

Ant Technology

Mobile Security Armor User Guide

Document Version: 20240808




Legal disclaimer

Ant Group all rights reserved ©2022.

No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Ant Group.

Trademark statement

 蚂蚁集团
ANT GROUP and other trademarks related to Ant Group are owned by Ant Group. The third-party registered trademarks involved in this document are owned by the right holder according to law.

Disclaimer

The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Ant Group reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through channels authorized by Ant Group. You must pay attention to the version changes of this document as they occur and download and obtain the latest version of this document from Ant Group's authorized channels. Ant Group does not assume any responsibility for direct or indirect losses caused by improper use of documents.

Document conventions

Style	Description	Example
 Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: If the weight is set to 0, the server no longer receives new requests.
 Note	A note indicates supplemental instructions, best practices, tips, and other content.	 Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings> Network> Set network type .
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click OK .
Courier font	Courier font is used for commands	Run the <code>cd /d C:/window</code> command to enter the Windows system folder.
<i>Italic</i>	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>
{ } or {a b}	This format is used for a required value, where only one item can be selected.	<code>switch {active stand}</code>

Table of Contents

1.About Mobile Security Armor -----	05
2.Terminology -----	06
3.Price introduction -----	07
4.New Tool Chain for Android Application Security Hardening -----	08
4.1. Getting started -----	08
4.2. user-guide -----	08
4.3. View security hardening logs -----	08
5.Security hardening for Android apps -----	10
5.1. Instructions for use -----	10
5.2. Quick start -----	10
5.3. User guide -----	10
5.3.1. Create a security hardening task -----	10
5.3.2. View the security hardening list -----	12
5.3.3. Download a hardened package -----	12
5.3.4. View security hardening details -----	12
5.3.5. View a hardening failure log -----	13
5.3.6. Search for and delete a task -----	13
5.4. OpenAPIs -----	13
5.4.1. Preparations for API calls -----	13
5.4.2. API references -----	14
5.5. Troubleshooting after Android hardening -----	20
5.6. FAQs -----	21
6.Security hardening for iOS apps -----	23

1. About Mobile Security Armor

Mobile Security Armor(MSA) service provides stable, simple and effective security protection for mobile application (App), improves the overall security level of applications, and protects applications from being cracked and attacked. MSA is powered by the mobile security reinforcement techniques of Alibaba Cloud Group. It has been tested and tried by Taobao series Apps (the amount of users and data are over 100 millions). It guarantees high reliability in terms of security. When you use the MSA service to reinforce mobile Apps of mPaaS, you can use the hotfix feature as usual. In other words, you can use the hotfix feature to fix bugs in an online version without concerning about the security of Apps.

MSA can reinforce Android Apps, but does not support iOS Apps.

Background

The Android system is open source. Therefore, Android Apps are extremely vulnerable to attacks such as piracy and reverse engineering. This severely affects the data and privacy security of the Apps. To resolve the preceding issues, MSA is developed to reinforce the APK or AAB packages and perform compatibility testing and functional regression testing on the reinforced APK or AAB packages, which can protect the Apps from being cracked.

Benefits

• Comprehensive reinforcement capabilities

MSA can integrate reinforcement techniques against various application security vulnerabilities in APK and AAB packages, without changing the source code of Android Apps. The overall security of the Apps is reinforced, which ensures the Apps cannot be pirated and infringed.

• Excellent reinforcement performance

The impact of reinforcement on the size of APK and AAB package and application performance is strictly controlled. Therefore, the size and performance of the Apps are not significantly changed after reinforcement.

How it works

MSA enhances the anti-cracking capabilities of applications by using various techniques, such as recompiling Android Apps, adding shells for protection, and modifying the sequence of command calls. Reinforcement intensity and compatibility are balanced, which can avoid the unavailability of Apps due to the blind pursuit of reinforcement strength in general reinforcement product.

Features

MSA provides the following reinforcement capabilities:

• APK or AAB reinforcement:

To protect the overall security of APK or AAB packages, MSA provides overall SSH and anti-tamper protection for DEX files and various techniques for APK or AAB packages. These techniques include anti-decompilation, preventing white box attacks, shell encryption algorithms, anti-debugging, anti-memory tampering, anti-hooking, anti-emulation, anti-repackaging, and anti-memory dump.

• Class reinforcement:

MSA obfuscates Java code to hide the actual operational process. This prevents the code from being decompiled by using JADX-GUI and JEB tools. In addition, this makes reinforced code difficult to be read manually.

2.Terminology

Terminology	Interpretation
Security reinforcement package	Refers to the reinforced APK or AAB package. The security reinforcement package of a task refers to the APK or AAB package reinforced in the task.
Reinforcement	Improves the anti-cracking ability of the application by recompiling the application, protecting the application based on shell technology, and modifying the command calls sequence of the application.

3. Price introduction

Android price introduction

The Mobile Security Armor (MSA) service is billed in subscription mode. You must purchase the service before you can use it. If you have activated Mobile PaaS (mPaaS), you can enjoy a seven-day free trial of the MSA service since an APK or AAB package is uploaded. After the free trial period ends, the system notifies that the service has expired. To continue to use the service, you must [purchase the MSA service](#). For more pricing information of the MSA service, see [Prepaid](#).

An MSA service can harden only one application. The service is bound to the application based on the APK or AAB package name. When you upload the APK or AAB, the MSA service verifies the package name of the uploaded APK or AAB.

