
- .cmdline PE section inside the UKI, created when the UKI is generated
- Advantages
  - Secure, measured and shipped with UKI



1. The embedded UKI .cmdline section

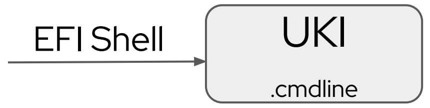
- .cmdline PE section inside the UKI, created when the UKI is generated
- Advantages
  - Secure, measured and shipped with UKI
- Disadvantages
  - Static, impossible to modify unless UKI is re-generated and shipped again

2. EFI Shell



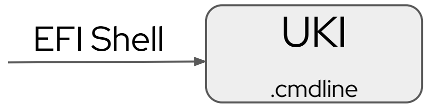
2. EFI Shell

- systemd-stub looks at this only if .cmdline is missing
  - This won’t happen in RHEL UKIs



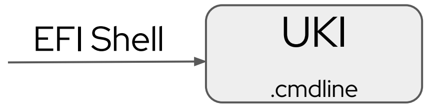
2. EFI Shell

- systemd-stub looks at this only if .cmdline is missing
  - This won’t happen in RHEL UKIs
- Advantages
  - Useful for type1 entries



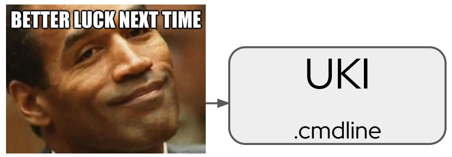
2. EFI Shell

- systemd-stub looks at this only if .cmdline is missing
  - This probably won’t happen in distro UKIs
- Advantages
  - Useful for type1 entries
- Disadvantages
  - Unsafe, an attacker can easily inject its own parameters

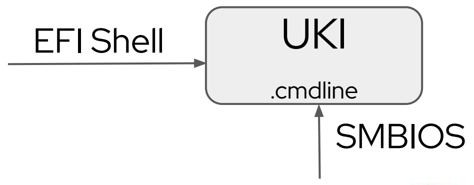


2. EFI Shell

- systemd-stub looks at this only if .cmdline is missing
  - This probably won't happen in distro UKIs
- Advantages
  - Useful for type1 entries
- Disadvantages
  - Unsafe, an attacker can easily inject its own parameters
- As a result, EFI Shell parameters are completely ignored in CVMs
  https://github.com/systemd/systemd/pull/28763

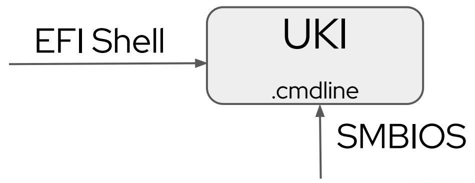


3. SMBIOS (System Management BIOS)



3. SMBIOS (System Management BIOS)

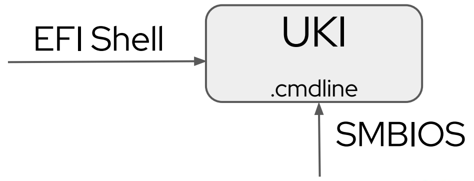
- On baremetal, SMBIOS is trusted, as it is coming from firmware/BIOS.



3. SMBIOS (System Management BIOS)

- On baremetal, SMBIOS is trusted, as it is coming from firmware/BIOS.
  - Does not apply on CVMs, as hypervisor injects it

```
# qemu-kvm […] -smbios type=11,value=io.systemd.stub.kernel-cmdline-extra=MY_CMDLINE
```

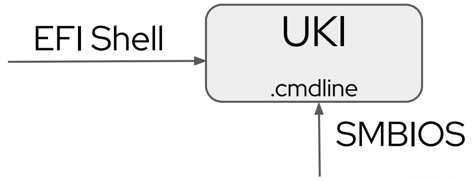


3. SMBIOS (System Management BIOS)

- On baremetal, SMBIOS is trusted, as it is coming from firmware/BIOS.
  - Does not apply on CVMs, as hypervisor injects it

```
# qemu-kvm […] -smbios type=11,value=io.systemd.stub.kernel-cmdline-extra=MY_CMDLINE
```

