Strong crypto defaults in RHEL 8 and deprecation of weak crypto algorithms

Updated February 10 2021 at 7:17 PM - English

TABLE OF CONTENTS

Red Hat Enterprise Linux includes several cryptographic components whose security does not remain constant over time. Algorithms such as (cryptographic) hashing and encryption typically have a lifetime after which they are considered either too risky to use or plainly insecure. That means we need to phase out those algorithms from the default settings, or completely disable them if they cannot be used securely at all. While in the past we did not disable algorithms in a consistent way (different applications utilized different policies), today we have a system-wide policy which is followed by all RHEL crypto core components. These policies allow us to consistently handle and deprecate algorithms system-wide. Decisions about policy contents are based on documents like RFC 7457 which gives a list of attacks taking advantage of legacy crypto algorithms.

The following text lists, in detail, the algorithms that are either completely removed or disabled in the different crypto policy levels in RHEL-8. For a high level description of the change see this blog post and this kbase article.

For further details, please see the crypto-policies manual page.

What policies are provided?

Four policies are provided under the names “LEGACY”, “DEFAULT”, “FUTURE” and “FIPS”. They are summarized and described in the table below.

Policy name	**Description**
LEGACY	This policy ensures maximum compatibility with legacy systems; it is less secure 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	The DEFAULT policy is a reasonable default policy for today's standards, aimed for a balance between usability and security. 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. The purpose of the policy is for testing infrastructure and applications for their readiness for future strengthening of requirements. The policy is not supposed to be used for general purpose systems. 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 mode.

Removed ciphersuites and protocols

These ciphersuites and protocols are completely removed from the core crypto libraries. They are either not present at all in the sources or their support is disabled during the build so it cannot be used by applications in any way.

- DES (since RHEL-7)
- All export grade ciphersuites (since RHEL-7)
- MD5 in signatures (since RHEL-7)

- SSLv2 (since RHEL-7)
- SSLv3 (since RHEL-8)
- All ECC curves < 224 bits (since RHEL-6)
- All binary field ECC curves (since RHEL-6)

Disabled in all policy levels

These ciphersuites and protocols are available but disabled in all crypto policy levels. They can be enabled only by explicit configuration of individual applications.

- DH with parameters < 1024 bits
- RSA with key size < 1024 bits
- Camellia
- ARIA
- SEED
- IDEA
- Integrity only ciphersuites
- TLS Ciphersuites using SHA-384 HMAC
- AES-CCM8
- all ECC curves incompatible with TLS 1.3, including secp256k1
- IKEv1 (since RHEL-8)

Disabled in DEFAULT policy, but enabled in LEGACY policy

These ciphersuites and protocols are disabled in the DEFAULT crypto policy level. They can be enabled by switching the system crypto policy level to LEGACY.

- 3DES
- RC4
- DH with parameters < 2048 bits
- RSA with key size < 2048 bits
- DSA (all key sizes)
- TLSv1.0
- TLSv1.1

Disabled in the FIPS policy in addition to the DEFAULT policy

The FIPS policy allows only FIPS approved or allowed algorithms. It must be used when the system is required to be FIPS compliant. It is automatically selected when enabling the system FIPS mode.

- SHA1 in digital signatures
- RSA key exchange
- X25519, X448, Ed25519 and Ed448
- Chacha20-Poly1305

Disabled in the FUTURE policy, but enabled in the DEFAULT policy

The FUTURE policy provides additional hardening of the system. It is a conservative security level that is believed to withstand any near-term future attacks. The policy is not supposed to be used for general purpose systems.

- CBC mode ciphersuites in TLS
- Symmetric ciphers with smaller keys than 256 bits

- SHA-1 and SHA-224 signatures in certificates
- DH with parameters < 3072 bits
- RSA with key size < 3072 bits

Please note that most of the current WWW site certificates use just 2048 bits RSA keys so it will not be possible to connect to most of the public WWW sites with this policy.