- If the APK or AAB package name has not been bound to an MSA service, and an MSA service is available during uploading, the system will bind the package name to the available MSA service. The binding process automatically occurs in the background without affecting the upload process.
- If the APK or AAB package name has not been bound to an MSA service, but no MSA services are available during uploading, the system displays a message to prompt you to purchase an MSA service.
- If the APK or AAB package name has been bound to an MSA service that has not expired, the upload process proceeds without interruption. When the service time is 30 days left, the system displays a message to indicate the remaining service time.
- If the APK or AAB package name has been bound to an MSA service that has expired, the system displays a message to indicate that the service has expired and a new MSA service needs to be purchased.

iOS price introduction

For iOS price issues, welcome to search group number 33417739 to join the DingTalk group for consultation.

H5 price introduction

For H5 price issues, welcome to search group number 33417739 to join the DingTalk group for consultation.

Note

The MSA service for applications is billed in subscription mode. Therefore, a purchased MSA service cannot be renewed. Ensure that a new MSA service is purchased and assigned to the application before the previous service is about to expire.

4. New Tool Chain for Android Application Security Hardening

4.1. Getting started

This topic describes how to use security hardening for Android to quickly harden an application and obtain a security hardening package.

Precondition

- An APK to be protected has been prepared. The APK should be unhardened and the APK size should be ≤ 300 MB.
- You have purchased the mobile security hardening service or are in the seven-day free trial period.

Procedure

The steps to use the new tool chain for security hardening are as follows:

1. Log on to the [mPaaS console](#) and select an application.
 2. In the left-side navigation pane, choose **Security > Mobile security armor > Android application security hardening**.
 3. Click **Create security hardening** go to the **Upload an application** page.
 4. Click **Upload an application** to upload an APK file.
 5. After the file is uploaded, you can confirm the application information and hardening information on the page of **Confirm security hardening information**.
 6. Turn on **Do you want to select a new tool chain for reinforcement**.
 7. **Select hardening capacity** as required (there will be detection protection when it is running).
 8. Optional. In the **Add classes that require security protection** section, select the classes that you want to reinforce and select important classes.
 9. Optional. In the **Select the So file to be protected** section, select the so file that you want to protect.
 0. Optional. In the **Select the Assets file to be protected** section, select the Assets file that you want to protect.
1. Click **Confirm hardening**.
 2. Return to the **Android application security hardening** page. A card for the security hardening task will be on the page that appears. you can view the hardening progress of the corresponding task in the card.
 - **Hardening**: indicates that the hardening task is in progress.
 - **Hardened**: The hardening task is completed.
 - **hardening Failed**: indicates that the hardening task failed.
 3. When the **Hardened** is displayed in the card, click the download icon (↓) to download the hardened APK or AAB file.

Note

The hardened installation package does not have signature information. You need to re-sign the downloaded hardened package and then release it to the corresponding application market.

Next steps

After hardening, be sure to check whether the functions of key components are normal, such as upgrade components and hot repair components. If the installation package fails to function after reinforcement, [submit a ticket](#) or contact the mPaaS technical helpdesk.

Note

If there are more access-related questions, please search group number 33417739 to join DingTalk group for consultation and exchange.

4.2. user-guide

For more information about how to use the new tool chain for security hardening, see [Security hardening for Android](#).

Note

It is not supported for AAB files to **Add classes that require security protection** or other files resource protection.

4.3. View security hardening logs

If the protection detects the corresponding risk while it is running, the app will exit after security hardening, and print a log as follows:

```
ashield process runtime info key ---> value
```

Key-value description

Feature	key	value
Signature verification	a0	Values greater than 0
Anti-hook	a1	Values greater than 0
Anti-debugging	a2	Values greater than 0

Anti-simulator	a3	Values greater than 0
Anti-root	a4	Values greater than 0
protection against SQL injection attacks	a5	Values greater than 0
Anti-opening	a6	Values greater than 0

5. Security hardening for Android apps

5.1. Instructions for use

Before you use Mobile Security Armor (MSA) to harden an APK or AAB file, ensure that the following requirements are met. To improve experience, read the following instructions before you use MSA.

- Ensure that the content of the onCreate function for the provider can be executed multiple times. If the onCreate function contains related logic, make sure the related logic can be executed at least twice. For example, if you want to initialize a single instance in the onCreate function for the provider, you need to check whether the instance has been initialized.
- The x86 and mips architectures are not supported. If there are related architectures configured, please remove them and repackage them for hardening.
- Currently minSdkVersion does not support 24 and above. Version 23 or earlier is recommended. If the value of the minSdkVersion is less than 23, MSA compresses and stores nativeLibraries in the APK file by default. If you need to set the value of the minSdkVersion equal to 23, you can perform one of the following operations:
 - In the `application` node, add `android:extractNativeLibs="true"`.
 - Repackage the hardened APK or AAB file and set whether compression is required based on your rules.

ⓘ Important

If you want to harden the Assets file in the application, you must ensure that minSdkVersion ≥ 21, that is, the Android version is not lower than 5.0.

5.2. Quick start

This topic provides guidance on how to use Mobile Security Armor (MSA) to harden Android apps with few steps and obtain hardened packages.

Prerequisites

- An APK or AAB that you want to harden is available. The APK or AAB must be not hardened and must be less than or equal to 300 MB.
- An MSA instance is purchased or within the seven-day free trial period.