- Advantages
  - Useful for baremetal

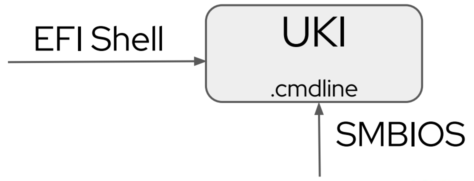


3. SMBIOS (System Management BIOS)

- On baremetal, SMBIOS is trusted, as it is coming from firmware/BIOS.
  - Does not apply on CVMs, as hypervisor injects it

```
# qemu-kvm [...] -smbios type=11,value=io.systemd.stub.kernel-cmdline-extra=MY_CMDLINE
```

- Advantages
  - Useful for baremetal
- Disadvantages
  - Unsafe, a malicious hypervisor can easily inject its own parameters

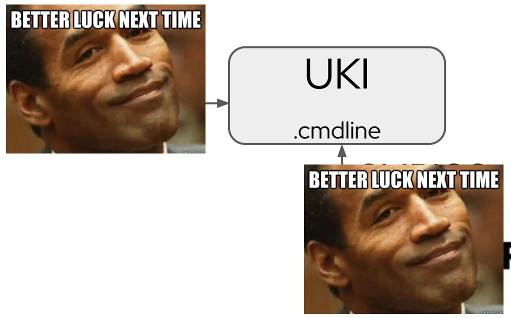


3. SMBIOS (System Management BIOS)

- On baremetal, SMBIOS is trusted, as it is coming from firmware/BIOS.
  - Does not apply on CVMs, as hypervisor injects it

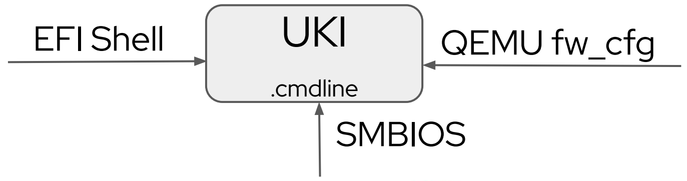
```
# qemu-kvm [...] -smbios type=11,value=io.systemd.stub.kernel-cmdline-extra=MY_CMDLINE
```

- Advantages
  - Useful for baremetal
- Disadvantages
  - Unsafe, a malicious hypervisor can easily inject its own parameters
- As a result, SMBIOS parameters are completely ignored in CVMs
  https://github.com/systemd/systemd/issues/27604



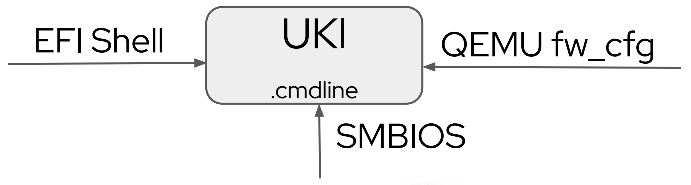
4. QEMU Firmware Configuration (fw_cfg)

- Can be used the same way as SMBIOS is
  - No baremetal, only QEMU.



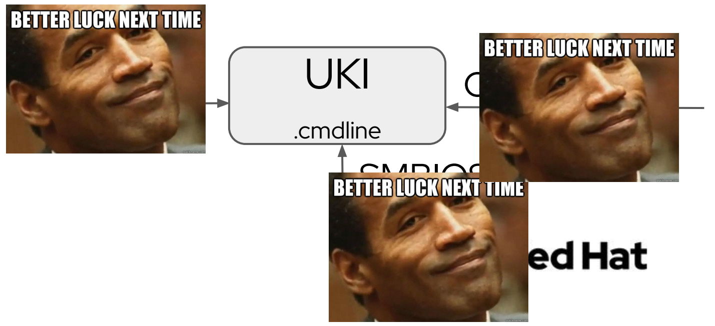
4. QEMU Firmware Configuration (fw_cfg)

- Can be used the same way as SMBIOS is
  - No baremetal, only QEMU.
- Disadvantages
  - Unsafe, a malicious hypervisor can easily inject its own parameters

