Using Linux as a Secure Boot Loader for OpenPOWER Servers

Nayna Jain
Thiago Jung Bauermann
IBM Linux Technology Center

Disclaimer

- This work represents the view of the author and does not necessarily represent the view of IBM
- All design points disclosed herein are subject to finalization and upstream acceptance. The features described may not ultimately exist or take the described form in a product
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.

Agenda

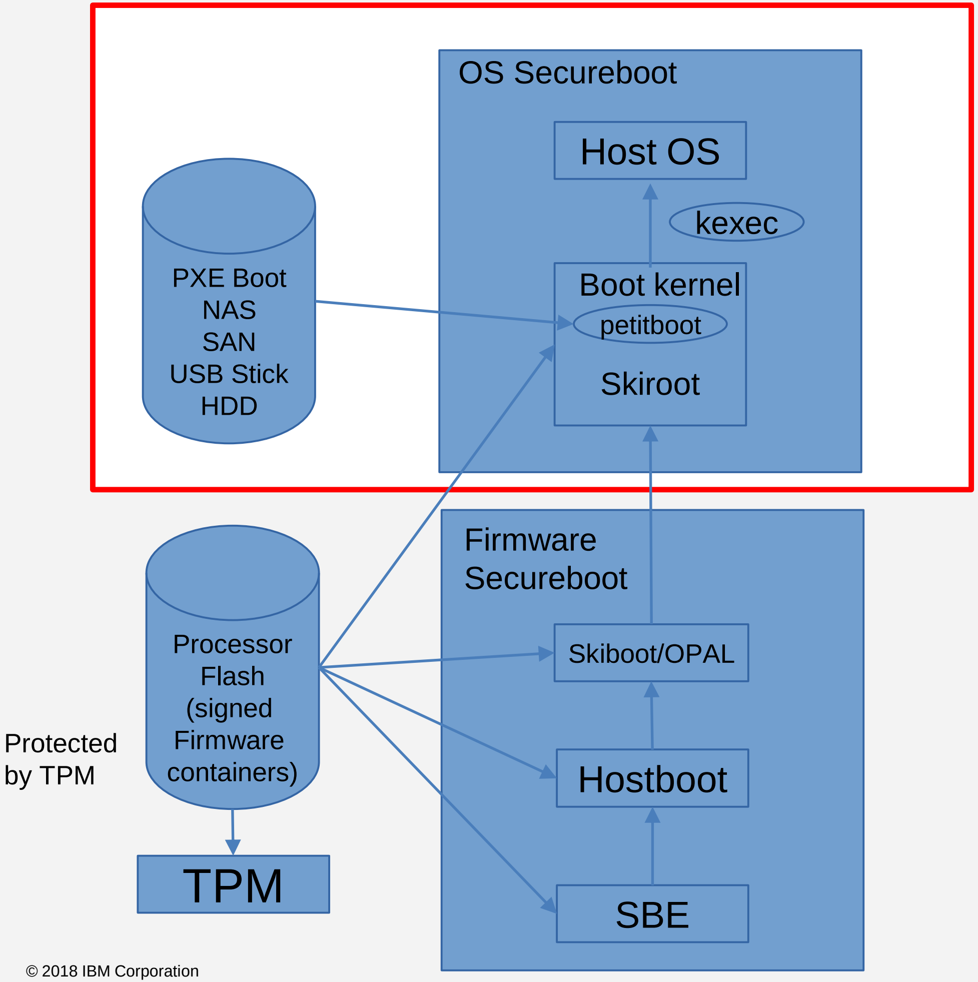
- Secure Boot Goals and Requirements
- Using the Kernel's Existing Methods for Secure Boot
- Missing Functionality
  - Lack of Verification of Network Provided Kernel Image Signature
  - Need for Firmware Keys
  - Lack of Runtime IMA Policies for Secure Boot Enabled Systems
- Proposal
- Patch Set Status
- Summary

Secure Boot Goals and Requirements

Motivation

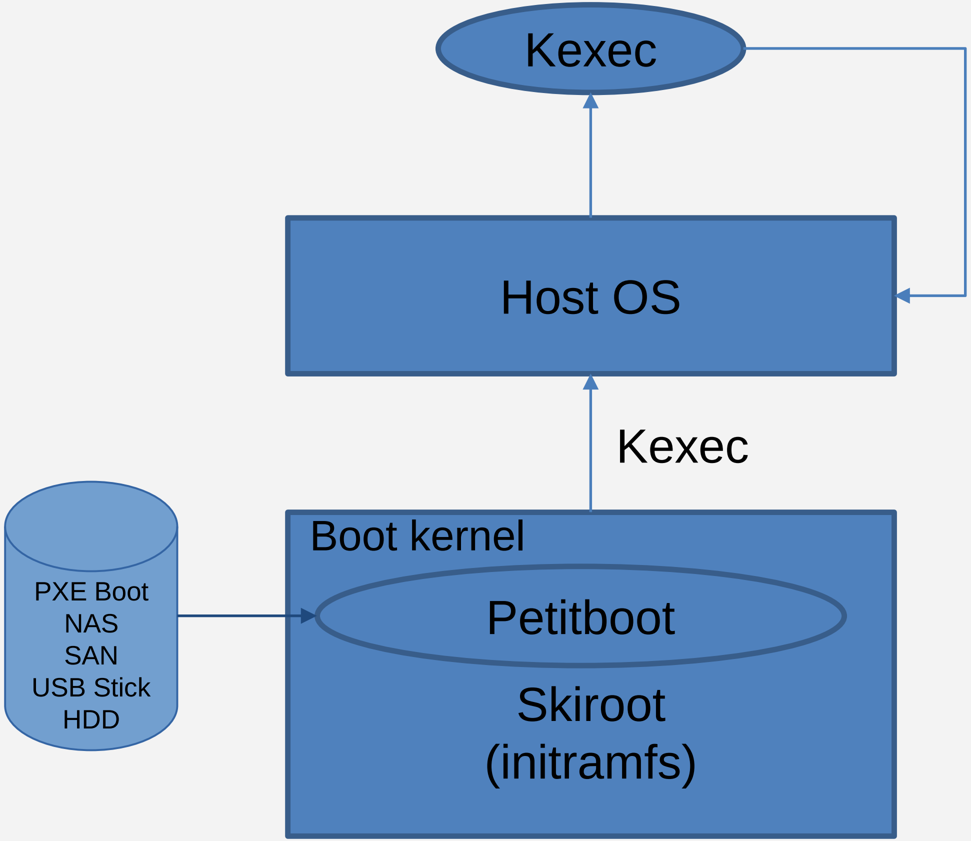
- Protection against bootkits and rootkits
- Only authenticated firmware and operating system are allowed to be executed in the OpenPOWER Secure Boot Solution
- Standards Compliance
  - NIST SP800-147B
  - Common Criteria OSPP 4.1

