GnuTLS config file format change for crypto-policies (2021)

Motivation and requirements

The current algorithm override mechanism has the following limitations:

- It does not provide a means to revert algorithm disablement; while it helps reduce attack surface, it can be problematic for legacy applications
- It is based on blocklisting, relying on the library defaults or build-time configuration. When the library defaults change, the end result can be unpredictable

To mitigate those limitations, a new configuration mechanism is proposed to provide:

- Fully revertible algorithm overrides
- Allowlist based construction of algorithm overrides

Existing configuration parameters

The following factors come into play when GnuTLS is used as a crypto library:

- The library defaults; embedded in the library source code
- The system wide override, set through a configuration file (on Fedora, `/etc/crypto-policies/back-ends/gnutls.config`)

In addition, the following factors take effect when used in TLS context:

- The default priority string, which is set at build time through a `--with-default-priority-string` (in Fedora “@SYSTEM”), shall be respected with the `gnutls_priority_init2` interface (new in 3.6.3)
- The user-supplied priority string, either with the `gnutls_priority_set*` functions (that require explicit “@SYSTEM:” at the beginning), or `gnutls_priority_init2`

Proposed solution

- Provide an alternative way of system wide override, based on **allowlisting** instead of blocklisting; this can be designated with a new option in the configuration file (o `verride-mode = allowlisting` in the `[global]` section); this mode is mutually exclusive to the existing blocklisting mode, i.e., any keywords available in the blocklisting mode cannot be used in allowlisting mode
- In the allowlisting mode, **all the algorithms are initially disabled but can be enabled later** with either a configuration file, a priority string (in TLS), or new API functions (for non-TLS uses)
- In the allowlisting mode, the “@SYSTEM” keyword is given a special meaning; if it is not defined in the `[priorities]` section, it is automatically constructed from the

configuration file. For example, if SECP256R1 is allowed in the configuration file, the user can assume “+GROUP-SECP256R1” is part of “@SYSTEM”

- In the allowlisting mode, applications can enable/disable algorithms at run time, while it is still prohibited in the blocklisting mode. The following API functions are added for that purpose:
  - `gnutls_ecc_curve_mark_disabled`
  - `gnutls_ecc_curve_mark_enabled`
  - `gnutls_sign_mark_insecure`
  - `gnutls_sign_mark_secure`
  - `gnutls_digest_mark_insecure`
  - `gnutls_digest_mark_secure`
  - `gnutls_protocol_mark_disabled`
  - `gnutls_protocol_mark_enabled`
- **This might change in the future revisions:** GnuTLS does not try hard to resolve algorithm dependencies. For example, when ECDSA-SHA256 is allowed as a signature algorithm in the configuration file, but SHA256 is not enabled as a hash algorithm, the TLS client will still advertise ECDSA-SHA256 while it cannot perform signing or verification with that algorithm

Examples

Example 1: Allow configuration similar to SECURE192

```
[global]
override-mode = allowlisting

[overrides]
enabled-version = TLS1.3
enabled-version = TLS1.2
enabled-curve = SECP384R1
enabled-curve = SECP521R1
secure-hash = SHA384
secure-hash = SHA512
secure-sig = ECDSA-SHA384
secure-sig = ECDSA-SHA512
secure-sig = RSA-SHA384
secure-sig = RSA-SHA512
secure-sig-for-cert = ...
tls-enabled-cipher = AES-256-GCM
```

```
tls-enabled-cipher = CHACHA20-POLY1305
tls-enabled-cipher = AES-256-CBC
tls-enabled-cipher = AES-256-CCM
tls-enabled-group = SECP384R1
tls-enabled-group = SECP521R1
tls-enabled-group = FFDHE8192
tls-enabled-kx = ECDHE-ECDSA
tls-enabled-kx = ECDHE-RSA
tls-enabled-kx = RSA
tls-enabled-kx = DHE-RSA
tls-enabled-mac = AEAD
```