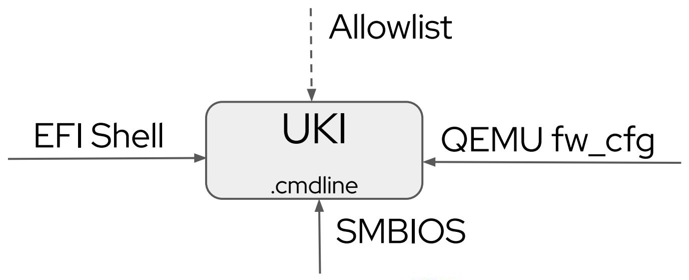


4. QEMU Firmware Configuration (fw_cfg)

- Can be used the same way as SMBIOS is
  - No baremetal, only QEMU.
- Disadvantages
  - Unsafe, a malicious hypervisor can easily inject its own parameters
- As a result, fw_cfg parameters are completely ignored in CVMs

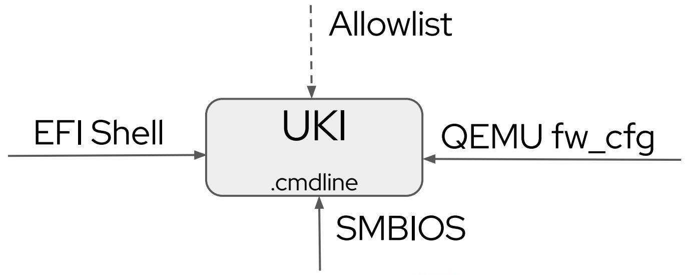


5. Our proposal: allowlist



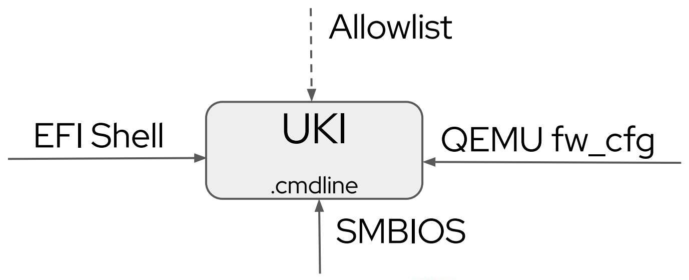
5. Our proposal: allowlist

- Create an allowlist to be stored in .allowlist PE section
  - Regex/globbing of allowed parameters (ex console=ttys*)



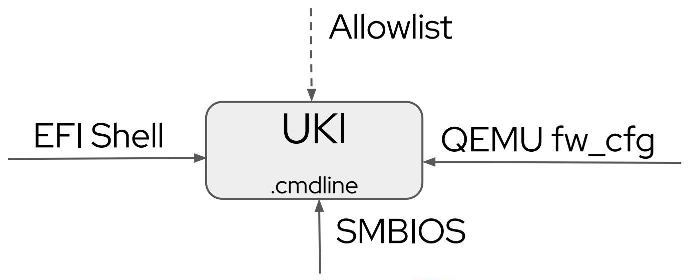
5. Our proposal: allowlist

- Create an allowlist to be stored in .allowlist PE section
  - Regex/globbing of allowed parameters (ex console=ttys*)
- Command line is coming from EFI Shell or SMBIOS



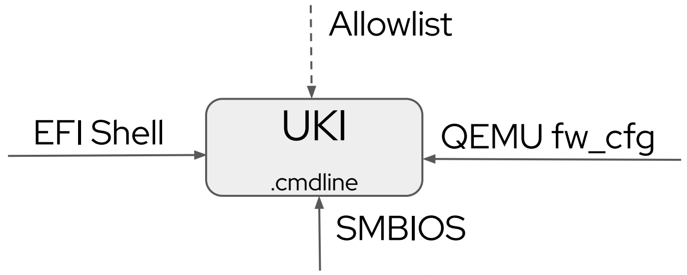
5. Our proposal: allowlist

- Create an allowlist to be stored in .allowlist PE section
  - Regex/globbing of allowed parameters (ex console=ttys*)
- Command line is coming from EFI Shell or SMBIOS
- Systemd-stub takes care of parsing the parameters against the allowlist