Procedure

To use MSA to harden an Android app, perform the following steps:

1. Log on to the mPaaS console and select the target app.
2. In the left-side navigation pane, choose **Mobile application security** > **Application security hardening**. The **Application security hardening** page appears.
3. Click **Create security hardening**. The **Upload applications to be hardened** page appears.
4. Click **Upload an application** to upload an APK or AAB file.
5. After the file is uploaded, the page automatically jumps to the **Confirm security hardening information** page. You can confirm the app information.
6. (Optional) In the **Add classes that require security protection** column, select the classes that you need to harden.
7. Click **Confirm hardening** to harden the app.
8. Return to the **Application security hardening** page. The card for the hardening task is added. You can view the hardening progress of the task on the
 - **Hardening**: indicates that the hardening task is in progress.
 - **Hardened**: indicates that the hardening task is complete.
 - **Hardening failed**: indicates that the hardening task failed.
9. When the **Hardened** state is displayed on the card, click **Download** to download the hardened package, that is, the hardened APK or AAB file.

📌 Note

The hardened installation package does not contain signature information. You need to re-sign the downloaded hardened package and then release the re-signed hardened package in the app market.

Subsequent steps

After hardening is complete, make sure to check whether the key components such as the upgrade component and the hotfix component are properly functioning.

5.3. User guide

5.3.1. Create a security hardening task

App security hardening is to harden the entire app and the core classes. This topic provides guidance on the complete process of creating a security hardening task.

Mobile Security Armor (MSA) supports hardening of the following objects:

- **Overall APK or AAB file**: To protect the overall security of APK and AAB packages, MSA provides anti-decompilation protection for APK and AAB packages, overall SSH protection and anti-tamper protection for DEX files, defense against white-box attacks, SSH encryption algorithms, anti-debugging, anti-memory tampering, anti-hooking, anti-emulator, anti-repackaging for APK and AAB packages, and anti-memory dump.
- **Core classes**: obfuscates the Java code, hides the actual running process, and protects the code from decompilers such as jadx-gui and JEB, making the hardened code difficult to be read by humans.
- **So files**: Encrypt and protect So files to increase the difficulty and cost of cracking So files.
- **Assets files**: Encrypt and protect Assets resource files to meet regulatory requirements.

📌 Note

The hardening of the overall APK or AAB file is required. The hardening of the core classes, So files or Assets files is optional. The APK can be reinforced according to requirements.

Prerequisites

Before you start this task, you need to prepare the app that you need to harden. The following requirements must be met:

- The file name extension must be `.apk` or `.aab`.
- The app must not be hardened, because MSA does not support repeated hardening of hardened installation packages.
- The APK or AAB package has been signed. In the hardening process, anti-repackaging is performed on the APK or AAB file. Therefore, the uploaded app package needs to be signed.
- If you want to harden the Assets file in the application, you must ensure that `minSdkVersion` \geq 21, that is, the Android version is not lower than 5.0.
- The size of the APK or AAB file must be less than 300 MB.

Procedure

To create a hardening task, perform the following steps:

1. Log on to the mPaaS console and select the target app from the app list.
2. In the left-side navigation pane, choose **Mobile application security** > **Application security hardening**. The **Application security hardening** page appears.
3. Click **Create security hardening**. The **Upload applications to be hardened** page appears.
4. Click **Upload an application** to upload the installation package that you need to harden. In the upload process, you can click **Cancel upload** on the page to cancel the upload. The **Upload applications to be hardened** page returns to the initial state, that is, the state when the upload operation is not performed.

Note

When the uploaded APK or AAB file does not meet the requirements, the upload fails. After you click **Re-upload**, the **Upload applications to be hardened** page returns to the initial state.

5. After the file is uploaded, the page jumps to the **Confirm security hardening information** page. On this page, you need to perform the following operations:
 - **Confirm application information:** In the **Application information** column, view the app information.
 - App name
 - App package name
 - App version
 - App size
 - **Confirm hardening information:** In the **Hardening information** column, view the hardening services provided for the overall APK or AAB file.
 - Shell protection
 - AndroidManifest file tamper protection
 - Signed file protection
 - Anti-debugging protection
 - Anti-native application debugging
 - Anti-memory dump protection
 - Anti-simulator run protection
 - Anti-Root Device Operation Protection
 - Anti-memory data read protection
 - Anti-memory data modification protection
 - Anti-hook attack protection
 - Anti-memory code injection protection
 - **Select Shell Mode:** **Quick Mode** is selected by default.
 - **Quick mode:** Apps packed in this mode start faster than apps hardened in compatibility mode, but crashes may occur on some Android models.
 - **Compatibility mode:** The startup speed of the application packed in this mode is slower than that of the application hardened in the fast mode, but the compatibility is higher, and the packed application generally does not appear abnormal during operation.

Note

It is recommended to use compatibility mode to pack the application.

- **Add classes that require security protection:** optional. To select the classes that you need to harden, perform the following steps:
 - (Optional) Enter a keyword for the class name, and click **Search** to search for the class. We recommend that you enter a complete class name to search. If more than 1,000 search results are found, no results are displayed on the platform. If this case occurs, you need to enter the complete class name to search again.
 - ii. Select one or more classes. Up to 300 classes are supported.

Note

The selected class names appear under the search box. You can click the cross sign (×) to clear the class on the same line as the cross sign.

- **Please select the So file to be protected :** Select the So file to be hardened, the operation method is as follows:

- Enter the keyword in the So file name and click **Search** to search for the target file.
- ii. Click the check box in front of the So file to be hardened to select one or more target So files.

