redhat.

Security Technologies in Red Hat Enterprise Linux

Red Hat Tech Day Belgium 2019

Alexander Bokovoy
Sr. Principal Software Engineer
February 5th, 2019

Security Technologies

Why does it matter?

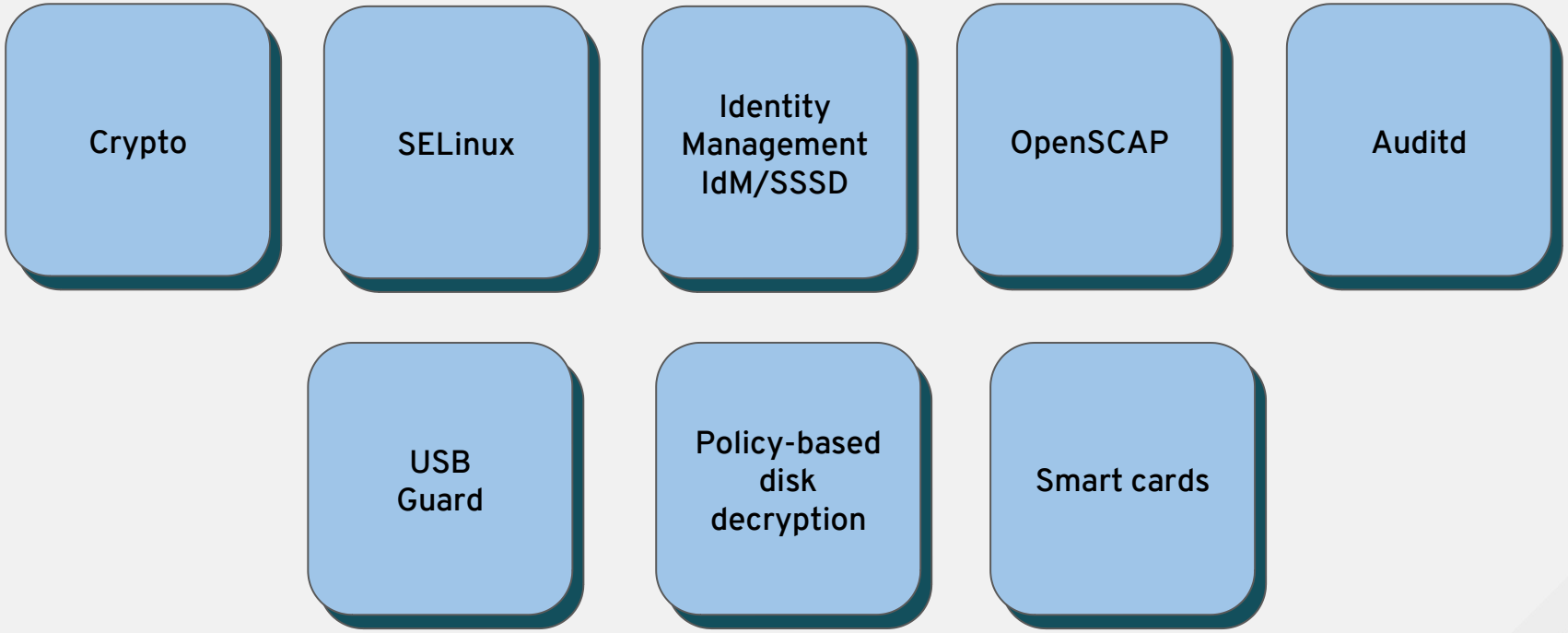
- Crypto needs to be maintained
  - New algorithms need to be added
  - Key sizes changed
  - Old insecure algorithms deprecated
  - Vulnerabilities and bugs addressed
- Crypto libraries enable compliance
  - FIPS
  - Common Criteria
- Libraries need to be integrated to provide solutions
  - TLS 1.3
  - Smart card authentication

Security Technologies

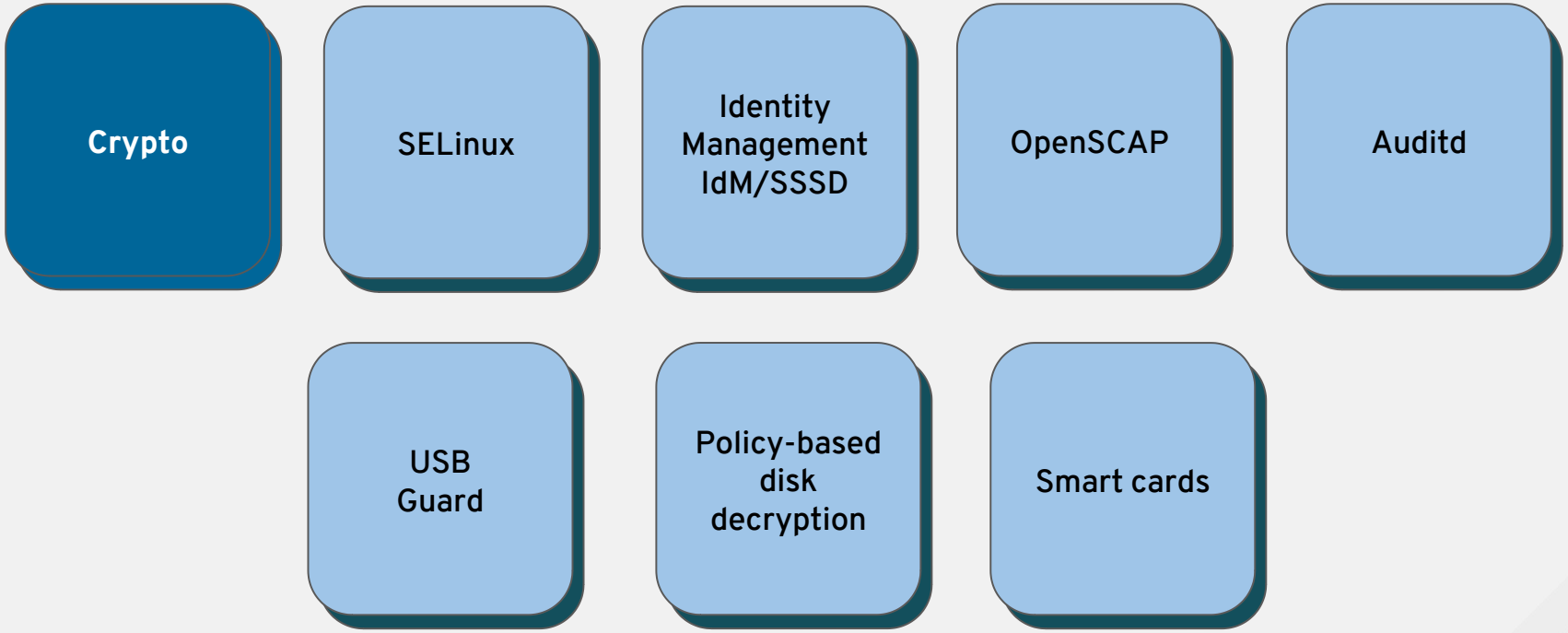
What does Red Hat provide?

