Cyber Security Precautions for HiMPP SDK Secondary Development

Issue 00B01

Date 2019-01-15

Trademarks and Permissions

Hi, HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road, Bantian,
Longgang District,
Shenzhen 518129 P. R. China

Website: http://www.hisilicon.com/en/

Email: support@hisilicon.com

About This Document

Purpose

This document describes the possible SDK-related cyber security risks posed to the products that are developed based on the delivery package of HiMPP series chip solutions and provides measures to address these risks.

The Hi3516C V500 delivery package is used as an example to describe the chips and codes in this document. Unless otherwise specified, the description is applicable to both chips listed in "Related Versions".

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3516C	V500
Hi3516D	V300

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 00B01 (2019-01-15)

This issue is the first draft release.

Contents

1 Introduction

The HiMPP delivery package is a chip solution delivery package, containing chip documents, hardware and software documents, SDK software packages, and PC tools. With such package, customers can develop customized products.

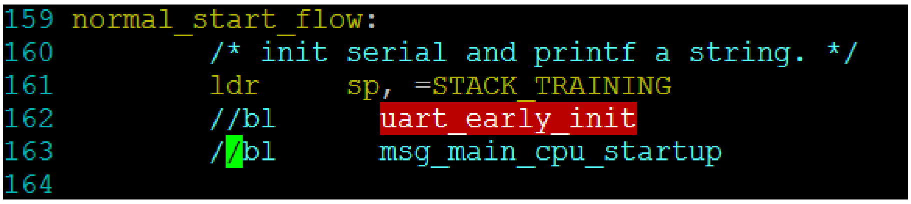
2 Cyber Security Precautions for HiMPP SDK Secondary Development

2.1 Precautions for U-Boot Usage

2.1.1 Serial Port

The serial port is a commonly used port for device communication. Generally, the port is an RS232-based serial port and used for the bottom near-end debugging of the device. In the Hi3516C V500 SDK, the serial port function of the U-Boot is enabled by default. Developers can interrupt the U-Boot execution process during the U-Boot startup phase using methods such as by pressing keys to enter the U-Boot command line interface (CLI) and perform debugging. After a product is officially released, perform the following operations to impose limitations on the serial port:

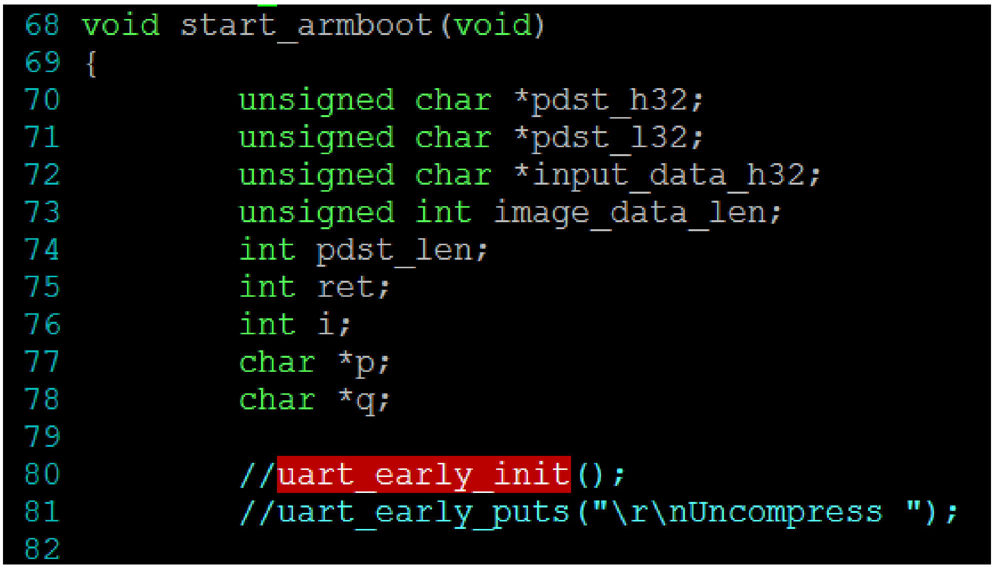
1. Remove the serial port physically during the formal product production.
2. Delete the codes related to UART initialization in the U-Boot to disable the serial port. For details, see the following steps (Hi3516C V500 used as an example):
   - Modify the **start.S** file in **u-boot-2016.11/arch/arm/cpu/armv7/hi3516cv500/** and delete the statements used to call uart_early_init.



```
normal_start_flow:
        /* init serial and printf a string. */
        ldr     sp, =STACK_TRAINING
        //bl      uart_early_init
        //bl      msg_main_cpu_startup

```

   - Modify the **startup.c** file in **u-boot-2016.11/arch/arm/cpu/armv7/hi3516cv500/hw_compressed/** by deleting the call statements related to uart_early_init and codes related to uart_early_puts.