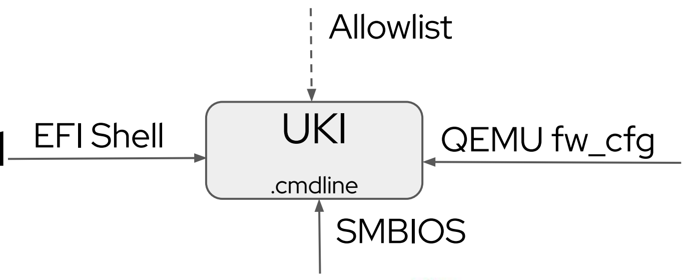
5. Our proposal: allowlist

- Create an allowlist to be stored in .allowlist PE section
  - Regex/globbing of allowed parameters (ex console=ttys*)
- Command line is coming from EFI Shell or SMBIOS
- Systemd-stub takes care of parsing the parameters against the allowlist
- Advantages
  - Allowlist takes care of rejecting unwanted parameters



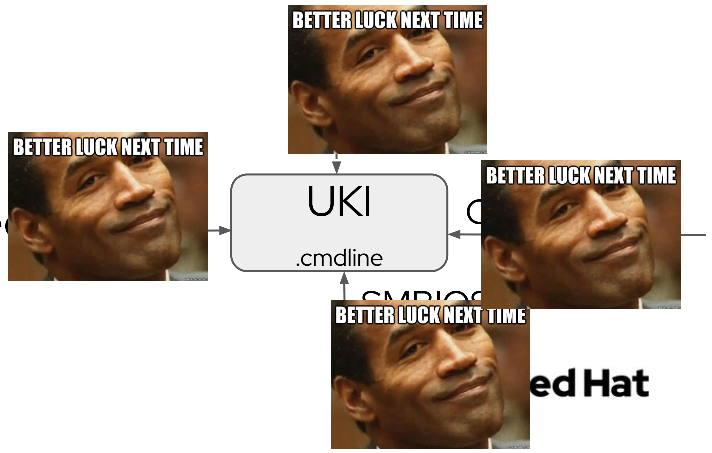
5. Our proposal: allowlist

- Create an allowlist to be stored in .allowlist PE section
  - Regex/globbing of allowed parameters (ex console=ttys*)
- Command line is coming from EFI Shell or SMBIOS
- Systemd-stub takes care of parsing the parameters against the allowlist
- Advantages
  - Allowlist takes care of rejecting unwanted parameters
- Disadvantages
  - Regex and globbing need to be very very carefully formulated

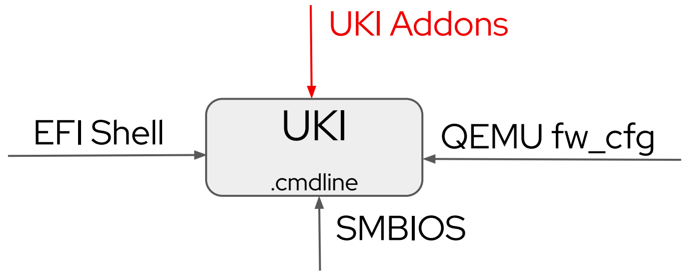


5. Our proposal: allowlist

- Create an allowlist to be stored in .allowlist PE section
  - Regex/globbing of allowed parameters (ex console=ttys*)
- Command line is coming from EFI Shell or SMBIOS
- Systemd-stub takes care of parsing the parameters against the allowlist
- Advantages
  - Allowlist takes care of rejecting unwanted parameters
- Disadvantages
  - Regex and globbing need to be very very carefully formulate
- This proposal was rejected by the systemd upstream community
- https://github.com/systemd/systemd/issues/24539

