Repository Security

Repository Security Improvements

As identified in Previous Repository Security Proposals by several authors, there is a need to improve the security of the Maven repository ecosystem. We have an opportunity to encourage users to consider security (and later other aspects such as licensing) by default, which fits well with Maven's view of development best practices.

Requirements

1. Verifiable downloads are the default behaviour for Maven 2.1+
2. Existing repository clients are supported (even if unsecure)
3. Allowing the use of unsigned artifacts should be simple, but done with full knowledge by the user
4. Different techniques for signing artifacts should be supported.
5. Non-Java artifacts must be supported
6. We must be able to support repositories not controlled by Maven itself
7. POMs must also be signed to ensure that unverified dependencies or repositories are not injected
8. All techniques must apply equally to other downloadable code - plugins, archetypes, extensions, parent POMs as well as dependencies
9. Additional configuration should not be required to run Maven and its distributed plugins

Out of Scope Requirements

There may be a need to reconsider how Maven determines the repositories to use to make this more feasible. For example:

- Users may want to be more selective about the list of repositories that can be used
- It will be desirable to retrieve signatures and keys from a verifiable source but large artifacts from mirrors.

However, this is not in scope for the initial implementation.

Current Work

The current work has been done on the matter:

- Commons OpenPGP (used by Wagon)
- Wagon OpenPGP

The feature branch in Maven is:

- Maven Artifact
- Maven trunk
- Maven site

Related JIRA issue(s): http://jira.codehaus.org/browse/MNG-2477

Proposed Solutions

OpenPGP

Keyrings

Maven will use the following keyrings:

- one in the Maven installation (`$M2_HOME/conf/pubring.pgp`, configured in the installation settings file)
- one in the user's Maven directory (`$HOME/.m2/pubring.pgp`, configured in the installation settings file)
- others added to `settings.xml` (see below)

The Maven installation will contain the keys of several well known individuals and repositories. Should a user not wish to accept this initial set of keys, they can simply remove the installation key ring and manually install keys they wish to trust. As described later, a trust store and automatic retrieval from a key server is not used, but is a future consideration.

The user's Maven directory will initially not include any keys, however Maven will gradually add accepted keys there. There are two possible ways this can be done:

1. Maven should provide a plugin for adding keys to the user's keyring (with configuration to allow adding to different keyrings)
2. Use of standard gpg command line tools

Maven will make no specific requirement on how keys are to be created. It is expected that many remote repositories will use a "role key" for signing all artifacts in the repository for convenience reasons, and that the key is adequately protected. However, some repositories may choose to allow artifacts to be signed by individuals, in which case the Maven user will need to add the key of each individual release manager to their public key ring (this is likely to be the case for the ASF repo initially).

Adding keys

A plugin will be available from Maven that will aid in adding keys to the user's key ring(s).

Keys may be added via a KEYS file, or via a keyserver.

All keys that are retrieved will need to be confirmed first, displaying the details including fingerprint.

Error messages in the artifact resolution will specifically direct the user in how to use the given plugin to add a key once it is known to belong to a project (providing the project's web site to show where to validate the fingerprint).

This will be present in the super POM using a checksum so that a key is not required to use it, avoiding the problem of not having the key necessary to add the key needed to use the plugin.

Signature Files

Maven will retrieve signature files from the same location as the artifact and other metadata. Note that, as above, the actual keys will not be retrieved from the mirror but must be added manually from the source or using a keyserver.

Signature files can contain one or more signatures (either concatenated, or a single ascii-armored message with multiple packets). The artifact is accepted if **any** of the signatures are found to be valid.

Handling Artifacts without Signatures

Initially, many of the artifacts in the repository will be unsigned. In addition, we will continue to receive upload requests that will not (and should not) sign artifacts. POMs written to be used against Maven 2.0.x should behave as designed (ie, with modelVersion 4.0.0).

To accommodate this, we should add configuration to the repository definitions, for example:

```
<repository>
  ...
  <releases>
    <signaturePolicy>warn</signaturePolicy> <!-- can be fail (default), warn or ignore -->
  </releases>
  <snapshots>
    <signaturePolicy>warn</signaturePolicy> <!-- can be fail, warn or ignore (default) -->
  </snapshots>
</repository>
```

Alternatively, a specfic checksum can be specified on the dependency (see below).

Measures should be put in place in the repository sync to ensure new artifacts all receive signatures.

There are two alternatives to implementing handling for unsigned artifacts.

Separation of central repository between valid-signed artifacts and unsigned artifacts

We should split the central repository along artifacts that are validly signed and those that are not:

- `http://repo1.maven.org/central/signed` - default included repository with signed, valid artifacts
- `http://repo1.maven.org/central/unsigned` - repository with legacy unsigned artifacts that can be included by users via their build or repository manager (with appropriate white listing) as desired
- `http://repo1.maven.org/maven2` - rewrite rules onto the other repositories for backwards compatibility

This requires some work on the repository side and so has not been executed yet. In the interim, signature checking is off by default (though in prototyping it was able to be turned on using the technique below).

Signing of artifacts (Not recommended)

To balance backwards compatibility while still allowing a strong default mode, old artifacts can be signed with a specific key issued by the Maven PMC. This will not be included in the distribution by default, but can easily be added by users that want to explicitly trust these artifacts en masse. This key can also be used for unsigned uploads via JIRA.

This was used for evaluation during prototyping without impacting the repository itself. It is still an option, but carries the concern of giving a false sense of security by signing artifacts that are not released by us and have never been validated.

