Custom crypto policies by examples

Tomáš Mráz
Principal SW Engineer, Red Hat
What we’ll be discussing today

Motivation

Crypto policies

Custom crypto policies

Examples

Future

Summary
Motivation
Motivation

Cryptography and cryptanalysis go hand in hand and the evolution of algorithms and protocols is faster and faster.

You can never think that a crypto system deployed in year $X$ will be still good enough in year $X+1$. We need to get used to changes.
Motivation
I: Introduction

1. Audience

Symptoms, symptoms, symptoms. They are a force multiplier.

2. Related publications


3. How to read this guide

This guide aims to accommodate two needs: first of all, having a handy reference owing to configure the most common services’ crypto settings and second of all, explains a lot of background on cryptography. This background is essential if the reader wants to choose his/her own cipher settings.

System administrators who want to cope & tune recommendations quickly without spending a lot of time on background reading are strongly encouraged to use the simplified references for the corresponding sections in...
Motivation

What if you need to apply the crypto-related configuration changes regularly to hundreds of machines physical and virtual in heterogenous environment?
To complicate things even more – various machines have various levels of need to communicate with legacy systems and devices. Everyone cannot accommodate every change.
Crypto policies come to rescue
System-wide crypto policies come to rescue

Centrally managed on the system

Multiple pre-designed policy levels

FIPS support simplification
Centrally managed on the system

Single command:

```
update-crypto-policies --set <LEVEL>
```
Centrally managed on the system

Controls:

- OpenSSL
- GnuTLS
- NSS
- Java
- Kerberos 5
- Bind
- OpenSSH client
- OpenSSH server
- libssh
- libreswan
When the `update-crypto-policies` command is run it transforms a simple policy definition into separate configuration file snippets that are loaded or included into default configurations of the supported backends.
## Multiple pre-designed policy levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LEGACY</strong></td>
<td>Legacy devices interoperability, RC4, 3DES</td>
</tr>
<tr>
<td></td>
<td>&gt;= 64bit security</td>
</tr>
<tr>
<td><strong>DEFAULT</strong></td>
<td>Reasonable but interoperable default</td>
</tr>
<tr>
<td></td>
<td>&gt;= 80bit security</td>
</tr>
<tr>
<td><strong>NEXT</strong></td>
<td>Fedora only equivalent of RHEL-8 DEFAULT</td>
</tr>
<tr>
<td></td>
<td>Removes TLS-1.0, 1.1, requires DH &gt;= 2048 bits</td>
</tr>
<tr>
<td><strong>FUTURE</strong></td>
<td>Conservative level, no SHA1, 256 bit ciphers only</td>
</tr>
<tr>
<td></td>
<td>&gt;= 128bit security</td>
</tr>
<tr>
<td><strong>FIPS</strong></td>
<td>FIPS approved/allowed algorithms only</td>
</tr>
<tr>
<td></td>
<td>&gt;= 112bit security</td>
</tr>
</tbody>
</table>
Simple command to enable FIPS mode

Just run:

```
fips-mode-set --enable
reboot
```
Simple command to enable FIPS mode

RHEL 7 for comparison:

```
yum install dracut-fips
dracut -f
<your-favourite-command-to-edit-boot-cfg>
reboot
```
System-wide crypto policies come to rescue

**Centrally managed on the system**
Single command controls all the core crypto libraries and applications using crypto.

**Multiple pre-designed policy levels**
Up-to-date security, communication with legacy systems, preparation for future

**FIPS support**
Simplify FIPS enablement

**Where?**
Current Fedora and Red Hat Enterprise Linux 8
But what to do if the pre-defined policy levels do not match your requirements?
Custom crypto policies come to rescue
Custom crypto policies

Define your own crypto policy from scratch

Or modify the existing pre-defined policy levels
Defining full policy from scratch

Placement of the full policy definition files:

/etc/crypto-policies/policies
/usr/share/crypto-policies/policies

The file needs to be named <POLICY>.pol (the upper case in the file name is important).
Simple policy definition format

See crypto-policies(7) manual page for all the key and algorithm names:

```plaintext
hash = SHA2-256 SHA2-384 SHA2-512 \n      SHA3-256 SHA3-384 SHA3-512 SHA2-224
group = X25519 X448 SECP256R1 SECP384R1 \n        SECP521R1 FFDHE-3072 FFDHE-4096 FFDHE-6144 \n        FFDHE-8192
min_tls_version = TLS1.2
min_rsa_size = 3072
```

excerpt from /usr/share/crypto-policies/policies/FUTURE.pol
Modification of existing policies by policy modifier modules

Placement of the policy modifier files:

/etc/crypto-policies/policies/modules
/usr/share/crypto-policies/policies/modules

The module file needs to be named <MODULE>.pmod (the upper case in the file name is again important).
Policy modifiers

Disable SHA1 hash:

```
hash = -SHA1
sign = -RSA-PSS-SHA1 -RSA-SHA1 -ECDSA-SHA1
```

/usr/share/crypto-policies/policies/modules/NO-SHA1.pmod

The hash value affects other use than signatures.
How to apply it?

Generate and set the customized policy:

```
update-crypto-policies --set DEFAULT:NO-SHA1
```

Any policy modifier module can be applied to other policies as well.
How to apply it?

Generate and set the customized policy:

```
update-crypto-policies --set FUTURE:NO-SHA1
```

So this can be used as well, although it would not be too useful.
Enable Camellia ciphers with priority to them:

```plaintext
tls_cipher = +CAMELLIA-128-CBC +CAMELLIA-128-GCM \\
+ CAMELLIA-256-CBC +CAMELLIA-256-GCM
cipher = +CAMELLIA-128-CBC +CAMELLIA-128-GCM \\
+ CAMELLIA-256-CBC +CAMELLIA-256-GCM
```

You can put this file for example into

/etc/crypto-policies/policies/modules/CAMELLIA.pmod
Policy modifiers

Enable Camellia ciphers but leave them last:

```plaintext
tls_cipher = CAMELLIA-256-GCM+ CAMELLIA-256-CBC+ \ 
CAMELLIA-128-GCM+ CAMELLIA-128-CBC+
cipher = CAMELLIA-256-GCM+ CAMELLIA-256-CBC+ \ 
CAMELLIA-128-GCM+ CAMELLIA-128-CBC+
```
Policy modifiers

Disable old TLS protocol versions:

```
protocol = -TLS1.1 -TLS1.0 -DTLS1.0
min_tls_version = TLS1.2
min_dtls_version = DTLS1.2
```

Some back-ends do not allow disabling protocol versions selectively. The min_tls_version value applies to them.
Policy modifiers

Allow smaller DH parameters and RSA keys in FUTURE policy:

```
# Parameter sizes
min_dh_size = 2048
min_rsa_size = 2048
```
Policy modifiers

Allow only ECDHE and ECDHE with PSK key exchanges:

\[
\text{key\_exchange} = -\text{RSA} - \text{DHE} - \text{DHE-RSA} - \text{PSK} - \text{DHE-PSK}
\]

Unfortunately the current version does not allow completely overriding a particular list value in policy modifier module.
Policy modifiers

Allow only ECDHE and ECDHE with PSK key exchanges:

```
key_exchange = ECDHE ECDHE-PSK
```

So this would not work properly currently.
Policy modifiers

Multiple modifiers can be applied:

```
update-crypto-policies --set DEFAULT:NO-SHA1:CAMELLIA
```
Configure-time generation of back-end configurations

The back-end configuration files in /etc/crypto-policies/back-ends are generated when update-crypto-policies is being run. This allows modifying the crypto libraries and/or the configuration generators in regards to the supported algorithms. Even completely new back-ends could be added in future and the policy will be still applied to them without need for modification.
Configure-time generation of back-end configurations

Example: OpenSSL back-end could allow more fine-grained configuration of TLS signature algorithms in future update. Or it could allow different behavior in regards to SHA1 signatures in TLS protocol vs. certificate signatures.
Custom crypto policies

Define your own crypto policy from scratch
In a simple policy definition file

Or modify the existing pre-defined policy levels
By adding or removing enabled algorithms or protocols

Generation of back-end configurations
When update-crypto-policies is run.
Future
What's in the works?

Handling of SHA1 deprecation
After the recent collision attack improvements the SHA1 use really needs to be abandoned.

More fine-grained back-end configurations
GnuTLS already improved, OpenSSL should follow.

Crypto policies and data at rest
We need to think about this.
Summary
Crypto policies simplify management of crypto on system with custom crypto policies allowing adjustments according to your needs.

- Single command to rule them (algorithms and libraries) all
- Multiple pre-designed policy levels
- Custom policies can be created from scratch or by policy modification
- Simple policy definition format
Thank you

https://gitlab.com/redhat-crypto/fedora-crypto-policies