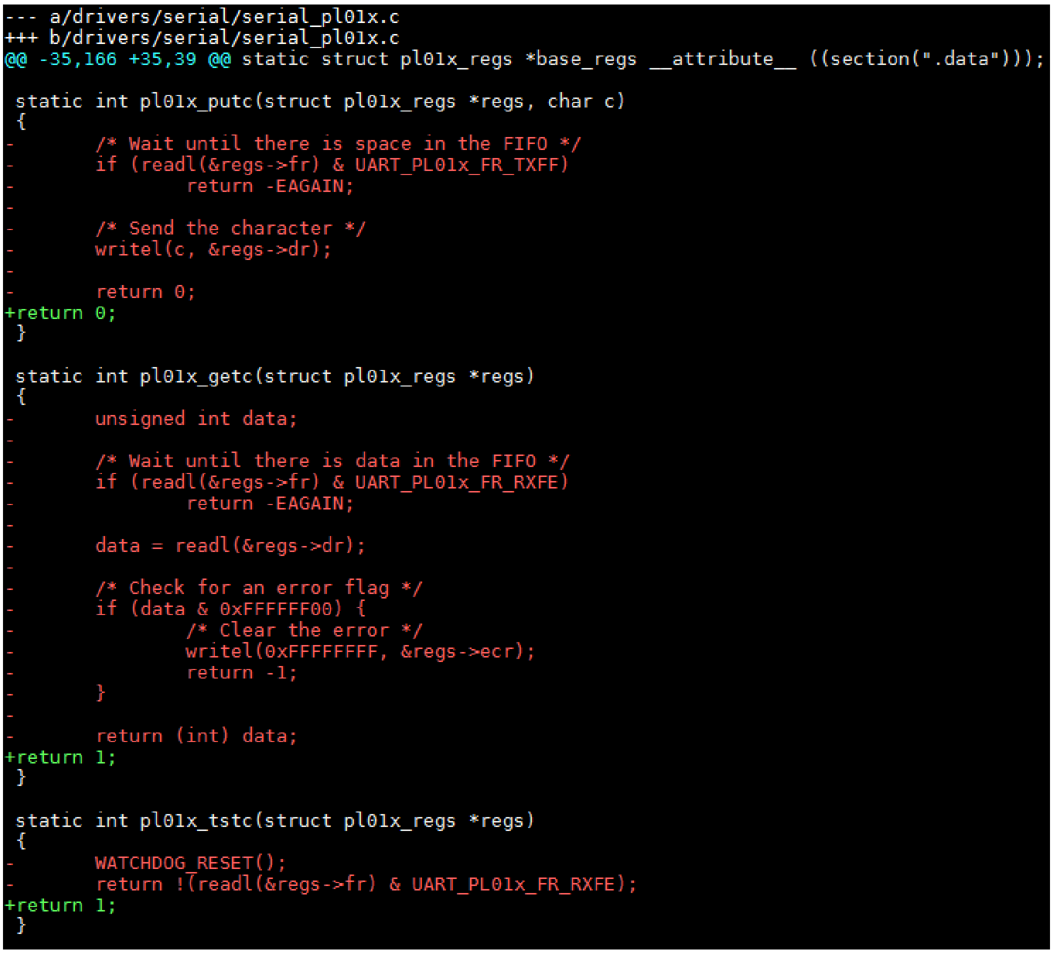


```
void start_armboot(void)
{
        unsigned char *pdst_h32;
        unsigned char *pdst_l32;
        unsigned char *input_data_h32;
        unsigned int image_data_len;
        int pdst_len;
        int ret;
        int i;
        char *p;
        char *q;

        //uart_early_init();
        //uart_early_puts("\r\nUncompress ");

```

- Modify the implementation code of pl01x_putc pl01x_getc pl01x_tstc pl01x_serial_putc pl01x_serial_tstc pl01x_serial_getc pl01x_serial_setbrg pl01x_serial_initialize pl01x_serial_probe pl01x_serial_setbrg pl01x_serial_putc pl01x_serial_pending in the **serial_pl01x.c** file in **u-boot-2016.11/drivers/serial/** and set the preceding functions to the null function.



```
--- a/drivers/serial/serial_pl01x.c
+++ b/drivers/serial/serial_pl01x.c
@@ -35,166 +35,39 @@ static struct pl01x_regs *base_regs __attribute__ ((section(".data")));

 static int pl01x_putc(struct pl01x_regs *regs, char c)
 {
-       /* Wait until there is space in the FIFO */
-       if (readl(&regs->fr) & UART_PL01x_FR_TXFF)
-               return -EAGAIN;
-
-       /* Send the character */
-       writel(c, &regs->dr);
-
-       return 0;
+return 0;
 }

 static int pl01x_getc(struct pl01x_regs *regs)
 {
-       unsigned int data;
-
-       /* Wait until there is data in the FIFO */
-       if (readl(&regs->fr) & UART_PL01x_FR_RXFE)
-               return -EAGAIN;
-
-       data = readl(&regs->dr);
-
-       /* Check for an error flag */
-       if (data & 0xFFFFFF00) {
-               /* Clear the error */
-               writel(0xFFFFFFFF, &regs->ecr);
-               return -1;
-       }
-
-       return (int) data;
+return 1;
 }

 static int pl01x_tstc(struct pl01x_regs *regs)
 {
-       WATCHDOG_RESET();
-       return !(readl(&regs->fr) & UART_PL01x_FR_RXFE);
+return 1;
 }
```