- Continued support and maintenance
- FIPS certification and compliance
- Integration to provide solutions like smart card login, HSM support, TPM support, trusted enclaves support, HTTPS/TLS
- Sane defaults and means to change the values
- Backward compatibility for the lifetime of a major release

Security Technologies



Security Technologies



Crypto Technologies

Overview

- RHEL 7.4 – 7.6 changes
  - Deprecating SSH 1.0, SSL 2.0
  - Disabling SSLv3 and DES/RC4 in mod_ssl
  - Adding support for SHA-256
- getrandom()
- Python crypto infrastructure

Crypto Technologies

Crypto algorithms and protocols

- RHEL 7.4 – 7.6 versions deprecated or switched off by default insecure crypto
  - Protocol SSH 1.0 removed from OpenSSH
    - Still available in OpenSSH client but switched off by default
  - SSL 2.0 removed completely from GnuTLS, NSS, and OpenSSL
  - Diffie-Hellman exchange in TLS supports 1024..8192 bits. Anything below 1024 bit is switched off by default.
  - MD5, MD4, and SHA-0 are switched off by default in OpenSSL
  - RC4 is switched off by default in OpenSSH
  - SSL 3.0 protocol and DES/RC4 ciphers disabled in mod_ssl

Crypto Technologies

Crypto algorithms and protocols

- SHA-256 is used by default by all crypto utilities
  - RHEL 7.4 added SHA-256 to OpenSSH
- OpenSSL 1.0.2k in RHEL 7, OpenSSL 1.1.1 in RHEL 8.0 Beta

Crypto Technologies

Crypto algorithms and protocols

- RHEL 8.0 beta
  - Unified system-wide crypto policies, https://access.redhat.com/articles/3666211 and https://access.redhat.com/articles/3642912
  - Four predefined policies: LEGACY, DEFAULT, FUTURE, FIPS

Crypto Technologies

System-wide crypto policies in RHEL 8.0 beta

Policy name	Description
LEGACY	Less secure than other policies and it includes support for TLS 1.0, TLS 1.1, and SSH2 protocols or later. The algorithms DSA, 3DES, and RC4 are allowed, while RSA and Diffie-Hellman parameters are accepted if larger than 1023-bits.
DEFAULT	A reasonable default policy for today's standards, compatible with PCI-DSS requirements. It allows the TLS 1.2 and 1.3 protocols, as well as IKEv2 and SSH2. The RSA and Diffie-Hellman parameters are accepted if larger than 2047-bits.
FUTURE	A conservative security level that is believed to withstand any near-term future attacks. This level does not allow the use of SHA-1 in signature algorithms. The RSA and Diffie-Hellman parameters are accepted if larger than 3071-bits.
FIPS	A level that conforms to the FIPS140-2 requirements. This policy is used internally by the fips-mode-setup tool which can switch the RHEL system into FIPS140-2 compliance mode.

Crypto Technologies

getrandom() syscall

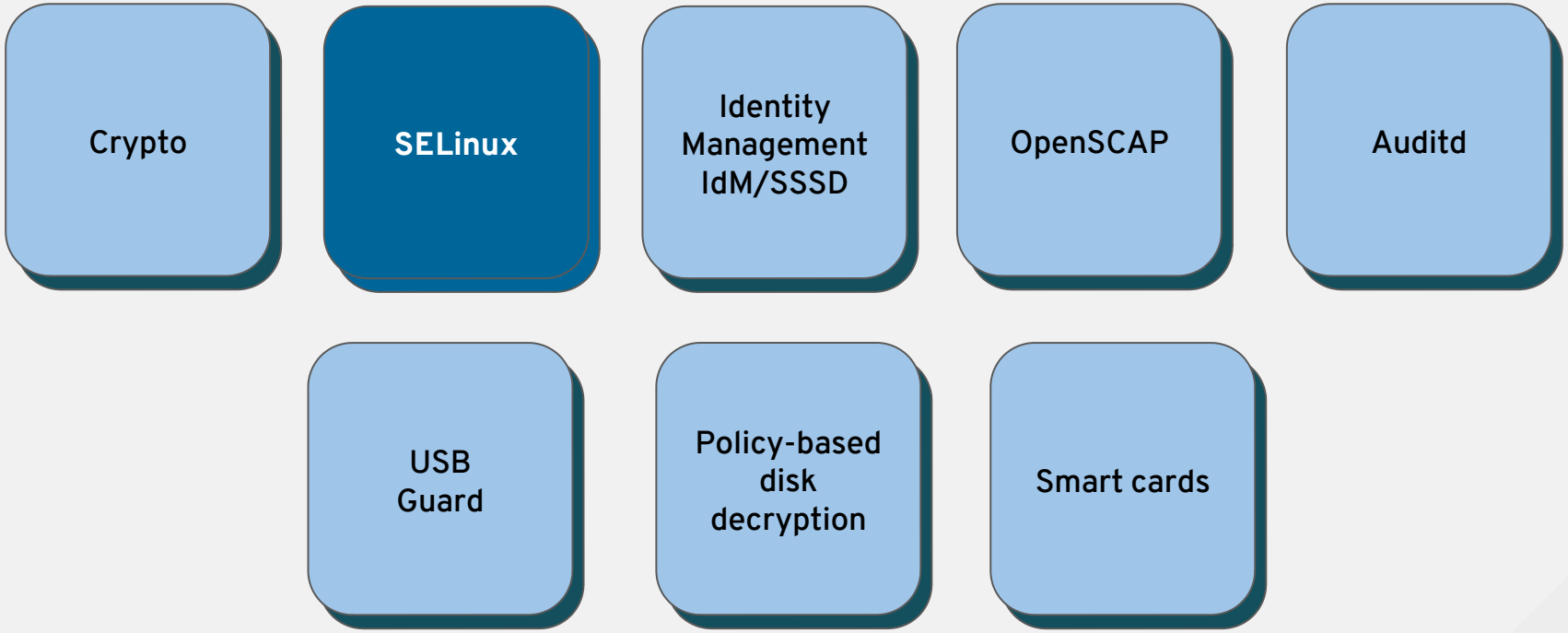
- Since RHEL 7.4, Linux kernel supports getrandom(2) syscall
  - Non-blocking access to the kernel CPRNG at runtime
  - During early boot an access is blocking until 128 bit of entropy is gathered
  - It is a best combination of /dev/random and /dev/urandom behavior for applications