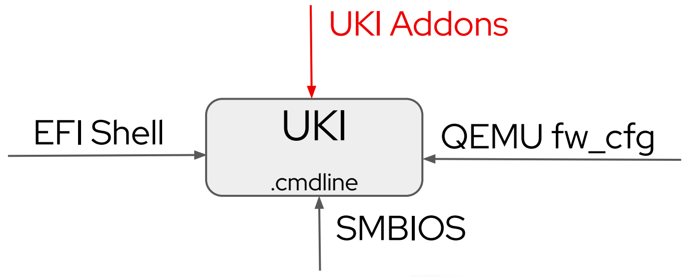


6. systemd solution: UKI addons



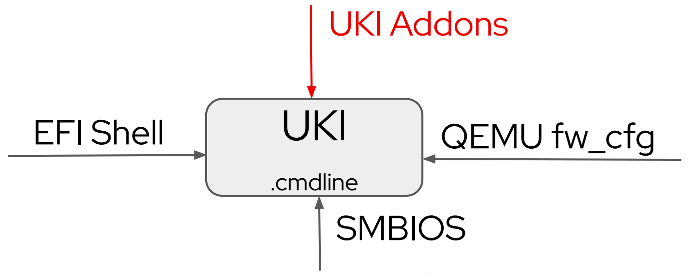
6. systemd solution: UKI addons

- Use shim shim_validate() function to validate PE signatures



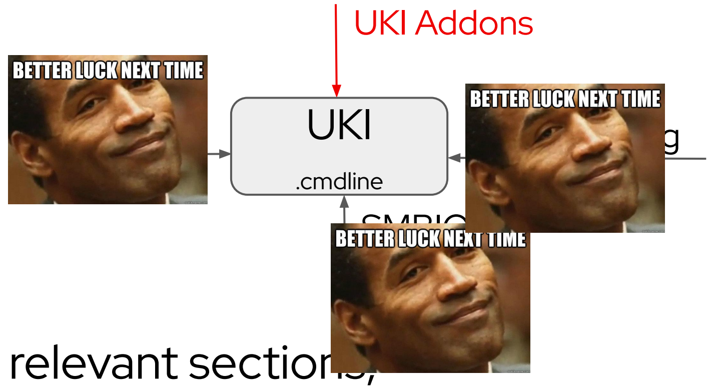
6. systemd solution: UKI addons

- Use shim shim_validate() function to validate PE signatures
- ukify: systemd tool that is able to build UKIs and much more
  https://www.freedesktop.org/software/systemd/man/ukify.html
  - Way simpler and faster than dracut and objcopy
  - Available from systemd v253, but many features in v254 and after



6. systemd solution: UKI addons

- Use shim shim_validate() function to validate PE signatures
- ukify: systemd tool that is able to build UKIs and much more
  https://www.freedesktop.org/software/systemd/man/ukify.html
  - Way simpler and faster than dracut and objcopy
  - Available from systemd v253, but many features in v254 and after
- ukify creates a PE file (addon) containing only .cmdline and other relevant sections
  - It also signs the PE with a provided key



```
/usr/lib/systemd/ukify build --signtool=pesign --secureboot-certificate-name='UKI' --cmdline='MY_CMDLINE'
--output=$BOOT/efi/EFI/Linux/my_addon.addon.efi
```

```
/usr/lib/systemd/ukify build --signtool=sbsign --secureboot-private-key=private.key
–secureboot-certificate=certificate.crt --cmdline='MY_CMDLINE'
--output=$BOOT/efi/EFI/Linux/my_addon.addon.efi
```

UKI addons

Workflow

Workflow

1. ukify creates the addon

Workflow

1. ukify creates the addon
2. Addon is put in a specific location in the ESP

Workflow

1. ukify creates the addon
2. Addon is put in a specific location in the ESP
3. systemd-stub looks for addons, finds them

Workflow

1. ukify creates the addon
2. Addon is put in a specific location in the ESP
3. systemd-stub looks for addons, finds them
4. systemd-stub calls shim_verify() on the addon

Workflow

1. ukify creates the addon
2. Addon is put in a specific location in the ESP
3. systemd-stub looks for addons, finds them
4. systemd-stub calls shim_verify() on the addon
5. Shim verifies the addon