OpenPOWER Secure Boot



- Each layer verifies next layer before loading
- Firmware is stored as signed containers in the Processor Flash
- Petitboot is the Linux Kernel based Bootloader
- Firmware is verified

OpenPOWER Bootloader Flow



- Petitboot can fetch the kernel in multiple ways
- Petitboot calls kexec to load the host operating system
- Host OS can kexec other kernel or itself

Requirements

- Verify the kernel before loading
- Honor the secure boot state of the system – setup, audit, user
- Carry IMA logs across the kexec
- Disable/Enable OS secure boot irrespective of firmware secure boot state to maintain backward compatibility with unsigned legacy kernels
- Maintain the secure boot policies across the kexec
- Disable kexec_load
- Support the firmware keys like db, kek
- Reuse existing mechanism

Using Kernel's Existing Methods for Secure OS Boot

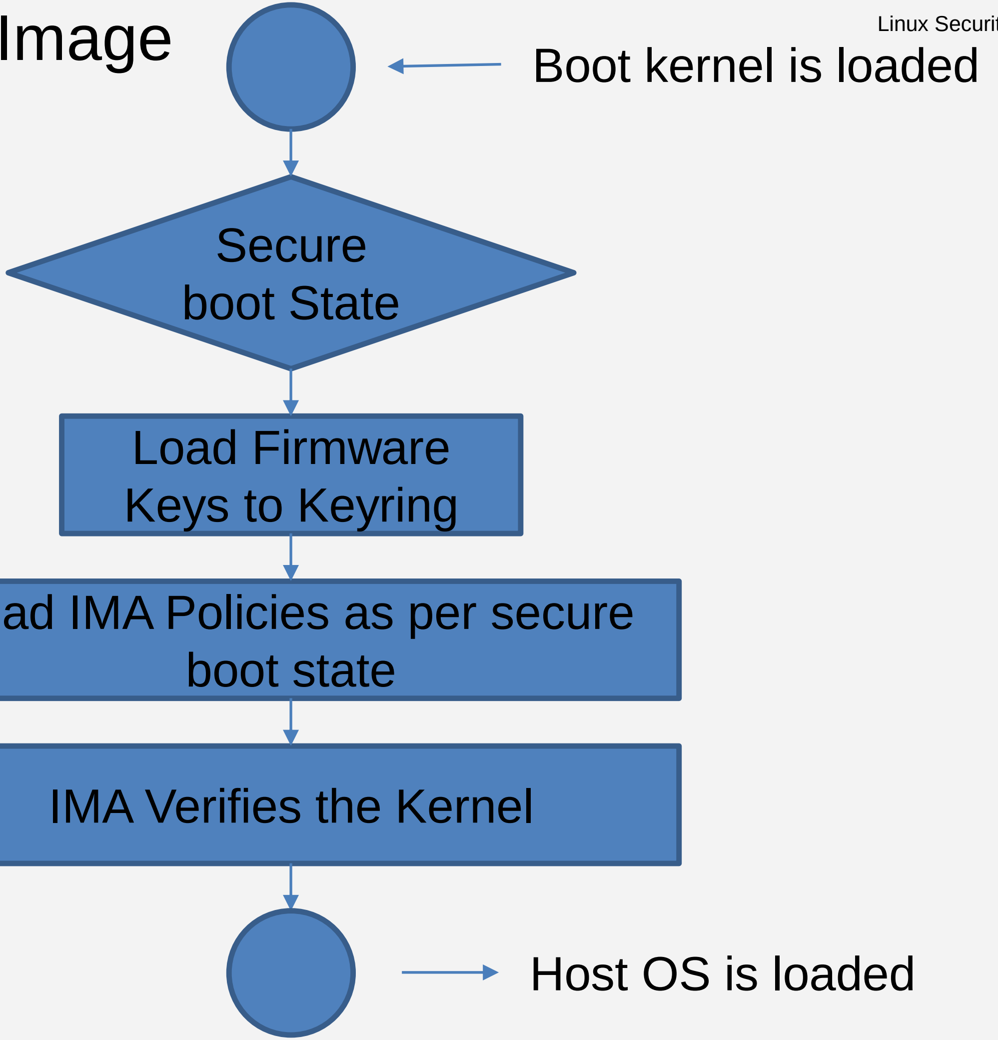
Integrity Measurement Architecture (IMA)

Integrity Measurement Architecture (IMA)

- *A well-tested and well-proven kernel security hook*
- *It does the measurement and appraisal of files based on defined IMA policies*