Apache Incubator

The Incubator will sign with key(s) that are not included in the Maven distribution by default but that can be easily added. Note that incubator releases must not be signed by an individual user since trusting that user for other releases would circumvent this.

Snapshots

As shown above, by default the policy for snapshots is to not require or check for a signature. It is possible to enable this if necessary.

Specified Checksums

While it is inconvenient, we should support specification of an exact checksum of the JAR to use. This could potentially be baked in to releases to help prevent or at least identify reproducibility problems in the event of a repository accident as well as malicious replacement. This would be specified as such:

```
<dependency>
...
  <requiredChecksum algorithm="sha1" value="01234abcdef..." pom="01234abcdef..." />
</dependency>
```

Rules:

- `sha1` would be the default algorithm.
- The POM checksum is optional (it will be verified using the normal PGP rules if not specified)
- Only one instance of the requirement need be given in the tree and any others would assume the same requirement.
- Should two specify different checksums (with the same algorithm), then Maven should fail to resolve the artifact.
- Should two specify checksums with different algorithms, both will be verified
- Maven should still be able to be configured to disable this check, and also whether to check it every run, or only when an artifact is first downloaded
- Maven can use the checksum in the local repository to avoid needing to recalculate it for later verification
- If this technique is used for an artifact, the PGP signature will no longer be checked.

Handling POM changes

Maven 2.0.x is able to handle the `requiredChecksum` element in a dependency as specified as it is an empty element, but not the `signaturePolicy` element. Neither are handled when building a project under 2.0.x - this is acceptable at this point.

We will initially add the elements under the "4.0.0" `modelVersion` on trunk until it handles multiple model versions, where it should be increased to "4.1.0".

On deployment, the `requiredChecksum` element can be left intact, but the `signaturePolicy` element should be removed and the modelVersion set to 4.0.0.

Key Revocation

Without the use of a keyserver initially, we will rely on Maven upgrades to revoke pre-distributed keys.

In addition, since old binaries continue to be used over time, once a key is revoked they won't be usable. At this point a user would need to upgrade, or use the known checksum or a specific exclusion for the dependency. Care needs to be taken that there wasn't a specific reason for the revocation that the artifact shouldn't be used.

Other resources

References

- http://wiki.debian.org/SecureApt
- http://www.cs.arizona.edu/people/justin/packagemanagersecurity/

Comments

- http://www.jroller.com/robertburrelldonkin/entry/maven_repository_security_comments
- http://www.jroller.com/robertburrelldonkin/entry/maven_repository_security
- http://hiramchirino.com/blog/2008/07/comments-on-maven-repository-security.html
- http://hiramchirino.com/blog/2008/08/new-checksum-plugin.html

Potential Future Features

The following are not needed initially, but could be considered for the future.

Per-artifact settings

By default, all keys in the user's keyring will be accepted. However, to strengthen a requirement, a project may require that the key be a particular one, or signed by a particular key. The key provided might be either an email address or the key ID.

```
<dependency>
  <validSigners>
    <validSigner>
      <keyId>123ABCD</keyId> <!-- can also be an email address -->
      <trustChain>false</trustChain> <!-- whether to trust artifacts signed by keys that are signed by this
one, default is true. If false, only trust artifacts signed directly by this key -->
    </validSigner>
  </validSigners>
</dependency>
```

Also, in line with the other repository enhancements, the signature policy (and other checksum and update policies) should also be available on the dependency element.

Get Signature Files from the source

Maven's repository system does not currently easily support locating the original source repository of an artifact in a secure way. In the event that the above is not available, Maven will still fall back to retrieving the signature from the repository specified, ignoring any requested mirrors.

However, we should consider implementing improved repository location, in conjunction with the other work on improving repository handling. One alternative may be to look into a Repository Switchboard. It may actually be desirable for users to declare all required repositories themselves, eliminating the ability to retrieve them from the POM (particularly transitively), or at least only using this as a suggestion. The user would then opt-in to repositories, possibly mapping them via the repository manager instead.

In any case, more direction towards the source repository should be balanced by the ability to Mirror Repositories conveniently and access them from Maven.

Web of Trust

Initially, the user will be required to explicitly add keys to their keyring that they accept. However, it could become more flexible by downloading from keyservers and choosing to trust a certain key, and also participating in a web of trust.

```
<settings>
   ...
  <openpgp>
    <keyrings>
      <keyring>${user.home}/.gnupg/pubring.gpg</keyring>
    </keyrings>
    <keyservers>
      <keyserver>http://pgp.mit.edu/</keyserver>
    </keyservers>
  </openpgp>
</settings>
```

Maven should prompt when an artifact is downloaded with an untrusted key, if it is in interactive mode. If it is accepted, then the key is added to the user's key ring.

The key should be downloaded from a trusted keyserver (as specified in `settings.xml` above), not from the repository that the artifact is being downloaded from.

The other settings driven keyrings are so that a user can participate in a web of trust (eg, going to an ApacheCon keysigning) and reuse that to trust certain keys.

In addition, this may be used to aid in the management of keys where individuals sign a release but whether it is used is managed by a single repository key and the trust values associated with it.

JAR Signing

Maven should support JARs using the Java security model without the need for additional verification.

Central repository management

Enhancements should be made shortly after availability of this feature to ensure that all new artifacts accepted to the central repository are signed. The unsigned repository can continue to be used for those that are not.