- Modify the **board_r.c** file in **./u-boot-2016.11/common/** to implement Linux image loading in run_main_loop.

2.1.2 Command Line

U-Boot command lines contain many commands that are reserved only for development and debugging by developers and can be removed from formal products.

You are advised to keep the necessary commands such as **bootm**, **go**, **sf (nand/mmc/ufs)**, **setenv**, and **saveenv** that are required by products.

U-Boot codes can be modified as follows to remove the commands that are used for development and debugging:

In the U-Boot source code, find the desired source code file by tool name. In the **Makefile** file, delete the statements used to compile the corresponding tool source code file. The following uses the mw tool as an example for description.

1. According to the description information "mw-memory write (fill)" of the mw tool, search for the source code file of the tool.

   Based on the displayed information, it is confirmed that the source file of the mw tool is **cmd/mem.c**.
2. In the **common/Makefile** file, comment out COBJS-$(CONFIG_CMD_MEMORY) += cmd_mem.o or delete the **CONFIG_CMD_MEMORY** macro definition, namely, do not compile the mw tool.

For details about deleting other commands, the operations are similar to the preceding steps.

2.2 Precautions for Linux and BusyBox Usages

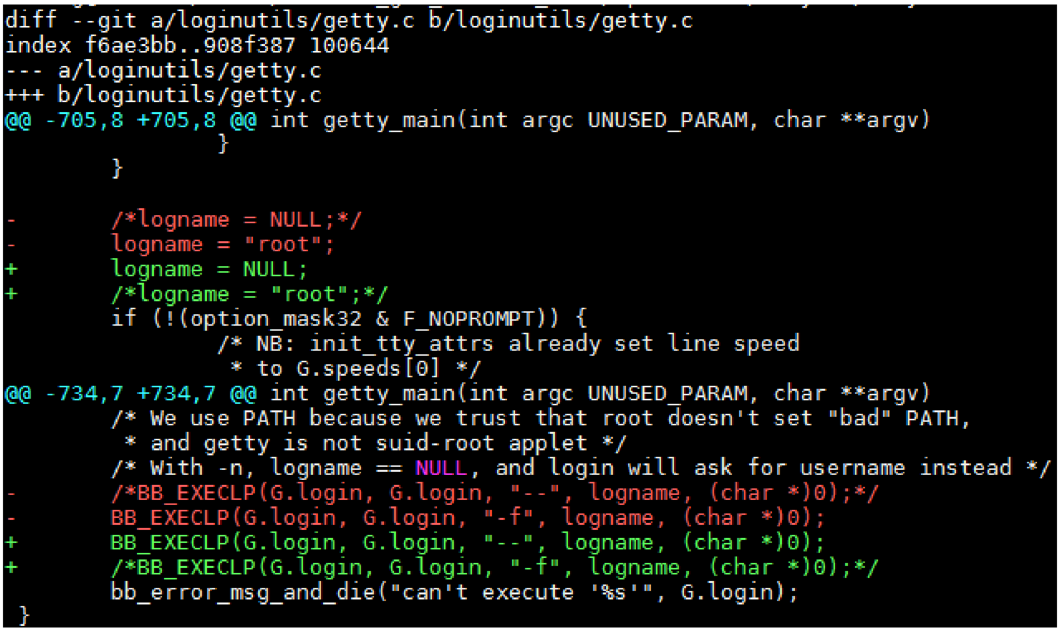
2.2.1 Root Account

The BusyBox in the Hi3516C V500 SDK only has the root user account by default, and the password is not set. The root account is reserved only for development and debugging by developers. You need to perform security hardening for the root account in actual products. The following measures are for reference:

Setting the Password for the Root Account

Step 1 Setting the function of requiring the user account and password for login

Modify the **./loginutils/getty.c** file of the BusyBox, and set the function of requiring the user account and password for login. The specific method is as follows.



```
diff --git a/loginutils/getty.c b/loginutils/getty.c
index f6ae3bb..908f387 100644
--- a/loginutils/getty.c
+++ b/loginutils/getty.c
@@ -705,8 +705,8 @@ int getty_main(int argc UNUSED_PARAM, char **argv)
                        }
                }

-               /*logname = NULL;*/
-               logname = "root";
+               logname = NULL;
+               /*logname = "root";*/
                if (!(option_mask32 & F_NOPROMPT)) {
                        /* NB: init_tty_attrs already set line speed
                         * to G.speeds[0] */
@@ -734,7 +734,7 @@ int getty_main(int argc UNUSED_PARAM, char **argv)
                /* We use PATH because we trust that root doesn't set "bad" PATH,
                 * and getty is not suid-root applet */
                /* With -n, logname == NULL, and login will ask for username instead */
-               /*BB_EXECLP(G.login, G.login, "--", logname, (char *)0);*/
-               BB_EXECLP(G.login, G.login, "-f", logname, (char *)0);
+               BB_EXECLP(G.login, G.login, "--", logname, (char *)0);
+               /*BB_EXECLP(G.login, G.login, "-f", logname, (char *)0);*/
                bb_error_msg_and_die("can't execute '%s'", G.login);
 }
```

Step 2 Modify the **/etc/passwd** and **/etc/shadow** files.

The password of the root account is null by default, which can be changed as follows:

1. Change the **/etc/passwd** file content to root:x:0:0:root:/root:/bin/sh.
2. Add the **/etc/shadow** file with the content set to root:yf/wq7vpRPGxE:0:0:99999:7:::

After the setting is complete, you need to enter the password to log in when restarting the board. Once the system is started, you can run the **passwd** command to change the default password. The new password is encrypted and saved in **/etc/passwd** and /**etc/shadow** files.

Step 3 Disabling login using shell

After the system is started, to disable login using shell, perform the following operations:

1. Modify the **/etc/passwd** configuration file.

   Change "root:x:0:0:root:/root:/bin/sh" in the **/etc/passwd** file to "root:x:0:0:root:/root:/bin/false".
2. Change "root:yf/wq7vpRPGxE:0:0:99999:7:::" in the **/etc/shadow** configuration file to "root:!:10000:0:99999:7:::".
3. Modify the **/etc/inittab** configuration file.

   In the **/etc/inittab** file, delete statements such as the following:

```
::respawn:/sbin/getty -L ttyS000 115200 vt100 -I "Auto login as I..."
```

After the proceeding sub-steps are performed, login to the board using shell is disabled.

----End

2.2.2 Execution Permission

You are advised to run your applications in non-root mode.

Step 1 Add the support of extended features to the file system.

After running the **make ARCH=arm CROSS_COMPILE=arm-himix200-linux- menuconfig** command, perform the following configurations on the configuration page:

- For JFFS2:

```
File systems  --->
<*>Journalling Flash File System v2 (JFFS2) support
[*]JFFS2 XATTR support
```

- For YAFFS2:

```
File systems  --->
<*>yaffs2 file system support
[*]Enable yaffs2 xattr support
```

- For RAMFS:

```
File systems  --->
Pseudo filesystems  --->
[*]Tmpfs extended attributes
```

Step 2 Add the filecap tool to support privilege operations of common users.

1. Source code package

   Use the **libcap-ng-x.x.x.tar.gz** source code package. You are advised to download the latest version from the Internet.

2. Method of cross compilation

```
$ ./configure  --prefix=out --host=arm-himix200-linux
$ make && make install
```

3. After the compilation is complete, copy the executable files in the **out/bin/** directory to the **rootfs/bin** directory and copy the library files in the **out/lib/** directory to the **rootfs/lib** directory. The following takes access to the **/dev/mem** device file as a non-root user as an example to describe how to add privilege attributes to the non-root user:

   – Log in to the system as the **root** user.

   – Add a new user **test**.

```
~ # adduser test
Changing password for test
New password: ******
Retype password: ******
Password for test changed by root
```

   – Check the permissions of the **/dev/mem** device file.

```
~ # ls /dev/mem -l
crw-------    1 root     root       1,   1 Jan  1 00:00 /dev/mem
~ #
```

   The statements indicate that only the root user has the read and write permissions.

   – Execute the test program btools as the test user.

```
~ # su test
sh: using fallback suid method
~ $ btools 0x82000000
*** Board tools : ver0.0.1_20121120 ***
```

```
====dump memory 0x82000000====
[error]: memmap():open /dev/mem error!
[error]: Memory Map error. exit:0XFFFFFFFF.{source/tools/himd.c:99}
[END]
```

- Run the following command to switch to the root user and use the filecap tool to add the sys_rawio and sys_admin permissions to the **/bin/btools** directory.

```
~ $ su root
# filecap /bin/btools sys_rawio sys_admin
```

Run the following command again:

```
# filecap /bin/btools
file                 capabilities
/bin/btools     sys_rawio sys_admin
In this case, "/bin/btools     sys_rawio sys_admin" is displayed,
indicating that the sys_rawio and sys_admin permissions have been
added to the btools program.
```