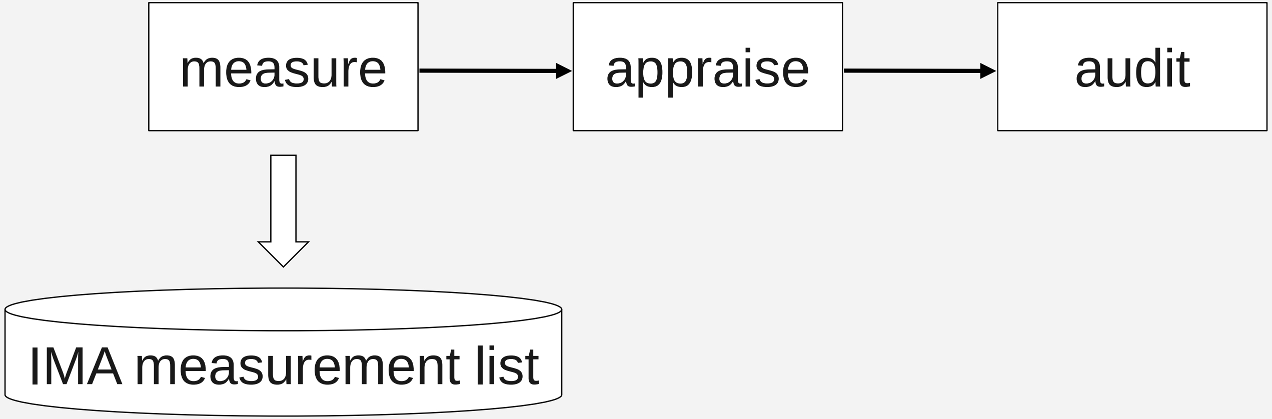
Git Repo: git://git.kernel.org/pub/scm/linux/kernel/git/zohar/linux-integrity.git
Maintainer: Mimi Zohar
Mailing List: linux-security-module@vger.kernel.org

Using IMA for Kernel Image Verification



Integrity Measurement Architecture Basics

```
measure   func=FILE_CHECK          mask=MAY_READ uid=0
appraise  func=KEXEC_KERNEL_CHECK appraise_type=imasig
audit     func=BPRM_CHECK
```



IMA Signature Verification Today

```
$ sudo evmctl ima_sign -a sha256 --key ima-key.pem signed-file
$ getfattr -e hex -n security.ima signed-file
# file: signed-file
security.ima=0x030204b5c1246a0200a79a7808291c23faade15d1fc9b03
9d27e704490463358bac5c48fed7ebcca164f409c3b1ff986837f31d8da67e
ea4a4d7160d4031430e2c6590ebebcb8afe947b27d9859ca7e95 …
```

struct signature_v2_hdr

```
uint8_t type;         → EVM_IMA_XATTR_DIGSIG
uint8_t version;
uint8_t hash_algo;    → HASH_ALGO_SHA256
__be32  keyid;
__be16  sig_size;
uint8_t sig[];
```

Extended Attribute (xattr)-based Signature Pros and Cons

Pros:

- No need to modify the binary being signed
- Don't get in the way of reproducible builds

Cons:

- Network boot: can't be transmitted via TFTP or HTTP protocols
- Filesystems common on boot devices: FAT-based, ISO 9660

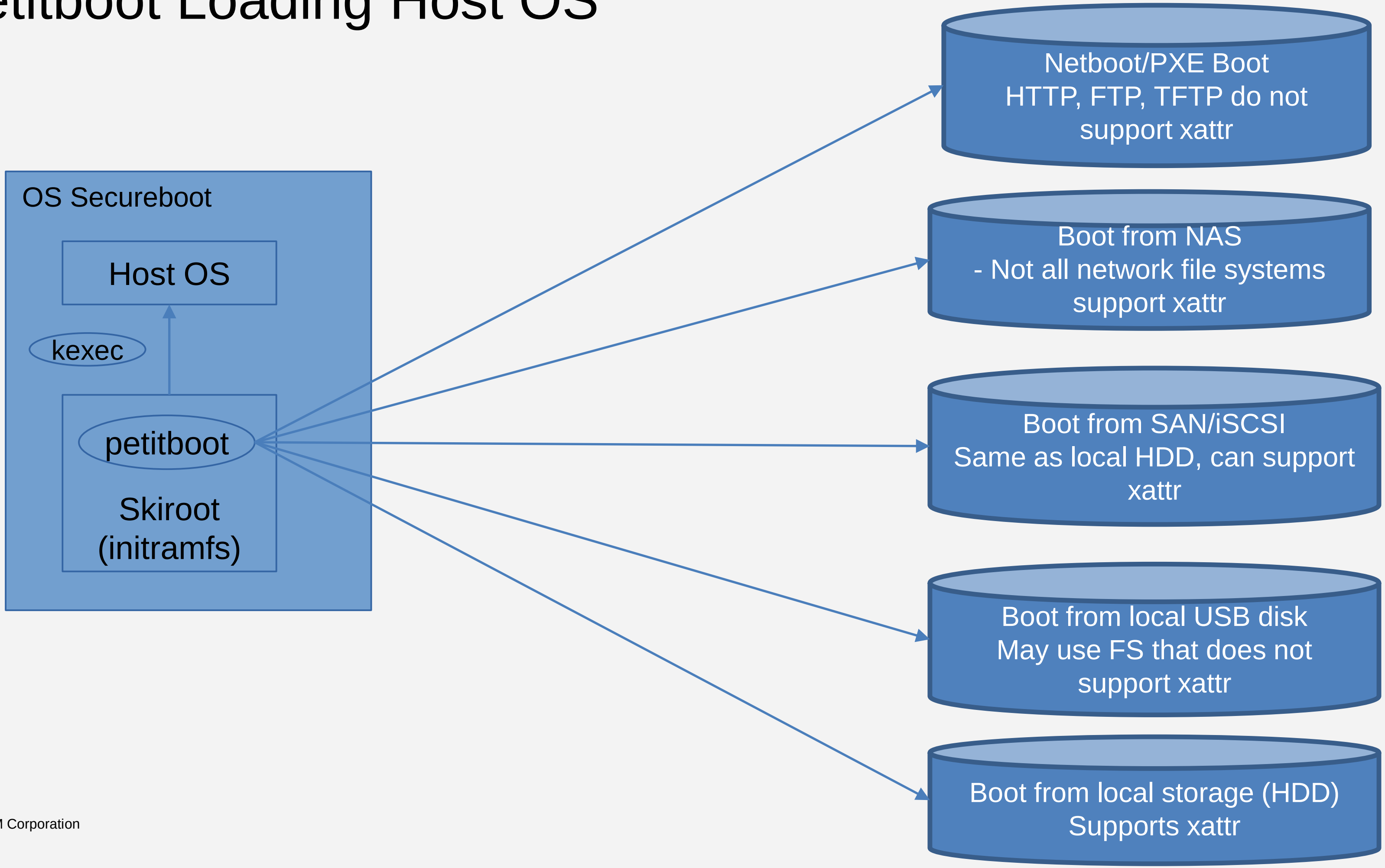
Missing Functionality

Three Main Problems

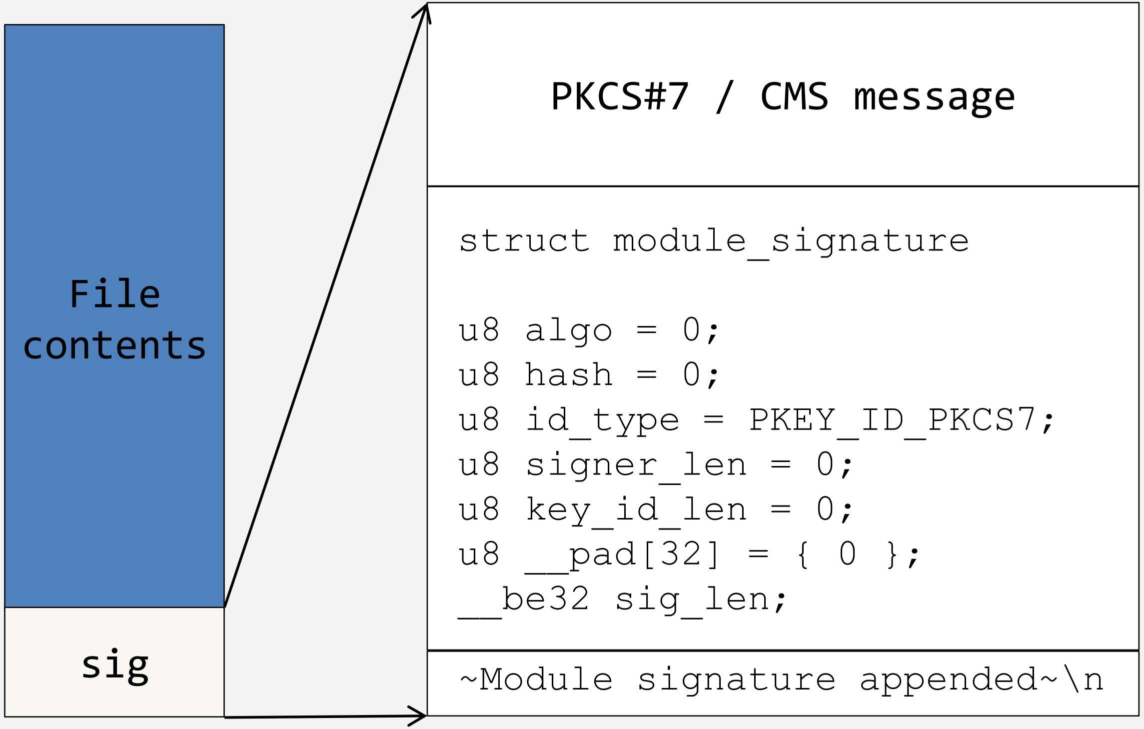
- Lack of verification of network provided kernel image signature
- Need for firmware keys
- Lack of runtime IMA policies for secure boot enabled systems

Lack of Verification of Network Provided Kernel Image Signature

Petitboot Loading Host OS



Kernel Module Signatures

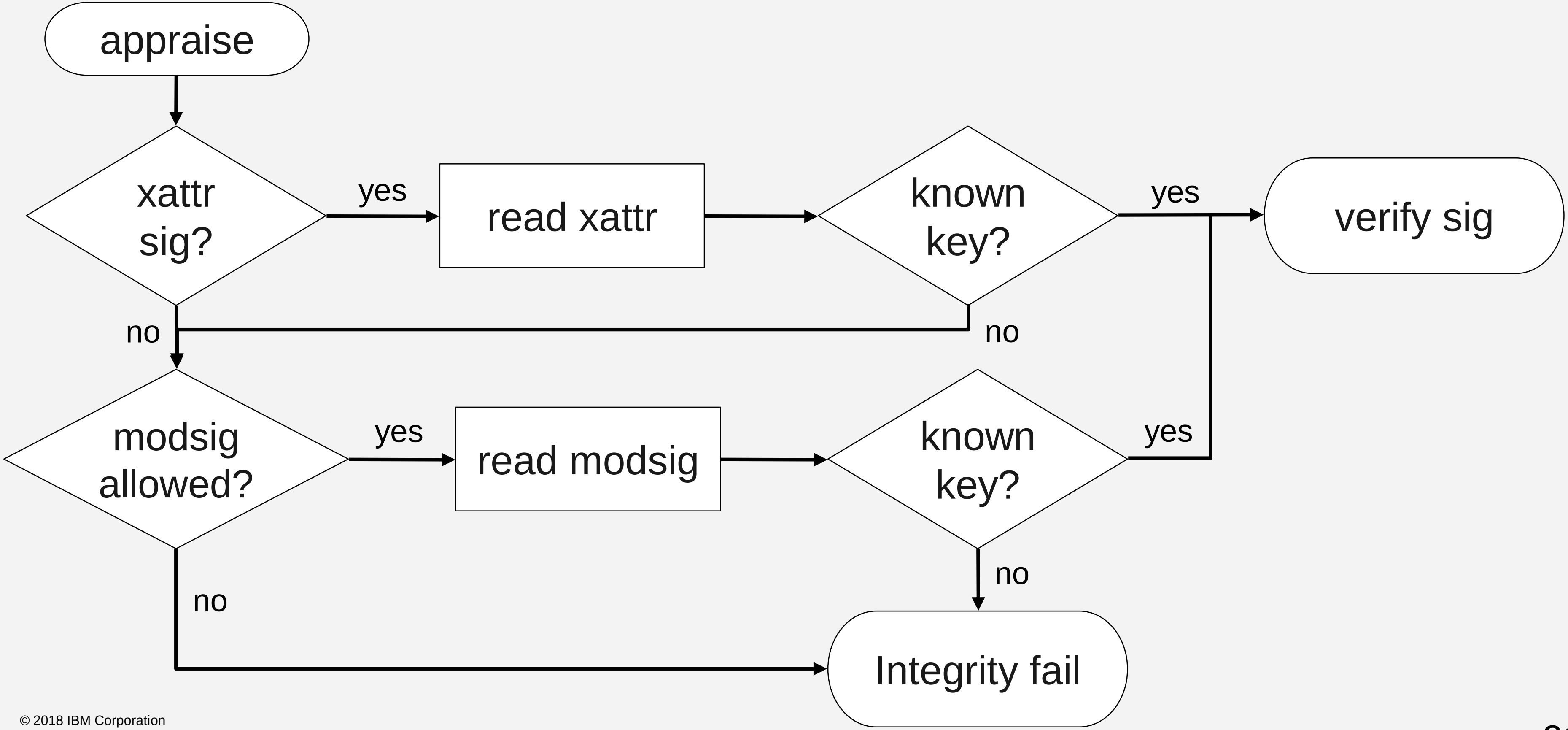


PKCS#7 / CMS

```
$ ~/src/linux/scripts/sign-file [-k] sha1 key.pem key.der signed-file
```

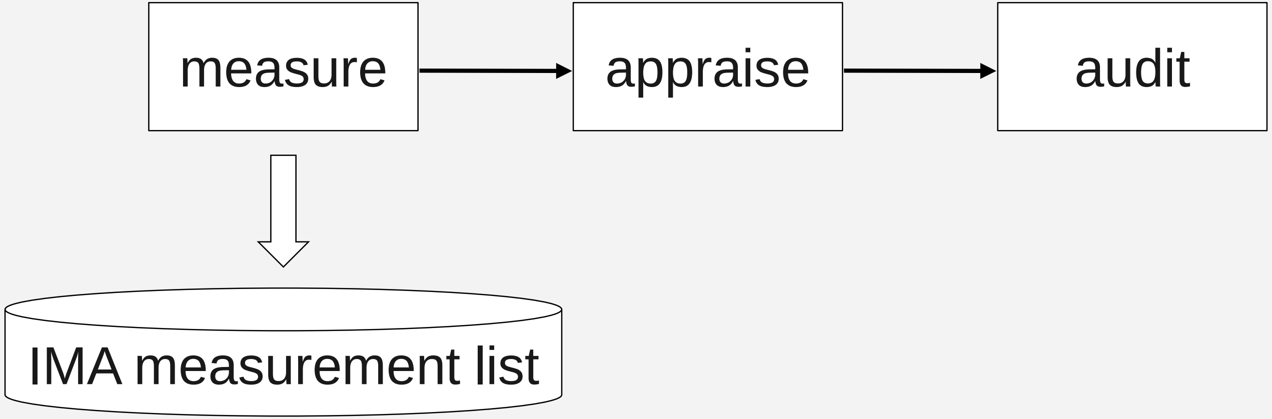
```
SEQUENCE {
  OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
  [0] {
    SEQUENCE {
      INTEGER 1
      SET {
        SEQUENCE {
          OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
          }
        }
      SEQUENCE {
        OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
        }
      SET {
        SEQUENCE {
          INTEGER 1
          SEQUENCE {
            SEQUENCE {
              SET {
                SEQUENCE {
                  OBJECT IDENTIFIER countryName (2 5 4 6)
                  PrintableString 'XX'
                  }
                }
              SET {
                SEQUENCE {
                  OBJECT IDENTIFIER localityName (2 5 4 7)
                  UTF8String 'Default City'
                  }
                }
              SET {
                SEQUENCE {
                  OBJECT IDENTIFIER organizationName (2 5 4 10)
                  UTF8String 'Default Company Ltd'
                  }
                }
              }
            INTEGER 00 89 82 7F 09 69 54 23 F0
            }
          SEQUENCE {
            OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
            }
          SEQUENCE {
            OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
            NULL
            }
          OCTET STRING
            98 35 8F 09 BD 29 5C 02 39 4F 33 7B 5D 52 3B 2D
            27 2A F1 CF C1 18 60 E6 93 AF 6E BA 2C 06 DE C5
            AC B4 E4 05 A3 26 E1 DB 3A 7C 72 D4 3A 81 42 2F
            2E 40 74 70 73 41 E3 AB B6 95 1E 16 D6 01 81 EA
            C9 28 26 68 B9 5D EA 00 8A BB D9 A4 D8 D7 DE 14
            6D C6 56 52 E3 E5 4D D1 78 72 CE AD 74 99 E4 4B
            ED 52 EB 48 22 78 5C 90 9D 14 F2 63 F9 00 7D 63
            87 18 6C 97 3D 48 17 AC 60 FC 74 21 CE 68 45 58
          }
        }
      }
    }
  }
```

File Appraisal Using modsig



modsig in the Measurement List

```
measure   func=KEXEC_KERNEL_CHECK
appraise  func=KEXEC_KERNEL_CHECK appraise_type=imasig
audit     func=BPRM_CHECK
```



IMA Template

Template fields:

- **d**: digest of the event. Either MD5 or SHA1.
- **n**: name of the event, maximum of 255 characters.
- **d-ng**: digest of the event, with arbitrary hash algorithm.
- **n-ng**: name of the event, with arbitrary length.
- **sig**: the file signature.

Template descriptor: `d-ng|n-ng|sig`

Predefined template descriptors:

- **ima**: `d|n`
- **ima-ng**: `d-ng|n-ng`
- **ima-sig**: `d-ng|n-ng|sig`

Kernel command line:

- `ima_template=ima|ima-ng|ima-sig`
- `ima_template_fmt=field1|…|fieldn`

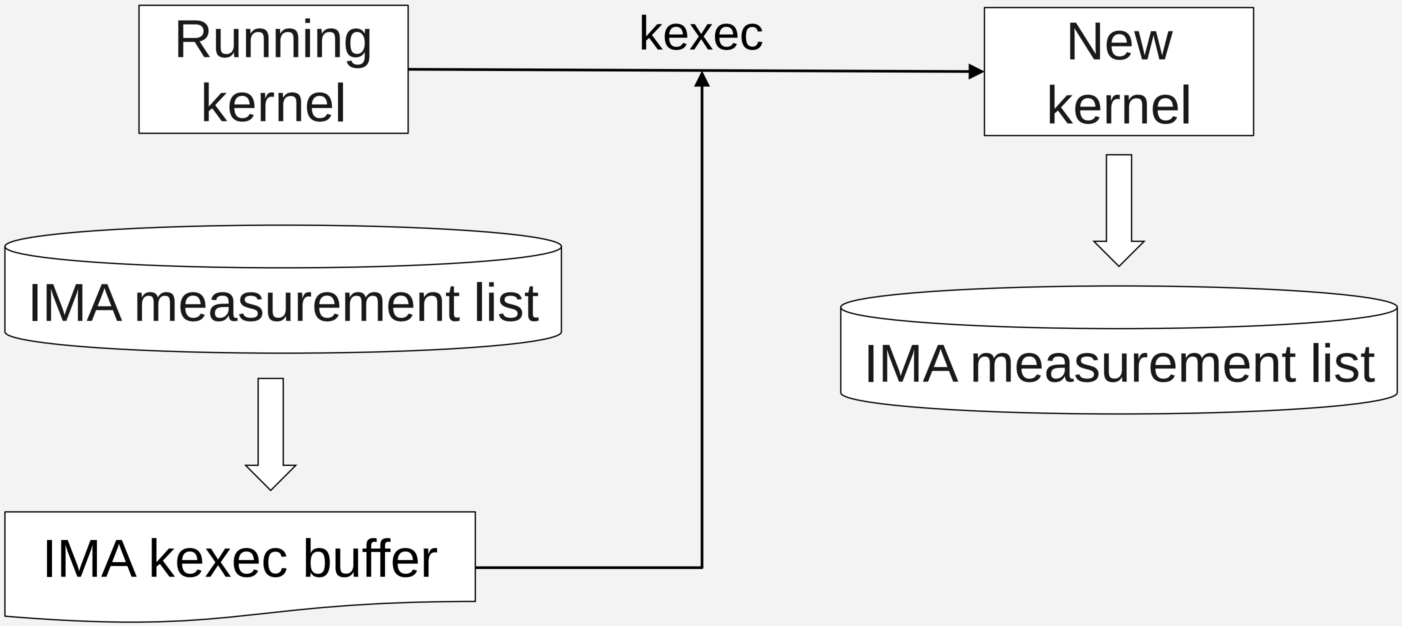
Measurements with Signatures

```
10 de300e21cec276166ec41a53c7e758e5a863ba3e ima-sig \
 sha256:14c156bb471251e4037ca5fe35346337a0a5ab08e24ed45f750bddc8a532b628 \
 /path/kernel-signed-trusted-xattr \
 030204696753f30200378ceeacefff28e7c88775b48b36497913ce36785352bba48df5eba984ebe5
0e8fc2b763385cda3b0dd868a0e51f3a201dafba2b298c99a18e4fa6ff116aa8fdda5088c1c615b4c
7815482565809 4d6fdf6c4cb00307808b80d1de8d5e5f61b0d8284f7c2f305e054859738855c66c25
729d0e004229431ea175b30573b1ab0aec71f1f1b0e2b86e63c877f2e6588d10ee2934ca7b0f6feea
4ae2519399d14a1efda4df7c0e5338d8163760dad1a1d24a0f24d0b43964deb881ec01d96fbb1e8f6
b4f8d29c6bb98ac8c00039801dd6aa34ac4ae814bd02b456726fd43bd86f3420332251ccf18b5a41
991fe95401c64d14c8d1dd491ba31c76ed14a63408fb68f98f5cef35d4393e7b26c8e2ed3e339773c
5b1ce2b6c04679a3ceb931b206ae11fb5219f64c825247aaaa77bafa1cdfc3d8c3e5abcb516012f30
6124331ef92124bf72a490b615c8ea78e3c92213bc739affa89938abb59b67b2386d8b18071f85f2e
a943cf59f41626db78b6f6bfc2ce1acb80abfb70929227ad7708869e5402ab7d1454c5001da249ac9
1b48ca514e7eced135a4432f4e31701ff2c1e5ca6801c605b0a25bb629818a8a809d32615b2ee26f9
a99f79a400c7c4a7762fb1a3134b610b6b3fdf6aa51364d95aa06cbbf4ede0500fc83cf415d6a03c8
e782a441c08c69944876f551e356b60eae8052ddc96ff4fa9769a6093da0b5a10171f9c
```

Measurements with Signatures

```
10 3c6d061d883073289e7353ae81d914c46cb95ed4 d-ng|d-sig|n-ng|sig \
 sha1:f760bf369fc049f6c32e79d0e6a2a6a5596c7e88 \
 sha256:14c156bb471251e4037ca5fe35346337a0a5ab08e24ed45f750bddc8a532b628 \
 /path/kernel-signed-trusted-modsig \
06308202c906092a864886f70d010702a08202ba308202b6020101310d300b0609608648016503040
201300b06092a864886f70d010701318202933082028f020101306a305d311b3019060355040a0c12
4c696e757820546573742050726f6a656374311d301b06035504030c14494d41207472757374465642
074657374206b6579311f301d06092a864886f70d01090116107465737440657861d706c652e6f72
67020900a925ba7010c2d8ba300b0609608648016503040201300d06092a864886f70d01010105000
48202001295fecd7cb428cc24373e6b9104b2cdb0791e2c084aee824f0a495e5d99fd3428db332b60
fb2944214e45d893ac4ba1126f41f2c7c24f583cc5a8c69a13385096a91b4982370a30370e869b2e6
5a0b680d693a19cebfc4223ae6f18e35facbbc6b8880706410a01b3480043545c5774a09588c5c678
29d2cbba5bfbadfc76bcf0f109c20662d38877589f3af199b597a5d863cfd8236dcfe78e6b419255d
09aeed5b53e5d82070bce8564b405e7ee5393a74415c02c0f94e948ddd078252c911178b348dcf057
ed5c838cad4219e79d1882cd3d8690a575adfe377c2e1e708da426aba545f6137f520dcb4fdffa4b2
122d077d614d92cfb5fc70875b49173dd09d1da9f …
```

Carrying the IMA Measurement List across kexec



Need for Firmware Keys

Existing Keyrings of Interest

Keyring Name	Origin	Used for	User Modifiable	Scope
.builtin_trusted_keys	Compiled-in	Verifying signatures on keys and kernel modules	No	Limited
.secondary_trusted_keys	Signed by .builtin_trusted_keys or .secondary_trusted_keys	Verifying signatures on keys and kernel modules	Yes	Limited
.ima	Signed by .builtin_trusted_keys or .secondary_trusted_keys	Verifying signatures on files	Yes	Broad
_ima – user defined keys	Unsigned, typically loaded in initramfs before system pivots root	Verifying signatures on files	Yes	Broad

Missing Keyring of Interest

Keyring Name	Origin	Used for	User Modifiable	Scope
.builtin_trusted_keys	Compiled-in	Verifying signatures on keys and kernel modules	No	Limited
.secondary_trusted_keys	Signed by .builtin_trusted_keys or .secondary_trusted_keys	Verifying signatures on keys and kernel modules	Yes	Limited
.ima	Signed by .builtin_trusted_keys or .secondary_trusted_keys	Verifying signatures on files	Yes	Broad
_ima – user defined keys	Unsigned, typically loaded in initramfs before system pivots root	Verifying signatures on files	Yes	Broad
	Compiled-in or loaded from firmware	Verifying signatures on kernel image (kexec_file_load syscall)	No	narrow

Proposed New Keyring - .platform_keys

- Platform Trusted or compiled-in keys loaded during the boot time
- Accepts non-verifiable keys
- Non-modifiable by userspace
- This keyring can be enabled by setting CONFIG_PLATFORM_KEYRING. The platform certificate can be provided using CONFIG_PLATFORM_TRUSTED_KEYS.

```
0481da82 I------     1 perm 1f0f0000     0     0 keyring   .ima: empty
04d8da6e I------     1 perm 1f030000     0     0 asymmetri Build time autogenerated kernel key: c
0553752f I------     2 perm 1f0b0000     0     0 keyring   .platform_keys: empty
```

```
[root@witherspoon1 ~]# keyctl show %keyring:.platform_keys
Keyring
  89355567 ---lswrv      0     0  keyring: .platform_keys
```

Lack of Runtime IMA Policies for Secure Boot Enabled Systems

Existing IMA Policies of Interest

Policy	When Defined	Custom Modifiable	Preserved across Kexec	Remarks
ima_tcb	Boot param	Yes	No	Default policies for measurement
ima_appraise_tcb	Boot param	Yes	No	Default policies for appraisal
secureboot	Boot param	Yes	No	Secure boot specific appraisal policies
build time	Compile time	No	No	Secure boot specific appraisal policies

Missing IMA Policy of Interest

Policy	When Defined	Custom Modifiable	Preserved across Kexec	Remarks
ima_tcb	Boot param	Yes	No	Default policies for measurement
ima_appraise_tcb	Boot param	Yes	No	Default policies for appraisal
secureboot	Boot param	Yes	No	Secure boot specific appraisal policies
build time	Compile time	No	No	Secure boot specific appraisal policies
	Runtime	No	Yes	Based on secure boot state of the system