Crypto Technologies

Python cryptography infrastructure

- Since RHEL 7.4 Python does validate X.509 certificates automatically
  - urllib, httplib, xmlrpclib
  - System-wide CA source
    - /etc/pki/ca-trust

Security Technologies



SELinux

Overview

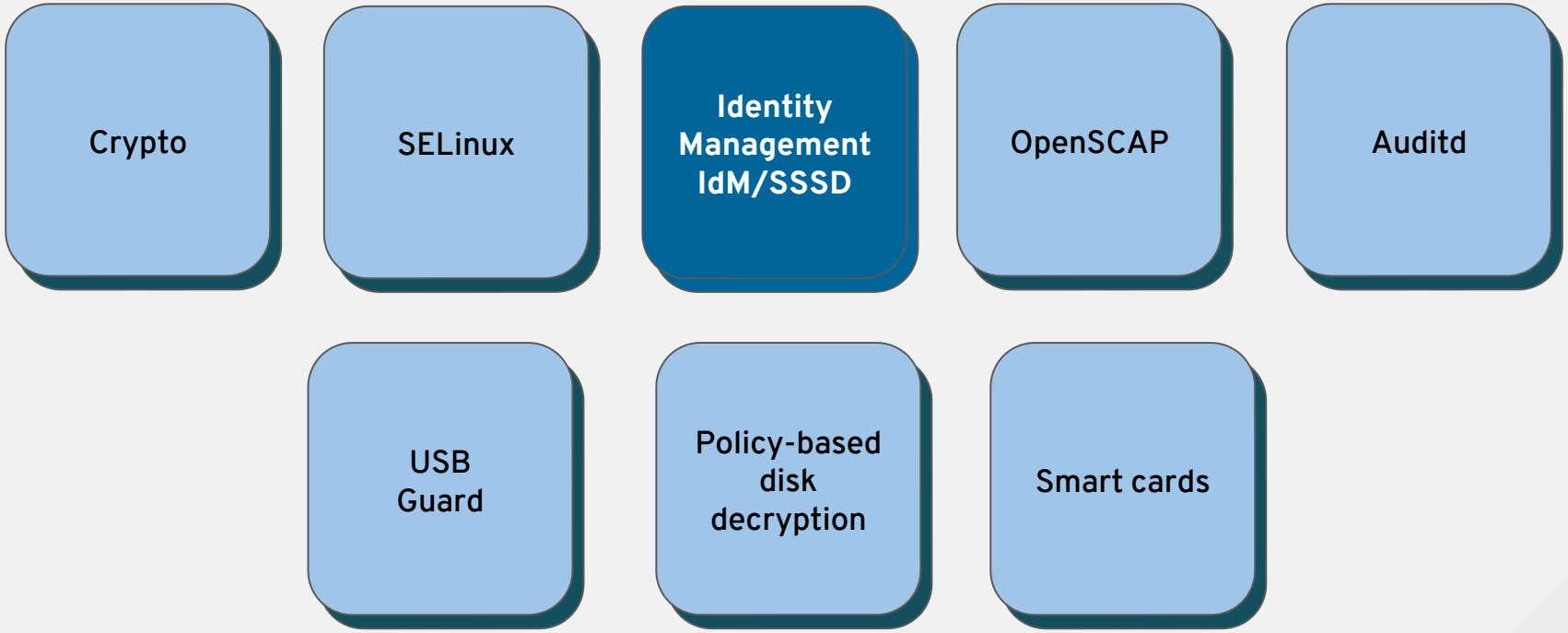
- SELinux is a mandatory access control system
- It is a security mechanism bringing proactive security in your systems
- Allows isolation and confinement of processes from each other and from system resources
- SELinux Enhancing technologies - systemd, sVirt, containers

SELinux

What is new in RHEL 7.4 – 7.6

- SELinux policies are up to date with all new classes and permissions supported by RHEL kernels
- Since RHEL 7.4, container policies are now combined in a single definition class
  - container.te + docker.if → container.pp
- Since RHEL 7.6, SELinux policy does check file permissions on the mmap() system call
  - Prohibits memory mapping of the files which access should be revalidated every time
- OverlayFS in RHEL 7.6 supports SELinux security labels

Security Technologies



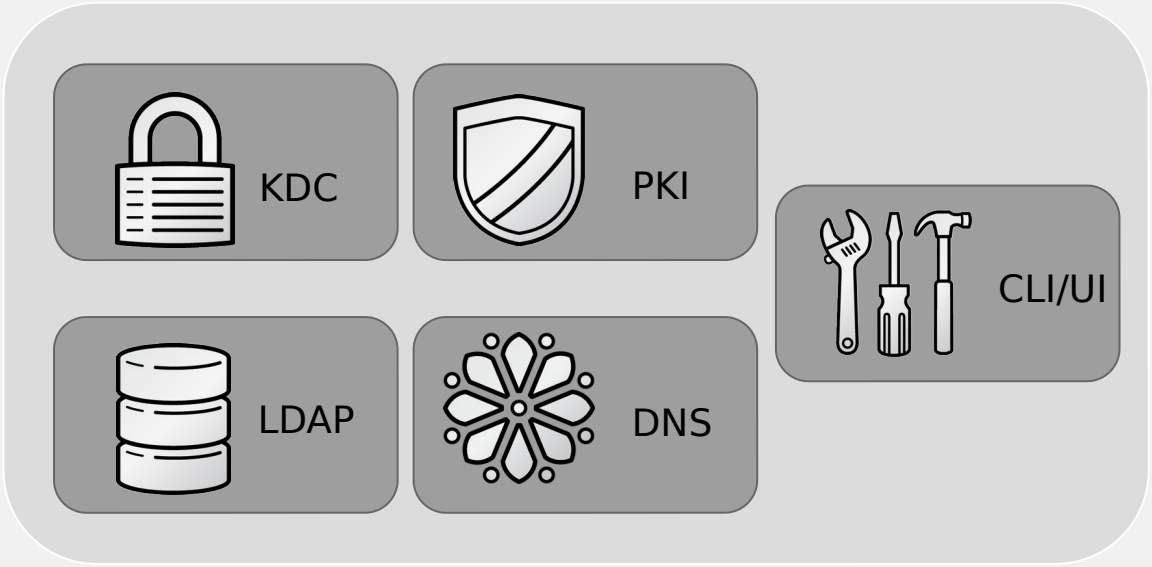
Identity Management

Overview

- Users and groups information for POSIX and application environments
- Authentication and single sign-on with Kerberos
- Centralized access control for multiple application types
- Centralized sudo policies