Important

When selecting the So file to be hardened, it is not recommended to choose a third-party So file for hardening, because hardening a third-party So file to improve application security is of little significance and is prone to compatibility issues.

- Please select the Assets file to be protected: Select the Assets file that needs to be hardened, and the operation method is as follows:

- i. Enter keywords in the Assets file name and click **Search** to search for the target file.
 - ii. Click the check box in front of the Assets file to be hardened to select one or more target Assets files.
- Click **Confirm hardening**. When the message **App is hardening** is displayed on the page, the hardening task has been created. Click **View hardening list**. On the **Application security hardening** page, you can view the security hardening list. A card for the current task is already added to the list. In the card, you can view the hardening progress of the task and download the hardened APK or AAB file.

Follow-up operation

- [Download a hardened package](#)

5.3.2. View the security hardening list

The information about the created security hardening tasks is displayed as cards in the security hardening list. On a hardening task card, you can view the security hardening list. The security hardening list is arranged in descending order of the task creation time. The task cards display the following information:

- **Application name**
 - When the hardening is successful, the name is displayed in blue.
 - When the hardening fails or is being performed, the name is displayed in black.
- **Hardening status**

On the right side of **Application name**, the hardening status of the current task is displayed. The values of the hardening status include **Hardened**, **Hardening**, and **Hardening failed**.

 - **Hardened**: indicates that the current hardening task is successful.
 - **Hardening**: indicates that the current hardening task is in progress.
 - **Hardening failed**: indicates that the current hardening task failed.
- **Package name**: the name of the uploaded APK or AAB package.
- **Version number**: the version number of the app.
- **Application size**: the size of the APK or AAB before the hardening.
- **Creation time**: the time when the current task was created.

Procedure

To view the security hardening list, perform the following steps:

1. Log on to the mPaaS console and select the target app from the app list.
2. In the left-side navigation pane, choose **Mobile application security > Application security hardening**. On the **Application security hardening** page, you

Related topics

In a task card, in addition to viewing the app information and the hardening status, you can perform the following operations:

- [Download security hardened package](#)
- [View security hardening details](#)
- [View a hardening failure log](#)

5.3.3. Download a hardened package

A hardened package is the APK or AAB file that has undergone security hardening. This topic provides guidance on how to download a hardened package in a task from a hardening task card.

Note

For tasks in the **Hardening** or **Hardening failed** state, no entries to download hardened packages are provided on the cards.

To download a hardened package, perform the following steps:

1. Log on to the mPaaS console and select the target app from the app list.
2. In the left-side navigation pane, choose **Mobile application security > Application security hardening**. The **Application security hardening** page appears.
3. In the security hardening list, click **Hardened package** on the target task card to download the hardened package in the task.

Important

- After the download is complete, make sure to check whether the key components such as the upgrade component and the hotfix component are properly functioning.
- The signature information of the app is deleted during the hardening process. Therefore, you need to re-sign the downloaded hardened package and then release the re-signed hardened package in the app market.

If the installation package works abnormally after hardening, please search for the group number 31591197 with DingTalk to join DingTalk group for further communication.

5.3.4. View security hardening details

After you create a hardening task, you can view the security hardening details of the task.

- **Basic information**: displays the information about the hardened app, including **Application name**, **Application package name**, **Application version**, and **Application size**. Note that the app size is the size of the app before the app is hardened.
- **Hardened package**: provides an entry to download the hardened package.

Important

The signature information of the app is deleted during the hardening process. Therefore, you need to re-sign the downloaded hardened package and then release the re-signed hardened package in the app market.

- **Hardening details**: displays the comparison of app details before and after hardening in terms of **Application size**, **MD5**, and **Security**.

- **Hardened class**: displays hardened classes and the comparison between the code before and after the hardening.
- **Hardened So files**: Show hardened So files.
- **Hardened Assets Files**: Display hardened Assets files.

Note

When creating a hardening task, if you choose to harden classes, So files, and Assets files, you can see the details of the **security hardened classes**, **security hardened So files**, and **security hardened Assets files** on the **security hardening details** page of the task.

Procedure

You can view the hardening details of tasks in the **Hardened** state. For tasks in the **Hardening** or **Hardening failed** state, no entries to view hardening details are provided. To view the hardening details, perform the following steps:

1. Log on to the mPaaS console and select the target app from the app list.
2. In the left-side navigation pane, choose **Mobile application security > Application security hardening**. The **Application security hardening** page appears.
3. In the security hardening list, click the app name on the target task card. On the **Security hardening details** page, you can view the hardening details.
4. (Optional) To view the hardened class, move the pointer to the question mark (🔍) that follows **Security hardening class** to view the comparison between the code before and after the hardening.

Note

If decompilation fails, the code screenshots before and after the hardening are blank.

5.3.5. View a hardening failure log

For tasks in the hardening failed state, you can download hardening failure logs. For tasks in the **Hardening** or **Hardened** state, no entries to view hardening failure logs are provided.

To download a hardening failure log, perform the following steps:

1. Log on to the mPaaS console and select the target app from the app list.
2. In the left-side navigation pane, choose **Mobile application security > Application security hardening**. The **Application security hardening** page appears.
3. In the security hardening list, find the target hardening task and click the **Download failed logs** icon (📄) in the upper-right corner of the task card to download the hardening failure log.

5.3.6. Search for and delete a task

In the security hardening list, you can search for and delete hardening tasks.

