Background

I encountered an issue related to building when using the java-tron project, which involves the process of dependency checksum validation. During the project build, it is common to check the checksum of downloaded dependencies to ensure the integrity of the files. However, I found that there were some issues with the checksum validation process, as it was not being performed.
Firstly, Gradle does not always perform checksum validation when downloading dependencies. It directly downloads the files without going through the checksum validation step. This raised concerns about the security and integrity of the project since we couldn't ensure that the downloaded dependencies were not tampered with or corrupted.
Therefore, I wanted to address this issue and improve the functionality of dependency checksum validation in future versions. My suggestion was to always perform checksum validation during the download of dependencies and provide more detailed error messages to assist developers in diagnosing issues.
There are two options for checksum validation:

1. Introducing a plugin
2. Using Gradle for validation, requiring an upgrade to Gradle

Implementation

Using a plugin

The checksum-dependency-plugin is a plugin-based dependency validation tool based on version 5.x. It can generate checksums and PGP signatures. To generate checksums, use the command: ./gradlew allDependencies -PchecksumUpdate. Here is a partial example of the generated content:

```
None
<?xml version='1.0' encoding='utf-8'?>
<dependency-verification version='2'>
```

```
 <trust-requirement pgp='GROUP' checksum='NONE' />
 <ignored-keys />
 <trusted-keys>
  <trusted-key
id='475f3b8e59e6e63aa78067482c7b12f2a511e325'
group='ch.qos.logback' />
  <trusted-key
id='b1b8eef3122e5988b08bd82e4796783adad489f0'
group='com.alibaba' />
  <trusted-key
id='cfdc233f20258133a454ef7a40f96c6798309f48'
group='com.aliyun' />
   ...
 </trusted-keys>
 <dependencies>
  <dependency group='antlr' module='antlr'
version='2.7.7'>

<sha512>311C3115F9F6651D1711C52D1739E25A70F25456CACB9A2
CDDE7627498C30B13D721133CC75B39462AD18812A82472EF1B3B9D
64FAB5ABB0377C12BF82043A74</sha512>
  </dependency>
  <dependency group='aopalliance' module='aopalliance'
version='1.0'>

<sha512>3F44A932D8C00CFEEE2EB057BCD7C301A2D029063E0A916
E1E20B3AEC4877D19D67A2FD8AAF58FA2D5A00133D1602128A7F509
12FFB6CABC7B0FDC7FBDA3F8A1</sha512>
  </dependency>
  <dependency group='com.beust' module='jcommander'
version='1.72'>
```

```
<sha512>2DA86589FF7ECBB53CFCE845887155BCA7400ECF2FDFDEF
7113EA03926A195A1CBCF0C071271DF6BEDC5CDFA185C6576F67810
F6957CD9B60AB3600A4545420E</sha512>
  </dependency>
   ...
 </dependencies>
</dependency-verification>
```

When the correct checksum of a dependency package is modified, an exception will be thrown, indicating the checksum/PGP violations detected during the resolution configuration.

Using Gradle for validation

Gradle has supported dependency verification since version 6.2. To enable this feature, the java-tron project needs to be upgraded to version 6.2 or higher. Gradle supports MD5, SHA1, SHA-256, and SHA-512 checksums. However, currently, only SHA-256 and SHA-512 checksums are considered secure. Dependency verification consists of two separate and complementary operations:

1. Checksum verification, which allows asserting the integrity of dependencies.
2. PGP signature verification, which allows asserting the source of dependencies.

Here is an example of the generated checksums:

```
None
<?xml version="1.0" encoding="UTF-8"?>
<verification-metadata
xmlns="https://schema.gradle.org/dependency-verificatio
```

```
n"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://schema.gradle.org/dependenc
y-verification
https://schema.gradle.org/dependency-verification/depen
dency-verification-1.1.xsd">
  <configuration>
   <verify-metadata>true</verify-metadata>
   <verify-signatures>false</verify-signatures>
  </configuration>
  <components>
   <component group="antlr" name="antlr" version="2.7.7">
     <artifact name="antlr-2.7.7.jar">
      <sha256
value="88fbda4b912596b9f56e8e12e580cc954bacfb51776ecfdd
d3e18fc1cf56dc4c" origin="Generated by Gradle"/>
     </artifact>
     <artifact name="antlr-2.7.7.pom">
      <sha256
value="100f793ba27f8b4e4204edb46171ebf36e54e0f94cfc0252
7fea07a0bb1fceb7" origin="Generated by Gradle"/>
     </artifact>
   </component>

  </components>
</verification-metadata>
```

One thing to note is the source of the checksums. Where do we obtain the checksums from?
Typically, Gradle fetches checksums from both public repositories and official websites to validate the correctness of dependencies. This is a common security

practice to prevent incorrect results due to tampering at a single node. Therefore, during the verification process, you will usually see two log entries for the JAR and POM files. If either of them produces an exception, the process will be aborted.
Example log message:

```None
> Dependency verification failed for configuration
':compileClasspath':
  - On artifact javaparser-core-3.6.11.jar
(com.github.javaparser:javaparser-core:3.6.11) in
repository 'MavenRepo': Artifact was signed with key
'8756c4f765c9ac3cb6b85d62379ce192d401ab61' (Bintray (by
JFrog) <****>) and passed verification but the key isn't in
your trusted keys list.
```

NOTE

Both approaches mentioned above can address the issue of dependency verification. The choice of approach will require different modifications to the java-tron project structure. However, dependency validation is a crucial task and should be implemented for enhanced security.