Remediation of most common issues missing algorithms and protocol support

- Switching to the LEGACY policy can be done by issuing the command `update-crypto-policies --set LEGACY` from the root account.
- DH with small parameters - either switch to the LEGACY policy, or fix the TLS server to provide at least 2048 bit DH parameters, or use ECDH.
- SSLv3 - fix the TLS server to provide TLSv1.2 protocol (or at least TLSv1.0 protocol and switch to the LEGACY policy).
- 3DES, RC4 - either switch to the LEGACY policy or fix the TLS server to provide AES based ciphersuites.
- RSA with small keys, DSA - either switch to the LEGACY policy or generate new keys (with at least 2048 bits but preferably more) and certificates for the TLS server.
- TLSv1.0, 1.1 - either switch to the LEGACY policy or update the TLS server to provide TLSv1.2 protocol support.
- SHA1 in signatures in FIPS mode - regenerate the certificates to use SHA256 based signatures.
- Camellia, SEED, or ARIA - if the application configuration allows for modification of the ciphersuite configuration string, follow the documentation of the application to add these ciphersuites to the configuration string. Or remove the symlink to crypto policy configuration of the appropriate crypto library in /etc/crypto-policies/back-ends and replace it with configuration that will include the needed ciphersuite. Note that any modification described above, is unsupported by Red Hat. In the future we intend to provide a method create and use custom policies.

- IKEv1 - To opt out from the policy, comment the line including /etc/crypto-policies/back-ends/libreswan.config from /etc/ipsec.conf and add ikev2=never to the libreswan connection configuration.

Product(s) **Red Hat Enterprise Linux** **Category** **Secure**

Tags **rhel_8** **security** **Article Type** **General**

9 Comments

LOG IN TO COMMENT

RK

NEWBIE

7 Points

27 October 2019 5:38 PM

Roland Keller

What is the recommended way to set this policies with anisble?

SS

RED HAT

NEWBIE

17 Points

11 February 2021 2:20 PM

Simo Sorce

Lukas, you may be interested in following this work: https://github.com/linux-system-roles/crypto_policies/

TM

RED HAT

NEWBIE

10 Points

1 November 2019 12:32 PM

Tomas Mraz

I do not know if there are any specialized ansible modules. But you can of course execute update-crypto-policies --set ... with it.

LF

6 October 2020 2:14 PM

Lars Fuerst

NEWBIE

5 Points

You can do something like

```
- name: get current crypto policy
   shell: update-crypto-policies --show
   register: current_crypto_policy
   changed_when: false

- name: set crypto policy
  shell: "update-crypto-policies --set "
  when: current_crypto_policy.stdout != crypto_policy
```

By the way, did anyone noticed that the system wide crypto policy seem to overwrite a specific service configuration? I tried to disable sha1 based alogrithms in the ssh configuration by defining the Ciphers, MACs and KexAlgorithms specifically in the ssh config, but the ssh daemon still uses the default set defined by the system wide crypto policies.

SS

RED HAT

NEWBIE

17 Points

6 October 2020 2:51 PM

Simo Sorce

crypto-policies for ssh are implemented via an include file. So you need to make sure your changes will override the default by placing them *before* the include file that pulls in the policies.

MV

NEWBIE

10 Points

29 September 2021 12:59 PM

Marco Verschuur

Why is RedHat actively discouraging the use of the FUTURE crypto policy? FUTURE policy description: The policy is not supposed to be used for general purpose systems.

SS

RED HAT

NEWBIE

17 Points

29 September 2021 2:16 PM

Simo Sorce

The FUTURE crypto policy is very restrictive and most applications will fail to work properly with it, or may have too much of a performance hit without real benefits. This is why we do not recommend it for production. The idea of

providing the FUTURE crypto policy is for testing applications with what a future policy *could* be, so people can be ready in case some new crypto analysis requires to toughen the default crypto policy to fend off new threats.

NEWBIE

10 Points

1 October 2021 10:48 AM

Marco Verschuur

But how is it then possible that a CIS hardening standard 'dictates' that for Red Hat either FIPS or FUTURE must be used and at the same time Red Hat as a vendor interper FUTURE policy for not suitable for production usage? I may see a valid performance reason i.r.t. strong cryptographics for embedded systems with very low CPU power, but is this actually a real issue for general purpose servers? Is it really a valid point to disencourage SHA-256 over insecure SHA-1 and 3072 bits RSA keys over 2048 bit keys, for the reason of handling these cryptographic schemas will lower overall system performance? The current problem is that consultants of Red Hat refer to this technote as 'not suitable for production', while at the same time the customer is bound to strict regulatory, like the requirement for CIS hardening.

I think Red Hat must not write "not supposed to be used for general purpose" but should write something like "When using FUTURE policy be aware of some performance impact in certain scenarios and be aware of the impact on client systems as well." FUTURE policy is a non-default, so the fact that one chooses to implement the FUTURE policy is already a conscious choice, so one already has considered switching towards this policy. Implementing security hardening in companies is always difficult, because people need to adapt, and adaptation is one of the hardest tasks for companies (and most people). Therefor, writing "not supposed to be used" is definitely not really helping in such adaptation

COMMUNITY MEMBER

40 Points

2 July 2022 1:20 AM

Timofey Andreevskiy RHCE

In my humble opinion, current update-crypto-policies is ugly! (Sorry)

Let's say I have two ansible roles and two additional modules for crypto-policy. One is already in the default modules (for example AD-SUPPORT), the second is specific to our organization, configures the ssh daemon, and residues in

separated file. Each role set one of these modules for absolutely different purposes and in absolutely different times. One role is for IdM, second is for ssh hardening. Should I say what problem I will facing? I can.

1. The sequence of roles and the combination of two (or more) modules with the current policy in one end policy of the server. But we only have --show and --set. We must to conjure.
2. Idempotency. No one of these settings is idempotent.
3. If we change the module file itself, the policy must be reloaded. That is, we need again to conjure with the --show and the --set.
4. If we have changed the policy, but we are not allowed to restart the server, we still have to restart the services.

This is my current example ansible block for working with crypto-policies:

```
- name: Get current policy
  command: 'update-crypto-policies --show' #command!!! long live
bashsible!
  register: curcrypto
  changed_when: False
- name: Non skip if unsetted policy
  debug:
    msg: ''
  when: polname not in curcrypto.stdout
  changed_when: True
  notify: sshd restart
- name: Set policy list
  set_fact:
    pollist: '{%if polname not in curcrypto.stdout %}:{%endif%}'
- name: Policy copy
  copy:
    src: '.pmod'
    dest: /etc/crypto-policies/policies/modules/
    owner: root
    group: root
    mode: '0640'
  notify: sshd restart
- name: Set policy
  command: 'update-crypto-policies --set ' #bashsible again
  changed_when: False
```

and similar blocks must be inserted into any role that works with crypto-policies. I don't know how to make it easier.

This all is quite close to being very inconvenient to use. It seems to me, that it would be much better if the update-crypto-policy would not be so oaky as if out of the last century. For example, update-crypto-policy could have options --reload and --add. And --add may be idempotent. Itself. Without my dances.

Also, the need to restart the Unix server looks very funny. In the 21st century. Even Microsoft has already learned not to reboot its operating system for any reason. But we are going the other way. Let's reboot the consolidated database server...

Well, the internal content of policy modules is, of course, a separate story. Absolutely different designations in the module and regulated services with the absence of fine-tuning of algorithms in some cases - this, of course, is very convenient, thank you! And all OS operators must also be specialists in crypto systems, of course. Apparently this should be added to the rhcsa course! Because at present, even the Red Hat' technical support cannot adequately help when opening a case.