Search for a task

You can search for a task in the security hardening list by app name or package name. To search for a task, perform the following steps:

In the search box in the upper-right corner of the security hardening list, enter the keyword for the app name or package name. The system searches the security hardening list based on the entered content in real time.

Delete a task

To delete a hardening task, perform the following steps:

1. Log on to the mPaaS console and select the target app from the app list.
2. In the left-side navigation pane, choose **Mobile application security > Application security hardening**. The **Application security hardening** page appears.
3. In the security hardening list, click **Delete** on the target task card. In the dialog box that appears, click **OK** to delete the task.

5.4. OpenAPIs

5.4.1. Preparations for API calls

Hardening of mobile apps can be implemented by calling OpenAPIs of Mobile Security Armor (MSA). The OpenAPIs connect the server on the user side to the server of the Mobile PaaS (mPaaS).

Procedure

To call OpenAPIs of MSA, perform the following steps:

1. Query the token uploaded to Object Storage Service (OSS).
2. Upload the APK or AAB to OSS.
3. Notify MSA of the uploaded APK or AAB.
4. Query the upload result by initiating a polling task and obtain the ID of the hardening task.
5. Instruct MSA to start hardening.
6. Query the hardening result by initiating a polling task and obtain the URL of the hardened app.
7. Download the hardened package.

Rate limiting and throttling

To prevent the overuse of the OpenAPIs from affecting the operating of apps, a rate limiting and throttling mechanism is implemented for calls to the OpenAPIs. The following content describes the specific mechanism:

- OpenAPIs of MSA adopt a single-instance rate limiting and throttling mechanism. The mechanism is implemented based on the `appid` and `workspaceId` fields.
- MSA provides two devices to receive API requests, which are then forwarded by using Server Load Balancer (SLB).

- In a single MSA instance, the API for uploading app packages can be called up to 10 times per minute, that is, once every 6 seconds. The remaining MSA APIs can be called up to 600 times per minute, that is, once every 0.1 seconds.

Preparations

Before you call an API, you must obtain the AccessKey pair, the app ID, the workspace ID, and the tenant ID, configure Maven dependencies, and configure a file upload.

Obtain the AccessKey pair

An AccessKey pair includes an **AccessKey ID** and an **AccessKey Secret**. For more information about how to obtain an AccessKey pair, see [Obtain an AccessKey pair](#).

- **AccessKey ID**: identifies a user.
- **AccessKey Secret**: the secret used to authenticate the user. Keep the secret confidential.

Obtain the app ID, workspace ID, and tenant ID

1. Log on to the [mPaaS console](#) and click the app.
2. On the **Overview** page, click **Code configuration** and select Android or iOS as needed. Then, click **Download configuration file** and click **Download now**. In the **Code configuration** panel, you can view the app ID, workspace ID, and tenant ID.

Configure Maven dependencies

Before you call an API, you must configure Maven dependencies. The following code shows sample configurations.

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-mpaas</artifactId>
  <version>3.0.3</version>
</dependency>

<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <optional>true</optional>
  <version>[4.3.2,5.0.0)</version>
</dependency>
```

Upload a configuration file

File streams are not allowed in all APIs. Therefore, to upload a file, you must invoke an upload tool class to upload the file to OSS. Then, you must pass the returned OSS address as a parameter to the specified API.

You can download the file upload tool class [OssPostObject.java.zip](#).

Examples

To view an example of how to use MSA APIs, see [mpaas-msa-client.zip](#).

5.4.2. API references

This topic describes the open APIs of Mobile Security Armor (MSA).

Query the token of an uploaded file

Request - GetFileTokenForUploadToMsaRequest

Parameter	Type	Description
appld	String	The app associated with the request.
workspaceId	String	The workspace to which the app belongs.
tenantId	String	The tenant to which the app belongs.
onexFlag	Boolean	The value is fixedly set to <code>true</code> .

Response - GetFileTokenForUploadToMsaResponse

```
{
  "resultContent": {
    "content": {
      "accessid": "LTAI7z7XPfKU****",
      "dir": "mds/tempFileForOnex/ONEXE9B092D/test/PUQYHL/8b574cb7-3596-403f-a0e9-208660fc2081/",
      "expire": "1584327372",
      "host": "https://mcube-test.oss-cn-hangzhou.aliyuncs.com",
      "policy": "QwM2YtYTB1OS0yMDg2NjBmYzIwODEvI1ldfQ==",
      "signature": "kisfP5YhbPtmES8+w="
    },
    "resultMsg": "",
    "success": true
  },
  "requestId": "8BAA3288-662E-422C-9960-2EEBFC08369F",
  "resultCode": "OK"
}
```

Response parameters

Parameter	Type	Description
requestId	String	The ID of the request.
resultCode	String	If <code>OK</code> is returned, the request is successful. If other codes are returned, the API request failed.
ResultContent.Content	Object	The returned object. For more information about the meanings of the fields in the object, see the following table.

The following table lists the fields contained in the returned object and describes the meanings of the fields.

Parameter	Type	Description
resultMsg	String	The value returned if the query failed.
success	Boolean	Indicates whether the query is successful.

Instruct MSA to start processing an app that is uploaded to OSS

Request - UploadUserAppToMsaRequest

Parameter	Type	Description
appId	String	The app associated with the request.
workspaceId	String	The workspace to which the app belongs.
tenantId	String	The tenant to which the app belongs.
fileUrl	String	The URL of the uploaded APK or AAB.

Response - UploadUserAppToMsaResponse

```
{
  "resultContent": {
    "data": {
      "id": 12345,
      "enhanceTaskId": 12345,
      "progress": 10,
      "status": 0
    },
    "resultMsg": "",
    "success": true
  },
  "requestId": "637D5BE0-0111-4C53-BC EE-473CFFA0DBAD",
  "resultCode": "OK"
}
```

Response parameters

Parameter	Type	Description
requestId	String	The ID of the request.
resultCode	String	If <code>OK</code> is returned, the request is successful. If other codes are returned, the API request failed.
resultContent	Object	The returned object. For more information about the meanings of the fields in the object, see the following table.

The following table lists the fields contained in the returned object and describes the meanings of the fields.

Parameter	Type	Description
data.id	String	The ID of the upload task. If hardening is in progress, you need to query the ID by initiating a polling task.

data.enhanceTaskId	String	The hardening task ID returned after the upload is complete. This ID is used to start a hardening task.
data.status	Integer	The status of the upload task. Valid values: -1: failed. 0: processing. 1: uploaded
resultMsg	String	The value returned if the query failed.
success	Boolean	Indicates whether the query is successful.

Query the upload status of an app

Request - GetUserAppUploadProcessInMsaRequest

Parameter	Type	Description
appId	String	The app associated with the request.
workspaceId	String	The workspace to which the app belongs.
tenantId	String	The tenant to which the app belongs.
id	Long	The ID of the upload task.

Response - GetUserAppUploadProcessInMsaResponse

```
{
  "resultContent": {
    "data": {
      "id": 12345,
      "enhanceTaskId": 12345,
      "progress": 10,
      "status": 0
    },
    "resultMsg": "",
    "success": true
  },
  "requestId": "637D5BE0-0111-4C53-BCEE-473CFFA0DBAD",
  "resultCode": "OK"
}
```

Response parameters

Parameter	Type	Description
requestId	String	The ID of the request.
resultCode	String	If <code>OK</code> is returned, the request is successful. If other codes are returned, the API request failed.
resultContent	Object	The returned object. For more information about the meanings of the fields in the object, see the following table.

The following table lists the fields contained in the returned object and describes the meanings of the fields.

Parameter	Type	Description
data.id	String	The ID of the upload task. If hardening is in progress, you need to query the ID by initiating a polling task.
data.enhanceTaskId	String	The hardening task ID returned after the upload is complete. This ID is used to start a hardening task.
data.status	Integer	The status of the upload task. Valid values: -1: failed. 0: processing. 1: uploaded
resultMsg	String	The value returned if the query failed.
success	Boolean	Indicates whether the query is successful.

Start a hardening task

Request - StartUserAppAsyncEnhanceInMsaRequest

Parameter	Type	Description
appId	String	The app associated with the request.
workspaceId	String	The workspace to which the app belongs.
tenantId	String	The tenant to which the app belongs.
id	Long	The ID of the hardening task.
taskType	String	The type of the task. Valid values: <code>shell</code> : adds shells. <code>enhance_shell</code> : uses Java2C.
classes	String	The core classes that you want to harden by using Java2C. We recommend that you add critical core classes only. You can separate the classes by commas (.). For example, set the value to <code>com.a.a,com.b.b</code> . If you specify this parameter, ensure that <code>taskType</code> is set to <code>enhance_shell</code> . However, this parameter is not applicable to some classes. If you set this field for these classes, hardening may fail.
totalSwitch	boolean	The total switch that specifies whether to enable the task. To use a switch subordinate to the total switch, you must set this parameter to <code>true</code> .
javaHook	Integer	The anti-hooking technique at the Java layer. Valid values: 0: killself. 1: warning
memoryDump	Integer	The anti-memory dump technique. Valid values: 0: killself. 1: warning
emulatorEnvironment	Integer	The anti-emulator technique. Valid values: 0: killself. 1: warning
nativeHook	Integer	The anti-hooking technique at the native layer. Valid values: 0: killself. 1: warning
dalvikDebugger	Integer	The anti-debugging technique at the Java layer. Valid values: 0: killself. 1: warning
nativeDebugger	Integer	The anti-debugging technique at the native layer and the rooting technique. Valid values: 0: killself. 1: warning

Response - StartUserAppAsyncEnhanceInMsaResponse

```

{
  "resultContent": {
    "data": {
      "afterMd5": "aaaaaaaa",
      "afterSize": 1000,
      "appCode": "ONEXxxxx",
      "appPackage": "com.example.app",
      "beforeMd5": "bbbbbb",
      "id": 1,
      "label": "Alipay",
      "progress": 0,
      "status": 2,
      "taskType": "shell",
      "versionCode": 1,
      "versionName": "1.0.0",
      "enhancedClasses": ["aaa", "bbb"]
    },
    "resultMsg": "",
    "success": true
  },
  "requestId": "F9C681F2-6377-488D-865B-1144E0CE69D2",
  "resultCode": "OK"
}

```

Response parameters

Parameter	Type	Description
requestId	String	The ID of the request.
resultCode	String	If OK is returned, the request is successful. If other codes are returned, the API request failed.

resultContent	Object	The returned object. For more information about the meanings of the fields in the object, see the following table.
---------------	--------	--

The following table lists the fields contained in the returned object and describes the meanings of the fields.

Parameter	Type	Description
resultMsg	String	The value returned if the query failed.
success	Boolean	Indicates whether the query is successful.
data.afterMd5	String	The MD5 checksum of the hardened APK or AAB.
data.afterSize	Long	The size of the hardened APK or AAB.
data.id	Long	The ID of the hardening task. This ID is used for subsequent polling.
data.label	String	The labels of the APK or AAB. The value is the same as that of the label field of the APK or AAB.
data.progress	Integer	The progress of the hardening process. Valid values: 0 to 100
data.status	Integer	The status of the hardening task. Valid values: 0: not started. 1: task submitted. 2: hardening. 3: hardened. 4: hardening failed
data.taskType	String	The type of the hardening task.
data.enhancedClasses	String	The class selected for Java2C hardening.

Query the progress of a hardening task

Request - GetUserAppEnhanceProcessInMsaRequest

Parameter	Type	Description
appld	String	The app associated with the request.
workspaceId	String	The workspace to which the app belongs.
tenantId	String	The tenant to which the app belongs.
id	Long	The ID of the hardening task.

Response - GetUserAppEnhanceProcessInMsaResponse

```

{
  "resultContent": {
    "data": {
      "afterMd5": "aaaaaaaa",
      "afterSize": 1000,
      "appCode": "ONEXxxxx",
      "appPackage": "com.example.app",
      "beforeMd5": "bbbbbb",
      "id": 1,
      "label": "Alipay",
      "progress": 0,
      "status": 2,
      "taskType": "shell",
      "versionCode": 1,
      "versionName": "1.0.0",
      "enhancedClasses": ["aaa", "bbb"]
    },
    "resultMsg": "",
    "success": true
  },
  "requestId": "F9C681F2-6377-488D-865B-1144E0CE69D2",
  "resultCode": "OK"
}

```

Response parameters

Parameter	Type	Description
requestId	String	The ID of the request.
resultCode	String	If OK is returned, the request is successful. If other codes are returned, the API request failed.
resultContent	Object	The returned object. For more information about the meanings of the fields in the object, see the following table.

The following table lists the fields contained in the returned object and describes the meanings of the fields.

Parameter	Type	Description
resultMsg	String	The value returned if the query failed.
success	Boolean	Indicates whether the query is successful.
data.afterMd5	String	The MD5 checksum of the hardened APK or AAB.
data.afterSize	Long	The size of the hardened APK or AAB.
data.id	Long	The ID of the hardening task. This ID is used for subsequent polling.
data.label	String	The labels of the APK or AAB. The value is the same as that of the label field of the APK or AAB.
data.progress	Integer	The progress of the hardening process. Valid values: 0 to 100
data.status	Integer	The status of the hardening task. Valid values: 0: not started. 1: task submitted. 2: hardening. 3: hardened. 4: hardening failed
data.taskType	String	The type of the hardening task.
data.enhancedClasses	String	The class selected for Java2C hardening.

Query the download URL of the hardened product

Request - GetUserAppDownloadUrlInMsaRequest

Parameter	Type	Description
appId	String	The app associated with the request.
workspaceId	String	The workspace to which the app belongs.
tenantId	String	The tenant to which the app belongs.
taskId	String	The ID of the hardening task.

Response - GetUserAppDownloadUrlInMsaResponse

```
{
  "resultContent": {
    "data": { "url": "https://xxxx" },
    "resultMsg": "",
    "success": false
  },
  "requestId": "8F76783A-8070-4182-895D-14E5D66F8BA3",
  "resultCode": "OK"
}
```

Response parameters

Parameter	Type	Description
requestId	String	The ID of the request.

resultCode	String	If OK is returned, the request is successful. If other codes are returned, the API request failed.
checkRsaKeyResult	Object	The returned object. For more information about the meanings of the fields in the object, see the following table.

The following table lists the fields contained in the returned object and describes the meanings of the fields.

Parameter	Type	Description
data.url	String	The download URL of the APK or AAB.
data.filename	String	The file name of the APK or AAB.
resultMsg	String	The value returned if the query failed.
success	Boolean	Indicates whether the query is successful.

Query a hardening log

Request - GetLogUrlInMsaRequest

Parameter	Type	Description
appId	String	The app associated with the request.
workspaceId	String	The workspace to which the app belongs.
tenantId	String	The tenant to which the app belongs.
taskId	String	The ID of the hardening task.

Response - GetLogUrlInMsaResponse

```
{
  "resultContent": {
    "data": { "url": "https://xxxx"},
    "resultMsg": "",
    "success": false
  },
  "requestId": "8F76783A-8070-4182-895D-14E5D66F8BA3",
  "resultCode": "OK"
}
```

Response parameters

Parameter	Type	Description
requestId	String	The ID of the request.
resultCode	String	If OK is returned, the request is successful. If other codes are returned, the API request failed.
resultContent	Object	The returned object. For more information about the meanings of the fields in the object, see the following table.

The following table lists the fields contained in the returned object and describes the meanings of the fields.

Parameter	Type	Description
data	String	The download URL of the log.
resultMsg	String	The value returned if the query failed.
success	Boolean	Indicates whether the query is successful.

5.5. Troubleshooting after Android hardening

This topic describes how to troubleshoot crash issues occurring after Android Apps are hardened.

Note

If an Android App can run normally before hardening but crashes upon being started after hardening, the reason may be that the runtime threat detection of Mobile Security Armor (MSA) is detected. At this time, the MSA service will kill the Android App, that is, the Android App cannot be used normally. The crash caused by hardening may be the desired crash of the hardening policy for protection purposes.

Troubleshoot in the logs of MSA

You can search for the following keywords by filtering the logs generated when the Android App crashes upon being started:

DEFENDER

DEFENDER indicates that the logs are printed by the MSA service.

behavior

The handling methods are displayed after **behavior**:

- 0 indicates that the Android App exits, 1 indicates that the logs are printed, and 2 indicates a pop-up window is displayed.
- The reason for the crash is displayed after the keyword **message**.

jaffer

If **jaffer** exists, a signature problem may occur, for example, a resignature.

check your **xxxx** (This is the reason why a threat is detected).

Troubleshoot in the App crash logs

FATAL

FATAL indicates a fatal threat. The following stack information is the reason for the crash.

5.6. FAQs

Why does the error code EnhanceError exist in the reinforcement failure log?

Cause analysis:

In the `AndroidManifest.xml` file of the application, no entry class is declared for the `application` tag.

Solution:

In the `AndroidManifest.xml` file, declare an entry class for the `application` tag.

Why did the selected class fail to be reinforced?

Cause analysis:

Before Mobile Security Armor (MSA) reinforces a class, it evaluates the operational risks of the class by assuming that the class is reinforced. If the class is prone to operational risks after reinforcement, MSA automatically gives up reinforcing the class.

Solution:

MSA does not reinforce a class that is prone to operational risks after reinforcement.

6. Security hardening for iOS apps

This topic describes how to harden iOS apps using Mobile Security Armor (MSA). Before you harden iOS apps using MSA, you need to read the usage note

Usage notes

Before you harden iOS apps using Mobile PaaS (mPaaS) MSA, you need to read the following usage notes and ensure that your projects meet related req

- It is recommended that the relevant code to be hardened is written in C or C++. iOS hardening has better and more stable support for C and C++. In addition, it partially supports Objective-C and does not support Swift.
- Hardening will bring performance losses and increase theoretical stability risks. It is recommended to only harden the core code that needs to be protected, and extract the C and C++ code that needs to be protected into a separate Framework, and then harden it.
- Currently supports X86/M1 machines. You can select About This Mac from the Apple menu in the corner of the screen to view overview information about your Mac, including processor information. If it shows Intel processor, it means your Mac is an X86 architecture.
- Currently supports Xcode 14.1/14.2/15.0.1. Since iOS hardening processes the compiler and requires adaptation of specific Xcode, you need to use a specific version of Xcode when using iOS hardening.

Important

Starting April 29, 2024, apps uploaded to App Store Connect must be built using Xcode 15 for iOS 17, iPadOS 17, Apple tvOS 17, or watchOS 10.

- Please make sure that the App project's workspace is set to **New Build System**. The check path is **Xcode > File > Project Settings > Build System**.

Procedure

1. Configure environment files. Generate the `MSAConfig.json` file according to the following method, and put it in the `$HOME` directory. Open the command line on the Mac machine and enter `echo $HOME` to get the `$HOME` directory. When using it, replace it with the real value. The fields are as follows:

```
{
  "appId": "application appId",
  "workspaceId": "application workspaceId",
  "tenantId": "application tenantId",
  "accessKeyId": "Ant Cloud account accessKeyId",
  "accessKeySecret": "Ant Cloud account accessKeySecret",
  "license": "blank",
  "domain": "xxx"
}
```

Note

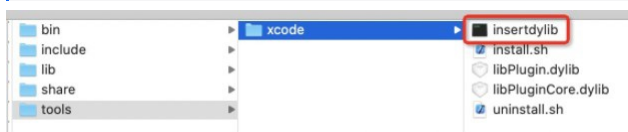
- The domain values are ap-southeast-1 and cn-hongkong, which correspond to Singapore and Hong Kong respectively.
- For how to obtain field values, please refer to [How to obtain iOS hardening configuration file information](#).

2. Install the hardening tool.

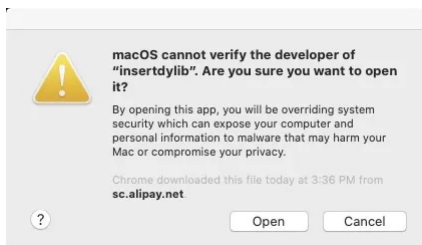
- i. Download the hardening tool, unzip it and go to the directory **tools> xcode**.

Note

- `xcodeplugin-x86_64-5.9.0.zip` is suitable for Xcode 15.0.1 + Mac X86.
- `xcodeplugin-arm64-5.9.0.zip` is suitable for Xcode 15.0.1 + Mac M1.
- `xcodeplugin-x86_64-5.7.2` is suitable for Xcode 14.1/14.2 + Mac X86 version.
- `xcodeplugin-arm64-5.7.2` is suitable for Xcode 14.1/14.2 + Mac M1 version.



- ii. Open the `insertdylib` file. In the Confirm dialog box, click **Open**.



- iii. Run the following command:

```
sh ./tools/xcode/install.sh
```

Note

After you run the command, the system automatically finds and replaces compilers in the `/Applications/Xcode.app/` directory. If you need to reset command.

3. Open the Framework or IPA project by using Xcode, and then run the `Build/Archive` command. A dynamic library is not supported for now.

Important

The project path name cannot contain space characters or Chinese characters. Otherwise, an error may occur in the compilation process.

4. (Optional)

After the above step is complete, check the hardening effects through decompilation. You can check the hardening effects by running the following command:

```
nm ./BinaryPath | grep obfuscator
```

Next steps

After hardening, please be sure to check whether the functions of the key components are normal. If the function of the installation package is abnormal after hardening, Please [submit a ticket](#) to contact mPaaS technical support.