- Execute the test program btools again as the test user.

```
# su test
sh: using fallback suid method
~ $ btools 0x82000000
*** Board tools : ver0.0.1_20121120 ***
====dump memory 0x82000000====
0000:  00000000 2f406567 69766564 2f736563
0010:  2f636f73 3a636f73 61626d61 3132312f
0020:  30303534 69702e30 00736972 49544341
0030:  633d4e4f 676e6168 45440065 54415056
0040:  642f3d48 63697665 732f7365 732f636f
0050:  613a636f 2f61626d 34313231 30303035
0060:  7269702e 53007369 59534255 4d455453
0070:  616c703d 726f6674 464f006d 4d414e5f
0080:  69703d45 00736972 465f464f 4e4c4c55
0090:  3d454d41 636f732f 626d612f 69702f61
00a0:  40736972 34313231 30303035 5f464f00
00b0:  504d4f43 42495441 305f454c 7369683d
00c0:  63696c69 702c6e6f 73697269 5f464f00
00d0:  504d4f43 42495441 4e5f454c 4d00313d
00e0:  4c41444f 3d534149 4e3a666f 69726970
00f0:  4e3c5473 3e4c4c55 73696843 63696c69
[END]
```

The preceding information is displayed, indicating that the test user can use the btools to properly open the **/dev/mem** device.

----End

2.2.3 Telnet Service

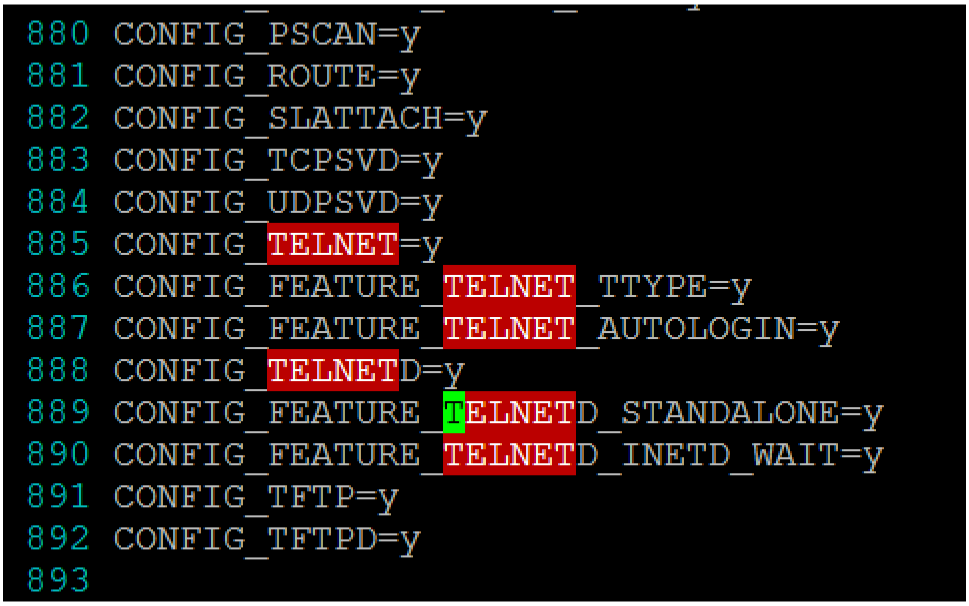
Telnet is an insecure transmission protocol. The password is transmitted in plain text, easily leading to network interception and data stealing.

The Telnet service of the BusyBox can only be used for development and debugging and cannot be used in formal products.

Telnet must be disabled. The following uses Hi3516C V500 as an example to describe how to disable Telnet:

Step 1 Open the **config_v200_a7_softfp_neon** file.

Step 2 Search for the Telnet-related configuration options.



```
CONFIG_PSCAN=y
CONFIG_ROUTE=y
CONFIG_SLATTACH=y
CONFIG_TCPSVD=y
CONFIG_UDPSVD=y
CONFIG_TELNET=y
CONFIG_FEATURE_TELNET_TTYPE=y
CONFIG_FEATURE_TELNET_AUTOLOGIN=y
CONFIG_TELNETD=y
CONFIG_FEATURE_TELNETD_STANDALONE=y
CONFIG_FEATURE_TELNETD_INETD_WAIT=y
CONFIG_TFTP=y
CONFIG_TFTPD=y
```

Step 3 Comment out the Telnet-related options.

Step 4 Save the file and then recompress the **busybox-1.26.2** folder as **busybox-1.26.2.tgz**.