Workflow

1. ukify creates the addon
2. Addon is put in a specific location in the ESP
3. systemd-stub looks for addons, finds them
4. systemd-stub calls shim_verify() on the addon
5. Shim verifies the addon
6. If validation is successful, systemd-stub reads the addon

Workflow

1. ukify creates the addon
2. Addon is put in a specific location in the ESP
3. systemd-stub looks for addons, finds them
4. systemd-stub calls shim_verify() on the addon
5. Shim verifies the addon
6. If validation is successful, systemd-stub reads the addon
7. systemd-stub gets .cmdline and appends it to UKI .cmdline section

Workflow

1. ukify creates the addon
2. Addon is put in a specific location in the ESP
3. systemd-stub looks for addons, finds them
4. systemd-stub calls shim_verify() on the addon
5. Shim verifies the addon
6. If validation is successful, systemd-stub reads the addon
7. systemd-stub gets .cmdline and appends it to UKI .cmdline section
8. Provide the final cmdline to the vmlinuz contained in .linux

Global and local addons

Global and local addons

- Global addons: applied to **all** installed UKIs
  - $BOOT/efi/loader/addons

Global and local addons

- Global addons: applied to **all** installed UKIs
  - $BOOT/efi/loader/addons
- UKI-specific addons: applied to the specific UKI
  - Example: 'UKI_devel' installed as $BOOT/efi/EFI/Linux/devel.efi
  - → all UKI_devel specific addons are installed in $BOOT/efi/EFI/linux/devel.efi.extra.d/

Naming conventions

Naming conventions

- UKIs are always located in $BOOT/efi/EFI/Linux/

Naming conventions

- UKIs are always located in $BOOT/efi/EFI/Linux/
- UKIs are always ending with .efi

Naming conventions

- UKIs are always located in $BOOT/efi/EFI/Linux/
- UKIs are always ending with .efi
- UKI addons always end with .addon.efi

Naming conventions

- UKIs are always located in $BOOT/efi/EFI/Linux/
- UKIs are always ending with .efi
- UKI addons always end with .addon.efi
- UKI-specific addons are always located in $BOOT/efi/EFI/Linux/<UKI_NAME>.efi.extra.d/

SBAT (Secure Boot Advanced Targeting)

https://github.com/rhboot/shim/blob/main/SBAT.md
https://github.com/rhboot/shim/blob/main/SBAT.example.md

Problem: an UKI/addon has a security issue

- Imagine the UKI/addon is issued and signed by a company like Red Hat

Problem: an UKI/addon has a security issue

- Imagine the UKI/addon is issued and signed by a company like Red Hat
- Solution 1: change certificate
  - Means invalidating all TPM measurements
  - Invalidates all other UKIs and addons
  - Impractical

Problem: an UKI/addon has a security issue

- Imagine the UKI/addon is issued and signed by a company like Red Hat
- Solution 1: change certificate
  - Means invalidating all TPM measurements
  - Invalidates all other UKIs and addons
  - Impractical
- Solution 2: add the hash of the addon to some Cloud provider blacklist

Problem: an UKI/addon has a security issue

- Imagine the UKI/addon is issued and signed by a company like Red Hat
- Solution 1: change certificate
  - Means invalidating all TPM measurements
  - Invalidates all other UKIs and addons
  - Impractical
- Solution 2: add the hash of the addon to some Cloud provider blacklist
- Solution 3: at attestation time, check if the addon with a specific hash is being measured. If so, reject it.

Problem: an UKI/addon has a security issue

- Solution 4: SBAT rules
  - Add a .sbat version “component,generation,vendor,pkg,pkg_version,url”
  - Shim checks its own sbat “component,generation” tuple with addon .sbat, if there is a match and shim generation is higher than generation, ignore the addon

UKI addons: workflow

1. ukify creates the addon
2. Addon is put in a specific location in the ESP
3. systemd-stub looks for addons, finds them
4. systemd-stub calls shim_verify() on the addon
5. Shim verifies the addon and checks SBAT component and generation
6. If validation is successful, systemd-stub reads the addon
7. systemd-stub gets .cmdline and appends it to UKI .cmdline section
8. Provide the final cmdline to the vmlinuz contained in .linux