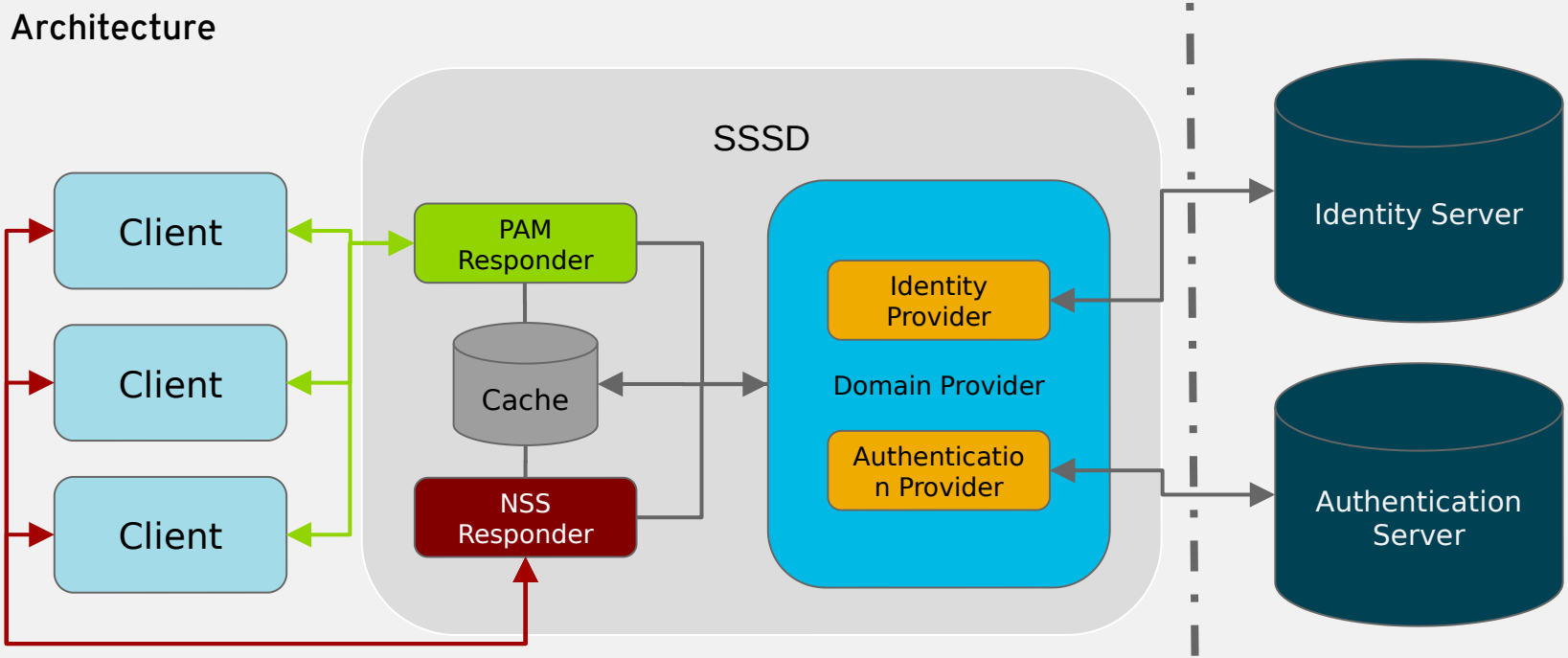
Identity Management - IdM

Architecture

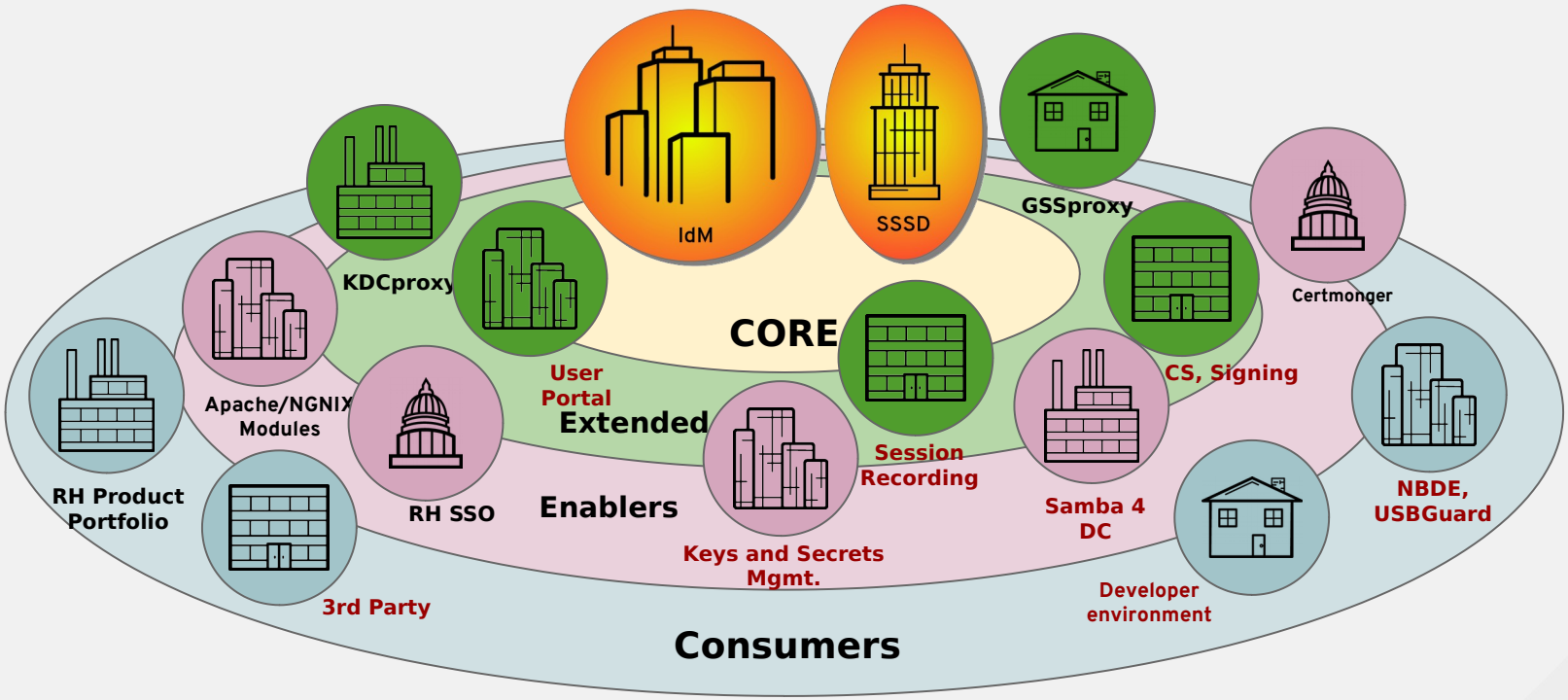


SSSD

Architecture



Identity Management Fabric

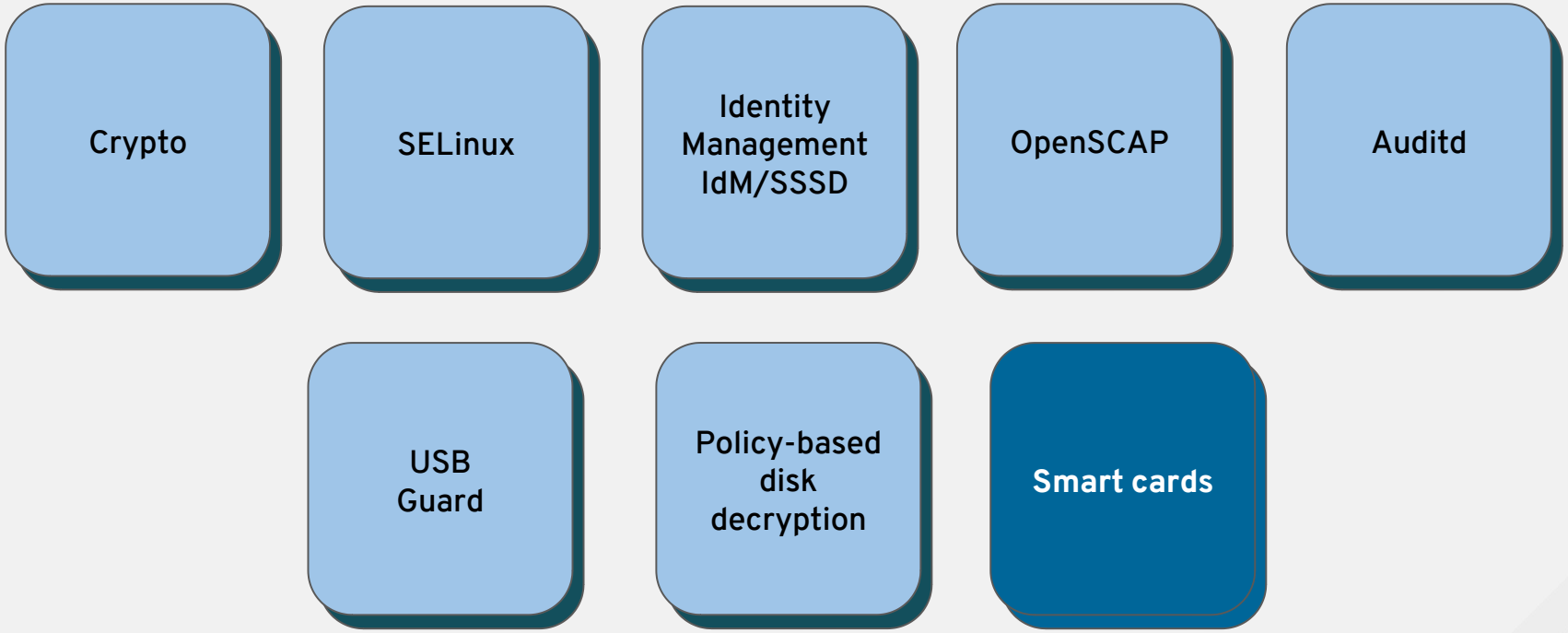


Identity Management

New in RHEL 7.4 – 7.6

- RHEL Identity Management updated to FreeIPA 4.5 (RHEL 7.4) and FreeIPA 4.6 (RHEL 7.6)
- Privilege separation for FreeIPA framework
- Smartcard authentication support
- FIPS mode deployment for Identity Management
- New container images rhel7/ipa-server and rhel7/sssd
- DNSSEC (technology preview)
- API FreeIPA (technology preview)

Security Technologies

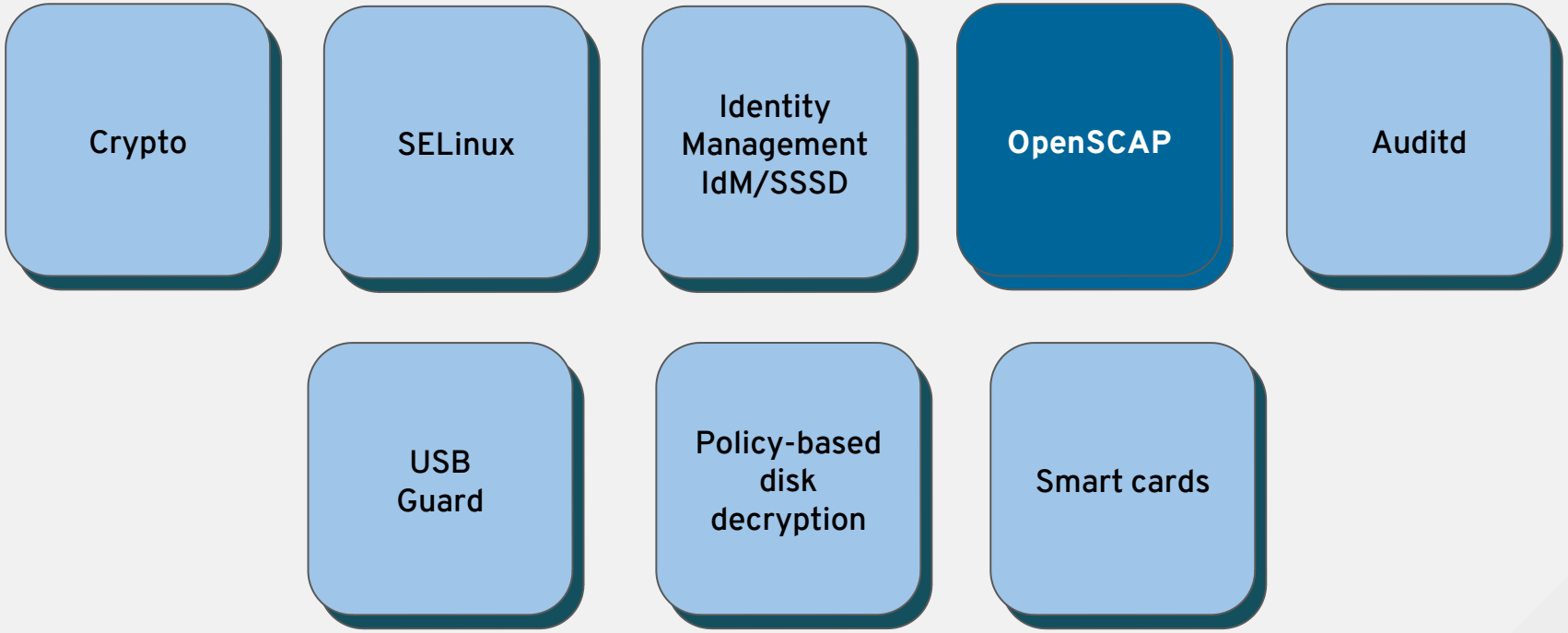


OpenSC

Smartcard support

- OpenSC 0.16.0 since RHEL 7.4
  - Common Access Card (CAC) support
  - PKCS#11 API implementation
  - Transition from coolkey package to OpenSC with Coolkey applet
- Wide range of smartcard readers supported
- CardOS 5.3 smartcards supported since RHEL 7.6
- Non-CCID-compliant PIN pad card readers are supported in RHEL 7.6 (with a catch)

Security Technologies



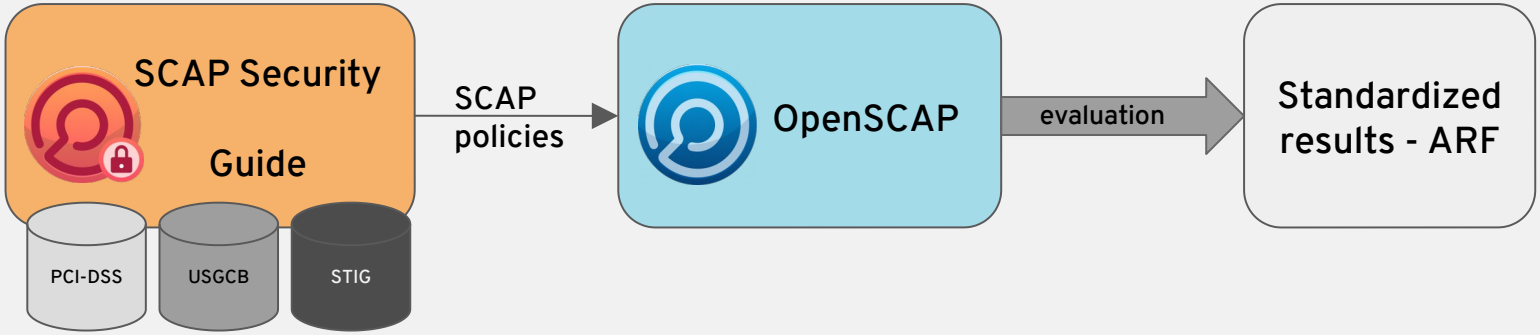
OpenSCAP

Overview

- Scan systems and containers:
  - Do I have a known vulnerability in my deployment?
  - Is the system configuration is in compliance with my baseline?
- OpenSCAP 1.2 has been certified by NIST

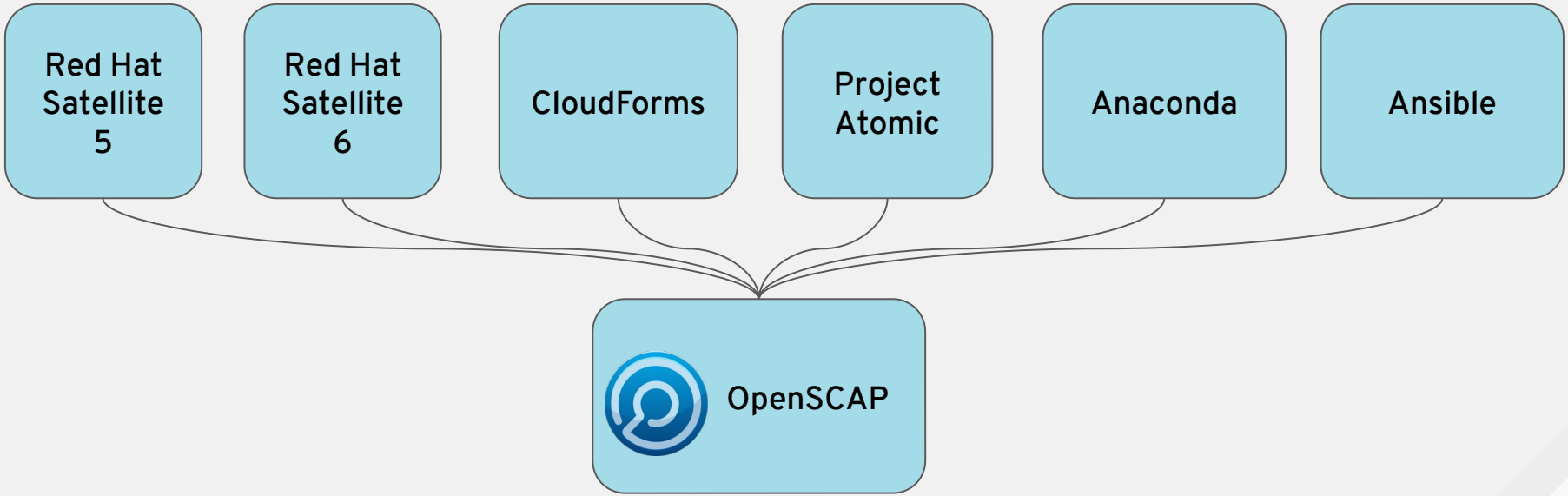
OpenSCAP

Architecture

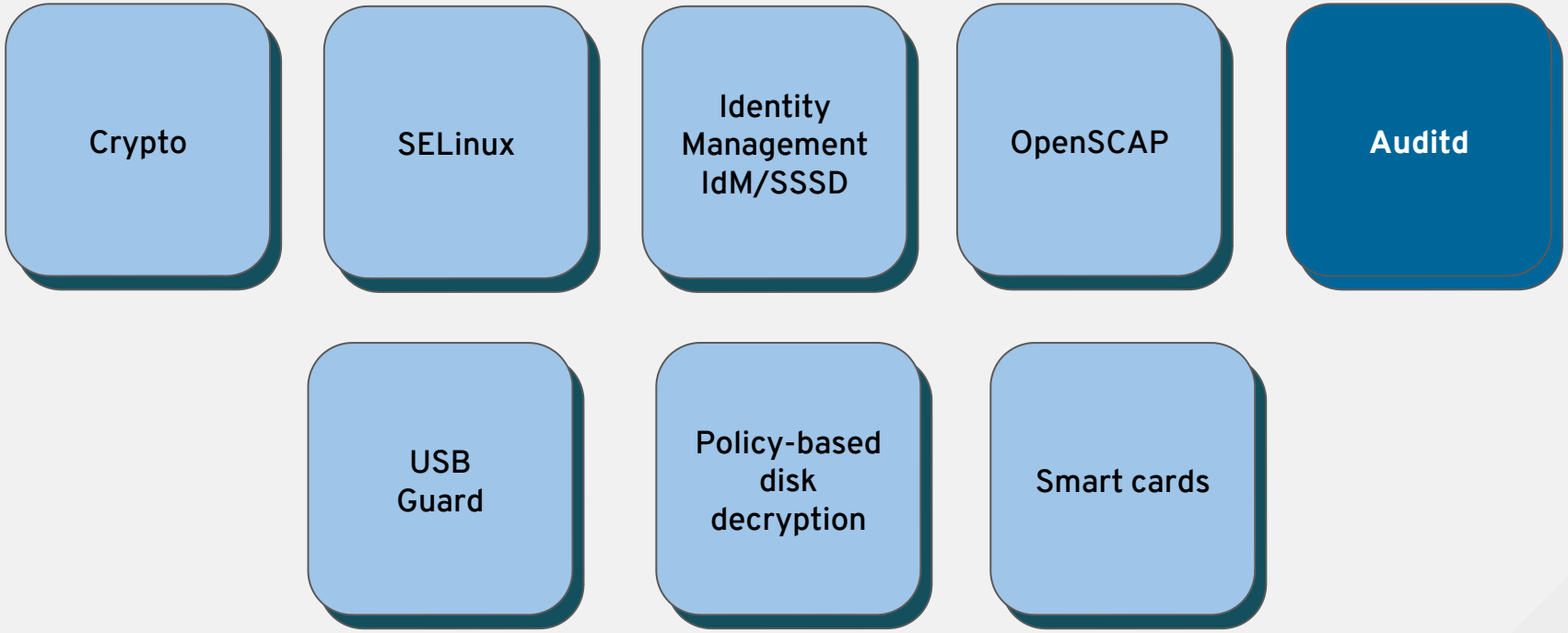


OpenSCAP

Integration



Security Technologies

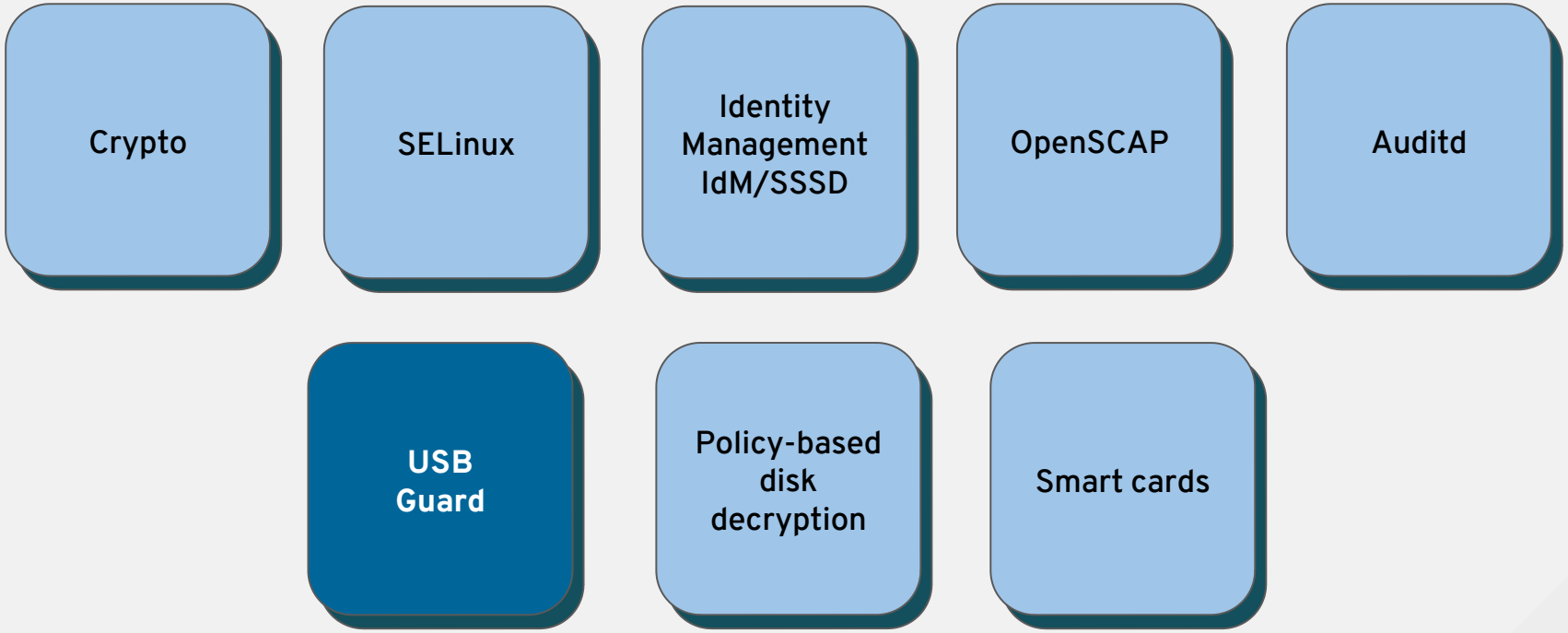


Auditd

Overview

- Low level kernel and system services trail of events
- Enriched to resolve IDs to names and normalized to provide human readable statements
- Multiple tools to dissect and query audit feed
- The foundation for auditing and compliance

Security Technologies



USB Guard

Overview

- As a **system administrator**, I want to define a USB device whitelist/blacklist so I can reduce the USB device driver attack surface.
- As a **system administrator**, I want to be able to define an implicit policy so that unknown devices are handled in a well defined way.