----End

2.2.4 TFTP Service

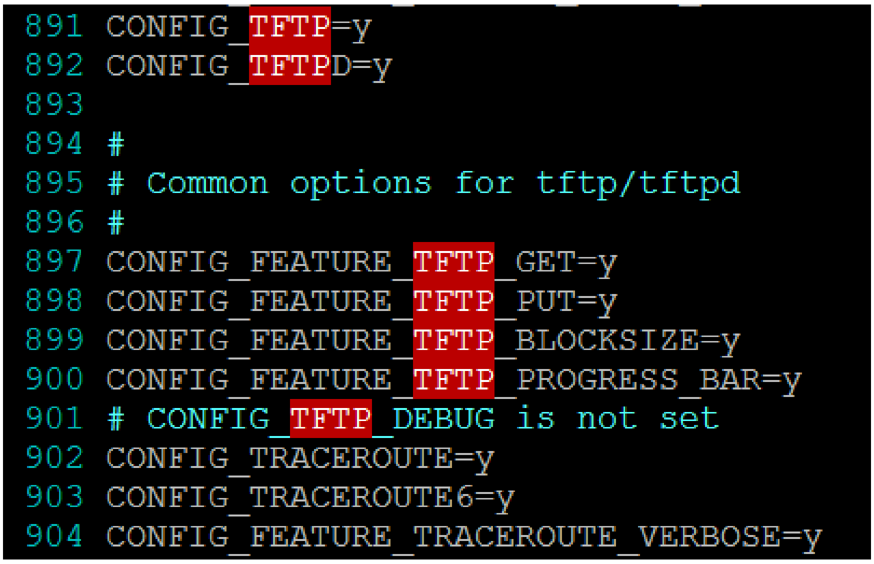
TFTP is an insecure transmission protocol. The password is transmitted in plain text, easily leading to network interception and data stealing.

The TFTP service of the BusyBox can be used only for development and debugging and cannot be used in formal products.

Take Hi3516C V500 as an example. The TFTP service can be disabled as follows:

Step 1 Open the **config_v200_a7_softfp_neon** file.

Step 2 Search for the TFTP-related configuration options.



```
CONFIG_TFTP=y
CONFIG_TFTPD=y

#
# Common options for tftp/tftpd
#
CONFIG_FEATURE_TFTP_GET=y
CONFIG_FEATURE_TFTP_PUT=y
CONFIG_FEATURE_TFTP_BLOCKSIZE=y
CONFIG_FEATURE_TFTP_PROGRESS_BAR=y
# CONFIG_TFTP_DEBUG is not set
CONFIG_TRACEROUTE=y
CONFIG_TRACEROUTE6=y
CONFIG_FEATURE_TRACEROUTE_VERBOSE=y
```

Step 3 Comment out the TFTP-related options.

Step 4 Save the file and then recompress the **busybox-1.26.2** folder as **busybox-1.26.2.tgz**.

----End

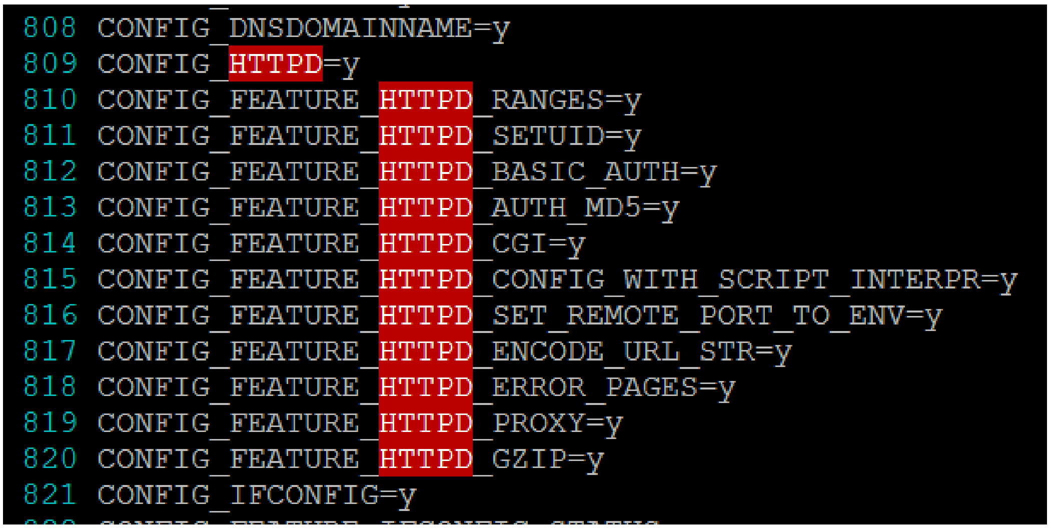
2.2.5 HTTPd Service

To support the VOD function, the HTTPd service is enabled for the BusyBox by default. You can perform the following steps to disable the HTTPd service:

Take Hi3516C V500 as an example:

Step 1 Open the **config_v200_a7_softfp_neon** file.

Step 2 Find the HTTPd configuration options.



```
CONFIG_DNSDOMAINNAME=y
CONFIG_HTTPD=y
CONFIG_FEATURE_HTTPD_RANGES=y
CONFIG_FEATURE_HTTPD_SETUID=y
CONFIG_FEATURE_HTTPD_BASIC_AUTH=y
CONFIG_FEATURE_HTTPD_AUTH_MD5=y
CONFIG_FEATURE_HTTPD_CGI=y
CONFIG_FEATURE_HTTPD_CONFIG_WITH_SCRIPT_INTERPR=y
CONFIG_FEATURE_HTTPD_SET_REMOTE_PORT_TO_ENV=y
CONFIG_FEATURE_HTTPD_ENCODE_URL_STR=y
CONFIG_FEATURE_HTTPD_ERROR_PAGES=y
CONFIG_FEATURE_HTTPD_PROXY=y
CONFIG_FEATURE_HTTPD_GZIP=y
CONFIG_IFCONFIG=y
```

Step 3 Comment out the HTTPd options.

Step 4 Save the file and then recompress the **busybox-1.26.2** folder as **busybox-1.26.2.tgz**.

---End

2.2.6 File Permission

You are advised to control the file and device permissions in the **rootfs** directory based on the principle of minimum permissions.

2.2.7 Debugging Tool

The btools in the **/bin** directory of rootfs is a tool used for debugging read and write registers, read and write commands of I^{2}C and SPI interface components. Commands such as **himc**, **himd**, **himd.l**, **himm**, **hiddrs**, **i2c_read**, **i2c_write**, **ssp_read**, and **ssp_write** are soft links of the btools. You are advised to delete the btools and the soft links from formal products.

All scripts in the **rootfs** directory are provided as samples for reference, debugging or demonstration only.

The OSDRV directory in the SDK provides some tools for board debugging, which are stored in the **osdrv/tools/board** directory. The e2fsprogs, gdb, mtd-utils, and reg-tools-1.0.0 are used only for development and debugging. They cannot be used in any actual products.

2.3 Precautions for Linux Driver Usage

2.3.1 Serial Port

The serial port is a commonly used port for device communication. Generally, the port is an RS232-based serial port and used for the bottom near-end debugging of the device.

The following measures must be taken to avoid risks in actual products:

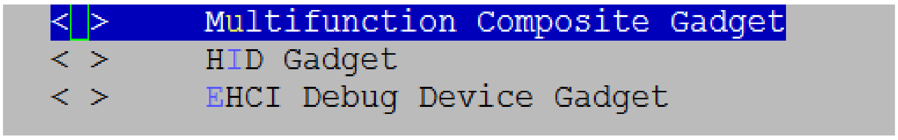
Disable the serial port: If the serial port is not used on the live network, it can be disabled at delivery. In this way, the illegal access of the serial port is prevented during device running.

2.3.2 USB

The Hi3516C V500 SDK Linux OS supports the USB interface as the debugging interface. The debugging function of the USB interface must be disabled in the actual products as follows:

On the **make menuconfig** interface of the Linux OS, do not select **Multifunction Composite Gadget**. The details are as follows:

```
Device Drivers  --->
    [*] USB support  --->
    <*>   USB Gadget Support  --->
    < >     Multifunction Composite Gadget
```



```
< >     Multifunction Composite Gadget
< >     HID Gadget
< >     EHCI Debug Device Gadget
```

2.4 Precautions for Application Development

2.4.1 Code Security

2.4.2 Cipher Driver

The cipher driver implements the standard symmetric encryption algorithms (AES, DES, and 3DES), asymmetric encryption algorithm RSA, signature verification algorithms RSA 1024, RSA 2048, RSA 3072, and RSA4096, and digest algorithms SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, and HMAC. It does not use any private algorithms.

- The DES algorithm has low security, because the improvement of computing capabilities makes the brute force cracking possible. Therefore, the DES algorithm is not recommended.
- The security of the 3DES algorithm is low. Its security level is lower than the AES-128 algorithm even when the K1, K2, and K3 are different from each other. Therefore, the 3DES algorithm is not recommended.
- A longer cipher key indicates a higher security level. You are advised to use an AES key of 128 bits or higher, or an RSA key of 2048 bits or higher.
- The security of the SHA-1 algorithm is low, and therefore is not recommended.

2.4.3 MPI or API

- When an application needs to call the MPI/API that requires the input of a user path, the validity and correctness of the input path must be ensured. The SDK does not provide the file system layout of a product. It is recommended that an application call the realpath function to verify the path validity.
- For the Linux OS, the**/dev/mem** device file of the kernel is used to access the physical address. As a result, the functions of the driver device nodes, MPIs, and APIs of the SDK must be called by the **root** user. The **user** account can be used to execute these functions only after it is assigned with the CAP_SYS_RAWIO and CAP_SYS_ADMIN file access permissions.
- The SDK allows secondary development engineers to manage and configure the physical addresses of chips. For the interfaces involving physical addresses in the SDK, you need to ensure the correctness of physical addresses and their lengths of MPIs and APIs. Otherwise, unknown errors may occur.
- For interfaces involving operations such as memory mapping, unmapping, and flushing, ensure that the call sequence is correct. Ensure that you access the corresponding memory and perform operations such as flushing the memory before the unmap operation and after the map operation. For details, see the *HiMPP V4.0 Media Processing Software Development Reference*. Otherwise, unknown exceptions may occur. The related interfaces are as follows:

  HI_MPI_SYS_Mmap, HI_MPI_SYS_MmapCache, HI_MPI_SYS_Munmap, and HI_MPI_SYS_MflushCache
- The SDK driver can obtain streams in select mode. However, the select mode does not support multi-thread operations. Therefore, you need to ensure that multiple threads are not used to obtain streams of the same channel in select mode.

- Some MPI/API parameters of the SDK are user-mode pointers (such as HI_MPI_VENC_ReleaseStream). Ensure that the parameter values are correct. Otherwise, segment errors may occur.

2.4.4 Sample/Tools/Extdrv

In the SDK, the **sample** directory stores example codes that allow developers to quickly understand the functions of MPIs and drivers. The **tools** directory is a tool directory and can be used by developers for debugging during media development. extdrv is the driver software of the peripheral chip of the demo board. It can be used during the demonstration of the demo board.

Some driver software is provided by third parties or contains third-party codes, which are provided only as samples for customers, reference, debugging, or demonstration. For details, see the **readme** file in the **extdrv** directory.

2.5 Other Precautions

2.5.1 Image Burning

The U-Boot allows data to be burnt to the USB and SD cards, which cannot be used in the upgrade of actual products. Because the sources of images, boards, and products are different, the SDK version does not support the upgrade function, which needs to be developed by customers. Upgrade security needs to be considered during the development.

2.5.2 PC Debugging Tool

The SDK provides the following PC debugging components:

- HiTool: used for debugging the firmware of the burning-supported board.
- PQTool: used for debugging the image quality and effect.
- AVS_CALIBRATION: used for calibrating the image stitching effect.
- IVE_CLIB: used for the IVE PC simulation library.
- IVE_Tool: used for IVEANN/SVM/CNN conversion.
- DPU_Tool: used for the conversion of the DPU correction lookup table.
- NNIE_Tool: used for NNIE compilation.
- NNIE_Lib: used for the NNIE PC simulation library.

The preceding tools are not debugged online and cannot be used in any released product.

The **OSDRV** directory (**osdrv/tools/pc**) in the SDK provides some tools for debugging on a PC and formulation of file systems. The tools in this directory are used only in the development phase for customers and can only run on PCs. They cannot be used in any customer's products.

2.5.3 Image Burning Without the U-Boot

The SDK package supports image burning through the USB or SD card or serial or network port when the U-Boot is absent from the start-up flash. You are advised to disable image burning in actual products.

- The USB and SD cards can be disabled through the hardware design.

- The function of burning a die through the USB or SD card can be disabled through software.

Take Hi3516C V500 as an example. In the U-Boot code **./include/configs/hi3516cv500.h**, do not define the CONFIG_AUTO_UPDATE macro. Delete the save_bootdata_to_flash function from the **board/hisilicon/hi3516cv500/hi3516cv500.c** file. Then, the new U-Boot will not be written to the flash.

2.5.4 Mount Permission of an SD Card

When a customer develops a product with a pluggable storage medium (such as the SD card or USB flash drive), ensure that the **-o noexec** option is added to the mount command before the file system of the storage device is mounted. This prevents third-party programs from running.

2.5.5 JTAG

Attackers may modify system configurations through the JTAG interface to deliberately damage to a system.

You are advised to physically remove the JTAG interface from devices before they are delivered.

3 Outlook

As a node on the network, HiMPP faces increasingly severe security threats. From the object perspective, HiMPP security involves not only HiMPP itself but also the cloud, client, and IPC that interact with HiMPP. From the layer perspective, it involves the management, physical, system, network, and service layers. Customers need to take corresponding security measures based on the security threat analysis.

The following security principles are for reference:

- Appropriate security

  Security design is the most appropriate security measure based on the analysis of the specific security risk scenarios and the consideration of the performance, cost, and service impact.

- Minimum authorization

  Assign the minimum permission and resources to users and maintenance personnel in terms of network elements, programs, and progresses according to responsibilities. This can reduce potential security risks.

- Active cooperative defense

  Identify malicious attack sources in a timely manner, and enable automated deletion of the connections between malicious users and the network before an attack causes serious damages. Alternatively, reduce the bandwidth and service quality of connections to minimize negative impacts.

- In-depth defense

  The principle of in-depth defense involves multiple levels of defenses against threats. For example, when a defense layer is not enough, another one takes effect to prevent a complete destruction.