SBAT example 1

Guest SBAT variable:

```
sbat,1
my_addon,2
```

Addon .sbat section:

```
sbat,1,SBAT Version,sbat,1,https://github.com/rhboot/shim/blob/main/SBAT.md
my_addon,2,My addon version, addon_product, 2.0, www.mycompany.com
```

SBAT example 1

Guest SBAT variable:

```
sbat,1
my_addon,2
```

Addon .sbat section:

```
sbat,1,SBAT Version,sbat,1,https://github.com/rhboot/shim/blob/main/SBAT.md
my_addon,2,My addon version, addon_product, 2.0, www.mycompany.com
```

✓

SBAT example 2

Guest SBAT variable:

```
sbat,1
my_addon,2
```

Addon .sbat section:

```
sbat,1,SBAT Version,sbat,1,https://github.com/rhboot/shim/blob/main/SBAT.md
my_addon,1,My addon version, addon_product, 2.0, www.mycompany.com
```

SBAT example 2

Guest SBAT variable:

```
sbat,1
my_addon,2
```

Addon .sbat section:

```
sbat,1,SBAT Version,sbat,1,https://github.com/rhboot/shim/blob/main/SBAT.md
my_addon,1,My addon version, addon_product, 2.0, www.mycompany.com
```

Open Problem: combining addons

- What if UKI+addonA is valid, UKI+ addonB, but UKI + addonA + addonB creates security issues
  - Couldn’t come up with a concrete example yet
  - Only solution would be to use attestation and see if addonA and addonB are measured, and if so reject the verification

Systemd-sysext initrd addons

https://www.freedesktop.org/software/systemd/man/latest/systemd-sysext.html

Extend initrd too

- To extend initramfs or even extend the host fs.
  “A system extension image extend the base system with an overlay containing additional files.”

Extend initrd too

- To extend initramfs or even extend the host fs.
  “A system extension image extend the base system with an overlay containing additional files.”

- Same concept, different tools
  - System extension image vs PE file
  - mkosi vs ukify to create a signed sysext (requires dm-verity loaded)
  - Same path where to put it

Extend initrd too

- To extend initramfs or even extend the host fs
  “A system extension image extend the base system with an overlay containing additional files.”

- Same concept, different tools
  - System extension image vs PE file
  - mkosi vs ukify to create a signed sysext (requires dm-verity loaded)
  - Same path where to put it

- Create sysext extension

Extend initrd too

- To extend initramfs or even extend the host fs
  “A system extension image extend the base system with an overlay containing additional files.”

- Same concept, different tools
  - System extension image vs PE file
  - mkosi vs ukify to create a signed sysext (requires dm-verity loaded)
  - Same path where to put it

- Create sysext extension
- Put it into $BOOT/efi/EFI/Linux/$UKI.efi.extra.d (must be a .raw)

Extend initrd too

- To extend initramfs or even extend the host fs.
  “A system extension image extend the base system with an overlay containing additional files.”

- Same concept, different tools
  - System extension image vs PE file
  - mkosi vs ukify to create a signed sysext (requires dm-verity loaded)
  - Same path where to put it

- Create sysext extension
- Put it into $BOOT/efi/EFI/Linux/$UKI.efi.extra.d (must be a .raw)
- systemd-stub will take care of copying it into initrd’s /.extra/sysext/ folder

Extend initrd too

- To extend initramfs or even extend the host fs.
  "A system extension image extend the base system with an overlay containing additional files."

- Same concept, different tools
  - System extension image vs PE file
  - mkosi vs ukify to create a signed sysext (requires dm-verity loaded)
  - Same path where to put it

- Create sysext extension
- Put it into $BOOT/efi/EFI/Linux/$UKI.efi.extra.d (must be a .raw)
- systemd-stub will take care of copying it into initrd's /.extra/sysext/ folder
- systemd-sysext will take care of taking the extension and using it before switching to root

https://github.com/systemd/mkosi/commit/c42d816

Red Hat

Target users

Who creates them

- 3 group of users can create them:

Who creates them

- 3 group of users can create them:
- Vendors (RH) that have already their keys inserted into the machine secureboot database
    - Add machine-specific cmdline (in the cloud, different VMs with different features) , debug addons for developers to debug kernel issues, …

Who creates them

- 3 group of users can create them:
- Vendors (RH) that have already their keys inserted into the machine secureboot database
    - Add machine-specific cmdline (in the cloud, different VMs with different features) , debug addons for developers to debug kernel issues, …
- Virt host admins that use host-side tools like virt-firmware to inject keys in machine OVMF variables
    - Custom cmdline, debug addons, …

Who creates them

- 3 group of users can create them:
- Vendors (RH) that have already their keys inserted into the machine secureboot database
    - Add machine-specific cmdline (in the cloud, different VMs with different features) , debug addons for developers to debug kernel issues, …
- Virt host admins that use host-side tools like virt-firmware to inject keys in machine OVMF variables
    - Custom cmdline, debug addons, …
- Guest admins that use guest-side tools like MOK to insert keys in the secureboot db
    - Note: usually not allowed by cloud providers, like Azure
    - Add custom cmdline, debug addons, …

Available tools

Systemd tools

- v253: ukify capable of creating UKIs
- v254: ukify support for UKI addons (`ukify build`)
- v255: ukify support for UKI/addons inspection (`ukify inspect`)
- Features still to merge:
  - Enable bootclt to find the addons and display for each UKI the full cmdline (default + all used addons)
- mkosi: create systemd-syext images

uki-direct (part of virt-firmware)

- kernel-bootcfg: add, update, remove UKIs (generally show and manage uefi boot entries)

uki-direct (part of virt-firmware)

- kernel-bootcfg: add, update, remove UKIs (generally show and manage uefi boot entries)
- Future releases:
  - kernel-addon: add, update, inspect and remove UKI addons
    - Requires `ukify inspect`

Future work

UKI addons RPM

UKI addons RPM

- Ship a RPM with a collection of addons (debug, cloud-specific)
    - Signed by the vendor

UKI addons RPM

- Ship a RPM with a collection of addons (debug, cloud-specific)
    - Signed by the vendor

- RPM installs them into a specific location not in ESP (/usr/lib/linux/extra.d/ for global, /usr/lib/linux/$UNAME/$UKI.efi.extra.d/ for UKI-specific)

https://github.com/uapi-group/specifications/pull/91

UKI addons RPM

- Ship a RPM with a collection of addons (debug, cloud-specific)
  - Signed by the vendor

- RPM installs them into a specific location not in ESP (/usr/lib/linux/extra.d/ for global, /usr/lib/linux/$UNAME/$UKI.efi.extra.d/ for UKI-specific)

- Use kernel-addon to install them globally or to a specific UKI

https://github.com/uapi-group/specifications/pull/91

UKI addons RPM

- Ship a RPM with a collection of addons (debug, cloud-specific)
    - Signed by the vendor
- RPM installs them into a specific location not in ESP (/usr/lib/linux/extra.d/ for global, /usr/lib/linux/$UNAME/$UKI.efi.extra.d/ for UKI-specific)
- Use kernel-addon to install them globally or to a specific UKI
- Useful when customer has a bug and developer needs to debug UKI

https://github.com/uapi-group/specifications/pull/91

Cloud

Cloud

- Cloud providers need to provide a way to the user to inject his own certificate into the secureboot db
  - Otherwise custom addons cannot be added

Cloud

- Cloud providers need to provide a way to the user to inject his own certificate into the secureboot db
  - Otherwise custom addons cannot be added
- This also implies that the certificate must be measured in PCR7
  - Solution: add dummy addon at first boot, so that the cert is measured

On prem

On prem

- Libvirt should do the same as what the cloud provider should offer: possibility to upload a certificate for secureboot
  https://issues.redhat.com/browse/RHEL-9690

On prem

- Libvirt should do the same as what the cloud provider should offer: possibility to upload a certificate for secureboot
  https://issues.redhat.com/browse/RHEL-9690
- Insert dummy addon for measurements with `virt-customize --upload`

Questions?

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat