Guide of MTP Shell Solution

INTRODUCTION

With this tool, MTP shell solution can be integrated into your continuous integration process and signature process of game. It can be automatically shelled and signed with one line of code.

The version adds so-export table confusion, V1/V2/V3 signature， custom shell version and AAB Shelling based on official version.

The version adds the shell to anti-plug-in SDK protection capability， New detection capabilities for dex files and so files in the lib directory and global-metadata encryption based on the new version.

Download the corresponding Command-line Tool for reinforcement.

Before using the tools to shell the games, you can read Developer Suggestions

Pay attention:

1. This tool only supports the Android version of the game.
2. Operating System Supported：Windows / MAC OS / Linux (For Windows and Linux systems, x86 and x64 versions are provided separately. Please confirm the operating system configuration of your computer before selecting the tool version.)
3. When the tool is running, the game will be authenticated first. MTP will issue a cert format certificate for each game. If there is no correct game certificate in your toolkit, please contact customer service.
4. This tool provides two modes: custom mode / AppBundle mode
5. Before calling the tool on the MAC or Linux systems, you should execute the following operations in the directory where the tool is located firstly.

```
cd where_your_mtpclientconsole_dir
chmod -R 777 ./*
```

Multi-channel attention

1. if you need to shell several game packages from different application markets, please script by refering to the example code on the buttom of article to scripting.
2. if you need to make multi-channel package on the shelled game, please use the CUSTOM MODE and cancel the verification of files which modified by channel in the configuration file during shelling.

CUSTOM MODE

Custom mode supports more comprehensive features. By calling a custom configuration file (config.json), you can freely choose to encrypt the specified .so files and .dll files, and you can freely choose whether to check all the files or partially. In addition, if the signature information is filled in the configuration file, the game will be automatically signed after processing.if you need to check some files, please read the document for details.

HOW TO USE：

```
cd where_your_mtpclientconsole_dir
MTPClientConsole.exe -d gameId apkpath outDir certPath -c configPath
```

PARAMETER DESCRIPTION：

Parameter	Meaning
-d	Standard Edition (default)
gameId	GameId (get from MTP)
apkPath	Full path of the apk file to be processed
outDir	File download location after processing
CertPath	Licence certificate issued by MTP (.cert format)
-c	Call configuration file (default, no modify)
configPath	The path of configuration file (config.json), Please put pairs of tags on the same line

APPBUNDLE MODE

Currently supported in beta

Android AppBundle file produced by packaging with Google's latest apk dynamic packaging and dynamic componentization technology. By filling in and calling the configuration file, you can freely choose to encrypt the specified .so files and .dll files. The shelling mode of AppBundle client does not support some kind of verification modes so these files in the AppBundle will be verified by default. (The current maximum size of abb file is 500M and the appbundle mode is only available on 64-bit machines.)

HOW TO USE：

```
cd where_your_mtpclientconsole_dir
MTPClientConsole.exe -a gameId aabPath outDir certPath -c configPath
```

PARAMETER DESCRIPTION：

Parameter	Meaning
-a	AppBundle Edition
gameId	GameId (get from MTP)
abbPath	Full path of the abb file to be processed (maximum file size is 500M)
outDir	File download location after processing
CertPath	Licence certificate issued by MTP (.cert format)
-c	Call configuration file (default, no modify)

Parameter	Meaning
configPath	The path of configuration file (config.json), Please put pairs of tags on the same line

FORMAT OF config.json：

```
<?xml version='1.0' encoding='utf-8'?>
<Config>
    <!--Server Tools Version, please use default value-->
    <TPVersion>default value</TPVersion>
    <!--ToolPath: mtp tools location,
     default MTPClientConsole/tools or cut this attribute-->
    <ToolPath>your_tool_absolute_path</ToolPath>
    <!--EncSo: so fileName that you want to encrypt,
     less than 5 files for performance-->
    <EncSo>libmono.so</EncSo>
    <EncSo>libGameCore.so</EncSo>
    <!--EncDll: dll fileName that you want to encrypt,
     less that 5 files for performance-->
    <EncDll>Assembly-CSharp.dll</EncDll>
    <!--Sign: information of keystore when using local sign-->
    <Sign>
        <sign-argument name='keystorePath' value='your_keystore_absolute_path'/>
        <sign-argument name='keypass' value='123456'/>
        <sign-argument name='storepass' value='123456'/>
        <sign-argument name='alianame' value='test'/>
        <!--Application Signing, Please read the Google official documentation-->
        <sign-argument name = "v1SigningEnabled" value="true"/>
        <sign-argument name = "v2SigningEnabled" value="false"/>
        <sign-argument name="v3SigningEnabled" value="false"/>
    </Sign>

    <!--Encryption configuration for globalmetadata-->
    <extparams>{"enable_globalmetadata_enc":true}</extparams>

    <!--FileCheck: verify all file in the installation package-->
    <!--selector: Optional file verification -->
    <FileCheck name="root" action="+">
        <!-- The following is an example configuration, please edit as needed -->
        <!-- Note：The content of the name key-value pair can't start or end with "/" --
        <!-- <selector name="assets/bin" action="-">
             <selector name="assets/bin/Data/Managed" action="+">
                 <selector name="assets/bin/Data/Managed/UnityEngine.dll" action="-"/>
                 <selector name="assets/bin/Data/Managed/System.dll" action="-"/>
             </selector>
             <selector name="assets/bin/Data/Resources" action="+"/>
         </selector>
         <selector name="res/layout" action="-">
             <selector name="res/layout/check_permission_layout.xml" action="+"/>
         </selector> -->
    </FileCheck>
</Config>
```

DESCRIPTION OF config.json：

Configuration item	Meaning
TPVersion	Required field, specify shell version number

Configuration item	Meaning
ToolPath	The tool will depend on the tools directory information in the toolkit.Please fill in the complete absolute path of the tools directory.(Don't add quotes.)
EncSo	It's the list of .so files you want to encript. Please only select the core files, otherwise game performance will be affected.(Don't add quotes.)
EncDll	It's the list of .dll files you want to encript. Please only select the core files such as Assembly-CSharp.(Don't add quotes.)
keystorePath	the path of the key file, string type (Please fill in the absolute path.)
keyPass	the password of keystore, string type
storePass	the password of repository, string type
aliaName	another name, string type
v1SigningEnabled	V1 signature, string type, Value is 'true' or 'false'
v2SigningEnabled	V2 signature, string type, Value is 'true' or 'false'
v3SigningEnabled	V3 signature, string type, Value is 'true' or 'false'
extparams	Enable global-metada encryption function
action	Part of the file check, "+" is checked, "-" is unchecked
selector name	The path of the file which is needed to check. The path can't start or end with "/".You should operate according to the xml hierarchy.

Pay attention:

***About TPVersion：** it is recommended to use the default version from the command-line tool. if you need to change the version, please contact us.

***About signature：** if you use local signature, you needn't to upload the certificate and add the certificate informaton. If the list of keystorePath, storePass, keyPass, aliaName, v1SigningEnabled or v2SigningEnabled are not filled or filled in incorrectly, the default is not signed. You need to manually sign after processing. Whether to use v1, v2, v3 signature is up to the developer. Please read the Google official documentation before using it. In general, aab only use v1 signatures.

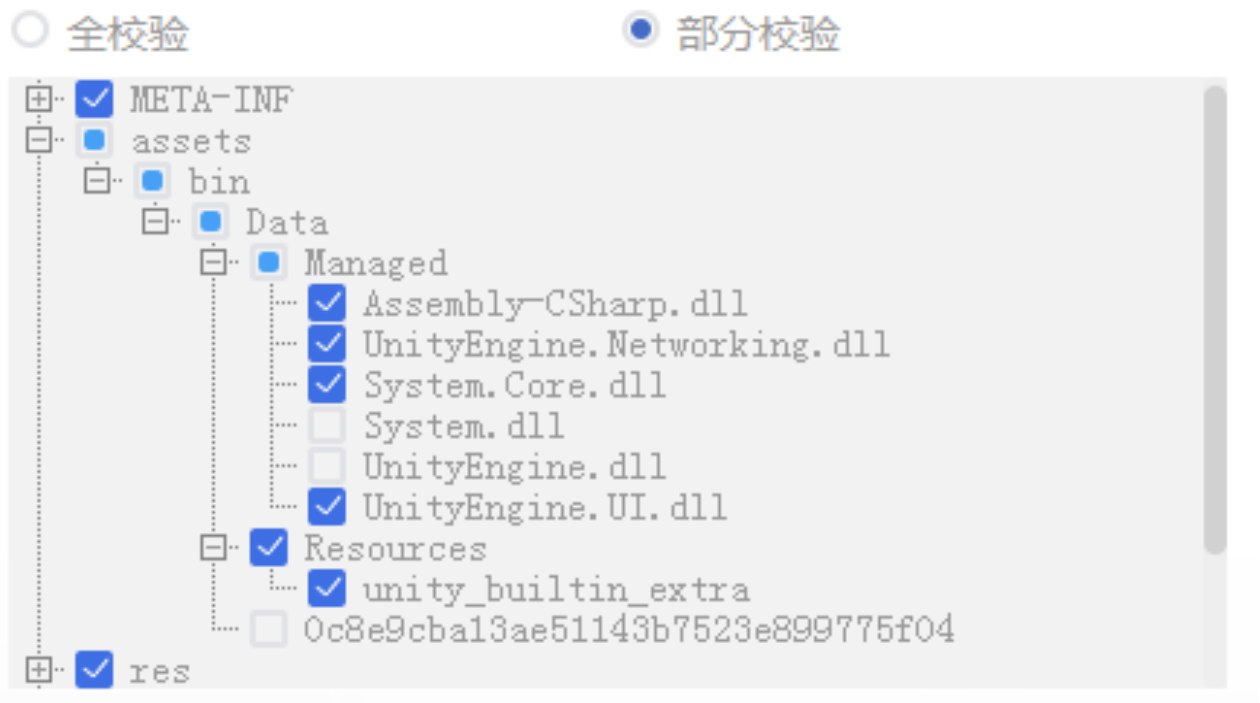
***About the list of encrypted files：** if list of EncSo and EncDll are empty or not filled, it will use the background default configuration to encrypt the files such as libmono.so, libil2cpp.so, Assembly-CSharp.dll and Assembly-CSharp-firstpass.dll.

***About the global-metadata encryption** The extparams field is the configuration used to encrypt global-metadata, If you don't need this function, you can directly delete the extparams configuration. the global-metadata encryption function is enabled by default in the latest shell version.

***About the resource file list：** if "FileCheck" is empty or not filled, The file is not checked by default when packing.

***About speed upload：** this version of the tool has integrated the Extreme Upload feature to streamline network transfer and increase transfer speed, so you needn't to upload the complete apk file.

***About selecting files for verification：** in FileCheck name, the root node is the default action ("+" is checked, "-" is unchecked). When you need to perform the opposite action from the node, you can add it into the selector attribute.For example, in the configuration file, the assets/bin directory is not selected. When the assets/bin/Data/Managed directory is selected, all the options in it are selected, then deselect the UnityEngine.dll file and System.dll file in it. Select all child files of the assets/bin/Data/Resourses directory. Then deselect all child files of the res/layout directory, only select the check_ permission _layout.xml in it.(It can be equivalent to the following operations in web shelling.)



COMMON RETURN RESULTS

Parameter	Meaning
The shelling task was finished	Success
Shell failed for time out	Fail
Configuration error	Fail
Command line parameter error	Fail
Shelling error	Fail
Local client error	Fail

Parameter	Meaning
Server error	Fail

RISK STATEMENT

The MTP Shell doesn't protect the unchecked files, and if the files is modified by players, MTP shell will not deal with it. Related information:

1. When shelling, if binary file is not checked, the player will modify and replace the binary files and generates a cracked installation package.
2. When shelling, if class.dex file is not checked. the player will delete the java code and causes some features to be unavailable.
3. When shelling, if class.dex file is not checked. the player will modify dex file to load the so file, so as to achieve the purpose of modifying the game logic and generating a cracked installation package.

Developer Suggestions

The following suggestions are provided in combination with current shelling issues and Google Developer Specifications:

1. Try to ensure that the so files are consistent under each architecture. If a file does not exist under a certain architecture, choose to delete or complete it.
2. Try to ensure that the number of dex methods is far below the 65535 limit of the Google specification. If the number is on the edge, MultiDex subcontracting is selected.
3. Try to ensure that the Shelling solutions are not mixed. If there are conflicts, shield the solutions for troubleshooting.
4. Try not to upgrade minSdkVersion and targetSdkVersion arbitrarily. During the upgrade process, you need to refer to relevant guidelines in the official Google documentation.
5. Try not to replace the signing certificate arbitrarily, there is a problem that cannot be installed due to overwriting.
6. Try not to upload a debug version of so to shell or publish. There is a risk of code leakage for files with symbols.
7. Try not to select too many so or dll when shelling, so as to avoid the performance impact.
8. Try to ensure that the declared application is placed under the main dex. if multidex subcontracting is selected, keep it in the main dex.
9. The so file protection function requires the shelled so file to contain the .init_array section, if not, contact MTP related personnel.
10. If the UE4 game uses the lld linker to fail during the shelling.See lld linker

lld linker solution

If the UE4 game uses the lld linker to fail during the shelling, Please contact MTP related personnel. At the same time, the developers need to modify GameActivity.java.template, Put System.loadLibrary("UE4") in the last line of the static block, As follows:

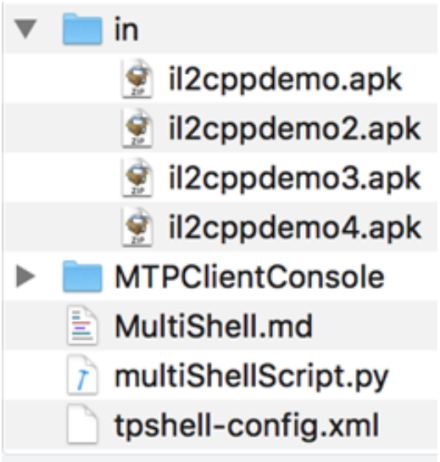
```
static{
        //$${soLoadLibrary}$$
        System.loadLibrary("c++_shared");
        System.loadLibrary("UE4");
}
```

EXAMPLE CODE

Pay attention

Shell involves many io operations, considering the machine cpu and network bandwidth, it is recommended that the number of threads opened at the same time is no more than 10, and specific numbers can be adjusted depends on machine performance, a large number of installation packages can be handled in batches by thread pool.

Directory Structure：




```python
#/usr/local/bin/python3
import os
import threading
import time
import shutil

from contextlib import contextmanager

g_temp_out = os.path.realpath("./temp")
g_threads = []

@contextmanager
def _cwd(path):
    #change workspace into multichannel
    saved_dir = os.getcwd()
    if path is not None:
        os.chdir(path)
    try:
        yield
    finally:
        os.chdir(saved_dir)

def get_timestamp():
    now = int(round(time.time()*1000))
    now = time.strftime('%Y%m%d%H%M%S',time.localtime(now/1000))
    return now

def shell_sub_main(apk_absolute_path):
    channal_name = get_timestamp() + "_" +
                   os.path.basename(apk_absolute_path).replace(".apk", "")
    workspace = os.path.join(g_temp_out, channal_name)
    if os.path.isdir(workspace):
        shutil.rmtree(workspace)
    os.makedirs(workspace)
    with _cwd(workspace):
```

```
        '''
         here run console command
         note:
         use Absolute Path in config!!!
         use Absolute Path in config!!!
         use Absolute Path in config!!!
        '''
        cmd = "mtpclient_absolute_path -d gameid " +
             "apk_absolute_path ./ cert_absolute_path -c config"
        os.system(cmd)

def shell_main():
    inDir = os.path.realpath("./in")
    files = os.listdir(inDir)
    for fi in files:
        path = "%s/%s"%(inDir, fi)
        if fi.endswith(".apk"):
            t = threading.Thread(target=shell_sub_main,args=(path,))
            g_threads.append(t)
    for t in g_threads:
        #sleep 2s here, very important
        time.sleep(2)
        t.setDaemon(True)
        t.start()
    t.join()
    print("all over")

if __name__ == '__main__':
    shell_main()
```