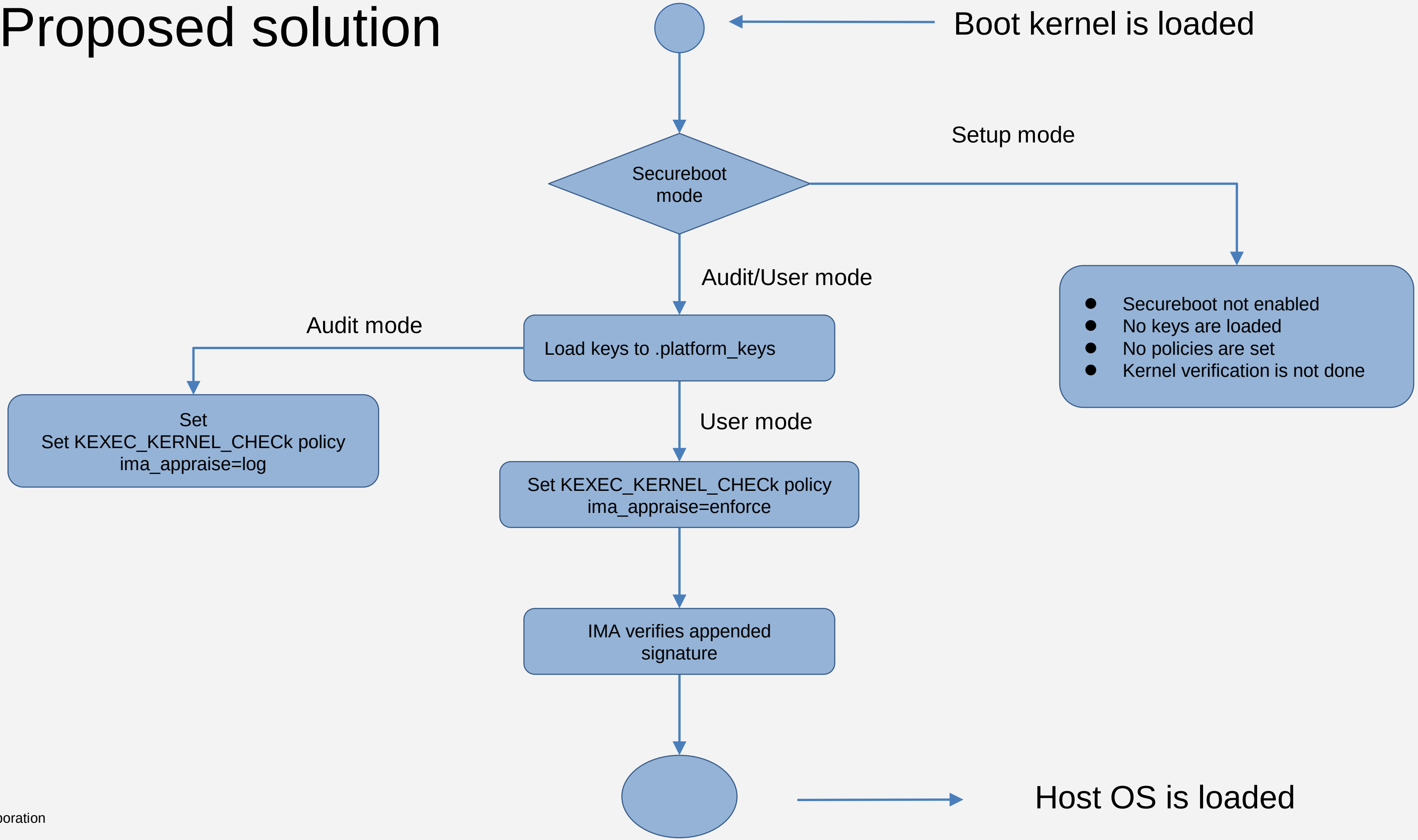
Proposed New Policy - Architecture-specific IMA policy

- Defined by the architecture based on its secure boot state and CONFIGs
- Highest priority
- Cannot be overridden by boot param option
- Disable ima_appraise type to avoid overriding by the boot_param
- This policy can be enabled by configuring IMA_ARCH_POLICY

```
/* arch rules for setup mode */
static const char * const default_arch_rules[] = {
        "measure func=KEXEC_KERNEL_CHECK",
        "dont_appraise func=KEXEC_KERNEL_CHECK",
        NULL
};

/* arch rules for audit and user mode */
static const char * const sb_arch_rules[] = {
        "measure func=KEXEC_KERNEL_CHECK",
        "appraise func=KEXEC_KERNEL_CHECK appraise_type=imasig|modsig",
        NULL
};
```

The Proposed solution



Patchset Status

Feature	Patchset	Status	Remarks	Reference Link
Appended Signatures	IMA Support for Appended Signatures	Awaiting acceptance	Posted upstream on 05/23/2018	https://lists.ozlabs.org/pipermail/linuxppc-dev/2018-May/173406.html
	Carry IMA measurements file across kexec	Upstreamed	Kernel v4.10	
Platform Keyring	Platform Keyring	Awaiting acceptance	Posted on 09/03/2018	https://patchwork.kernel.org/patch/10270991/
	Loading keys into platform keyring	Work In Progress	Based on David Howells patch	Patchset - KEYS: Blacklisting & UEFI database load [6/9] efi: Add EFI Signature Data Types [7/9] efi: Add an EFI Signature Blob Parser
Architecture Specific IMA Policies	IMA support for architecture specific policies	Soliciting Acks/Reviews for x86 patches	Branch –> linux-integrity-queued	git://git.kernel.org/pub/scm/linux/kernel/git/zohar/linux-integrity.git
	Power specific arch policies	Work In Progress		

Summary

- IMA provides kernel – based functions in support of secure boot
- A new, restricted platform keyring supports additional keys
  - Users cannot modify the keys
  - Key usage is limited to kernel images
- A new, IMA supported arch-specific policy can be set at runtime based on secure boot state of the system
- New support for appended signatures enables secure boot even from network
- Multiple patchsets are in process of upstream – awaiting acceptance, awaiting review and under development

References

- General
  - OpenPOWER Foundation: https://openpowerfoundation.org/
  - OpenPOWER Github: https://github.com/open-power
- Skiboot(OPAL)
  - Github: https://github.com/open-power/skiboot
  - Mailing List: https://lists.ozlabs.org/pipermail/skiboot/
- Secure and Boot
  - OpenPOWER secure and trusted boot, Part 2 –Protecting system firmware with OpenPOWER secure boot: https://www.ibm.com/developerworks/library/l-protect-system-firmware-openpower/index.html

Q & A