USB Guard

Overview

- USBGuard is a software framework for implementing USB device authorization policies (what kind of USB devices are authorized) as well as method of use policies (how a USB device may interact with the system)
- Implemented as a user space daemon and a set of CLI and GUI tools

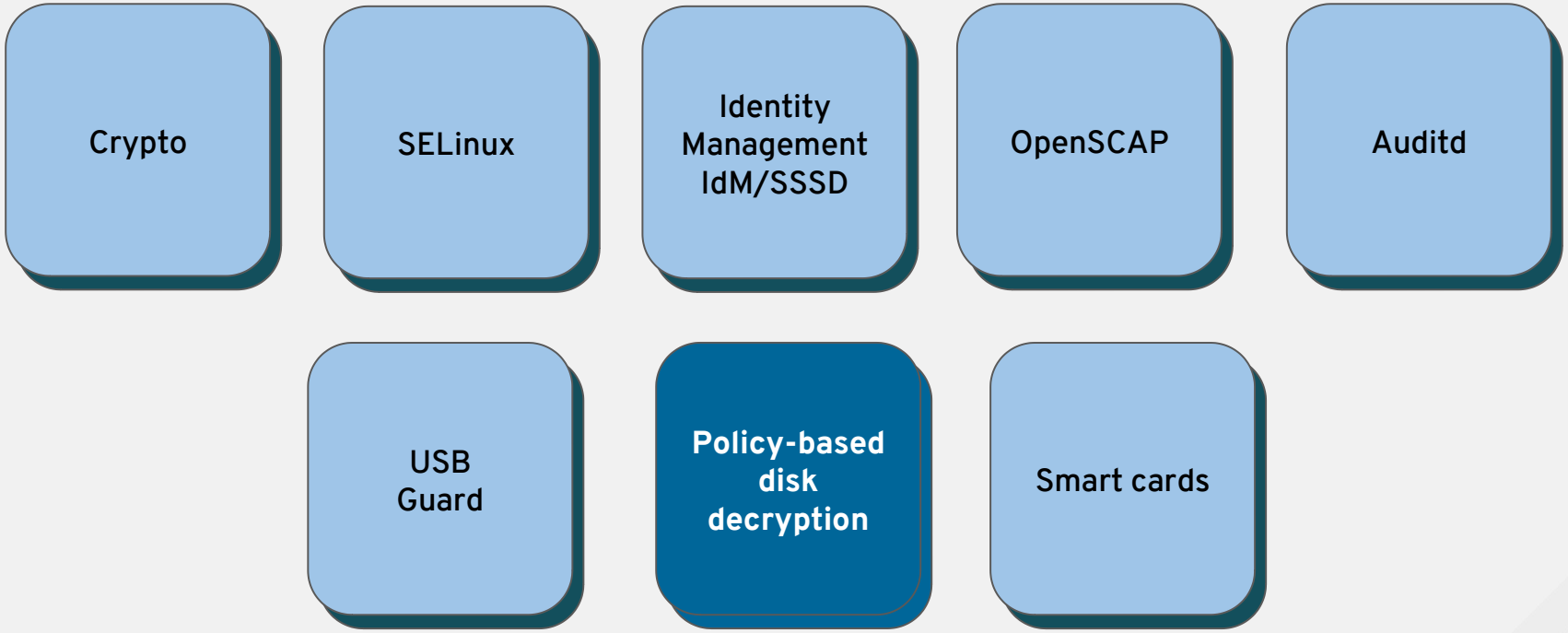
USB Guard

Policies

- USBGuard provides a rule language to describe a policy
- Examples

- Allow keyboard-only USB device only if there isn't already a USB keyboard attached:
  - `allow with-interface one-of { 03:00:01 03:01:01 } if !allowed-matches(with-interface one-of { 03:00:01 03:01:01 })`
- Allow a specific Yubikey device on a specific USB port, reject anything else on that port:
  - `allow 1050:0011 name "Yubico Yubikey II" serial "0001234567" via-port "1-2" hash "044b5e168d40ee0245478416caf3d998"`
  - `reject via-port "1-2"`

Security Technologies



Policy-based disk decryption

Overview

- Technology that allows decryption of a key using a very flexible policy
- A first application is Network Bound Disk Encryption

Network bound disk encryption

Why does it matter?

- Problem:
  - I need to encrypt hard drives because my security policy or compliance baseline requires me to do so
  - I can't use LUKS as is because if my datacenter reboots in the middle of the night I do not want to go and type the password to unlock the drive
- Solution:
  - Many 3rd-party solutions, but they depend on a complex infrastructure
  - Clevis/Tang in RHEL 7.4+ allows an easy answer (see a talk later today)
    - Clevis is a client-side framework to bind an encryption of a data to a source of truth

Clevis

What is it?

- Pluggable framework for automated decryption
- Contains a number of “pins” that implement decryption policies
- CLI to decrypt data using these pins
- Allows to bind a pin to a LUKS volume
  - Have different LUKS unlockers:
    - Dracut
    - Udisk2
    - clevis LUKS unlock

Chassis-bound disk encryption

What is it?

- Using TPM 2.0 to tie encrypted material to the physical device
- Trusted Platform Module 2.0: standard for a secure cryptoprocessor
  - Random number generator
  - Secure genreation of cryptographic keys for limited uses
  - Remote attestation
  - Binding of a data: encryption using the TPM bind key
  - Sealing: binding + a specific TPM state to decrypt the data

Chassis-bound disk encryption

What is it?

- Clevis pin for TPM 2.0 integration added in RHEL 7.6
- Allows to bind an encryption key of a LUKS volume to a key in TPM
  - LUKS volume cannot be decrypted outside of the chassis

THANK YOU


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHatNews


youtube.com/user/RedHatVideos