

Author: Thomas Perkins

Detailed analysis of the Contec CMS8000 Patient Monitor  
  
Abstract:  
The Contec CMS8000 Patient Monitor is a medical device designed to provide real-time monitoring of vital signs. This report provides a comprehensive analysis of the device, revealing critical security vulnerabilities that pose a risk to patient privacy, data integrity, and overall system security. This analysis examines the device's communication protocols, firmware structure, update mechanisms, and corporate affiliations to provide assess a full view of the security posture of Contec Medical Systems device.

# Key findings

* By default, the Contec CMS8000 Patient Monitor streams anonymized real-time HL7 data to the IP address: 202.114.4.120 on port 51. This communication is automatically established without any need for user configuration. Although the data is anonymized you can reconstruct detailed graphs of patient medical information such as heartbeat and other vital signs.
* The firmware sits on a UnilC DRAM chip specifically the SCN00SA1T1 model. The firmware is organized in Yet Another Flash File System (YAFFS) and contains a total of 211 files.
* The device uses Point-to-Point (PPP) protocol for internal network communication. This is unusual but plausible due to the type of device.
* The device contains a potential backdoor ELF binary file named “cmddog”. This file listens on a UNIX sock that is globally accessible due to the user of chmod 0777 on the sock. This allows all internal processes to communicate with the sock. The “backdoor” listens for commands in a specific format and executes them within the context of the user.
* The CMS8000 Patient Monitor has an update mechanism that retrieves updates from a mounted NFS share on IP address: 202.114.4.119. The update procedure does not attempt to validate the update in any way.
* An analysis of the corporate structure of Contec Medical Systems reveals a direct connection to the Chinese Communist Party through independent board member Mr. Yang Changdong.

# Real time data stream

By default, the Contec CMS8000 Patient Monitor device streams real time HL7 patient data to an IP address located in China. The streamed data includes everything that the device is capable of monitoring, the format of which is broken down below:

* Message header
  + Separator
  + Encoding characters
  + Sending application
  + Sending facility
  + Receiving application
  + Receiving facility
  + Datetime timestamp
  + Message type
  + Message control ID
  + Processing ID
  + Version ID
* Patient identification (PID)
  + Patient ID
  + Patient name
  + Date of birth
  + Gender
* Patient visit (PV1)
  + Patient class
  + Location
* Observation request (OBR)
  + Observation date and time
* Observation results (OBX)
  + Base64 encoded pulse waves and observation data

The data is streamed to IP address 202.114.4.120 on port 511. By default, the data stream is anonymized, the patient information consisting of random characters and/or numbers. Notably the device steams data without external attachments.

We modified the default IP address in the device and successfully intercepted the data stream by activating the monitor. The intercepted HL7 data allowed us to reconstruct visual representations from the base64 encoded pulse wave without any identifying patient information. It is important to note that this reconstruction without identifying patient information **does not** violate HIPPA regulations.

For reference the default IP address can be found in the user manual on page 28.

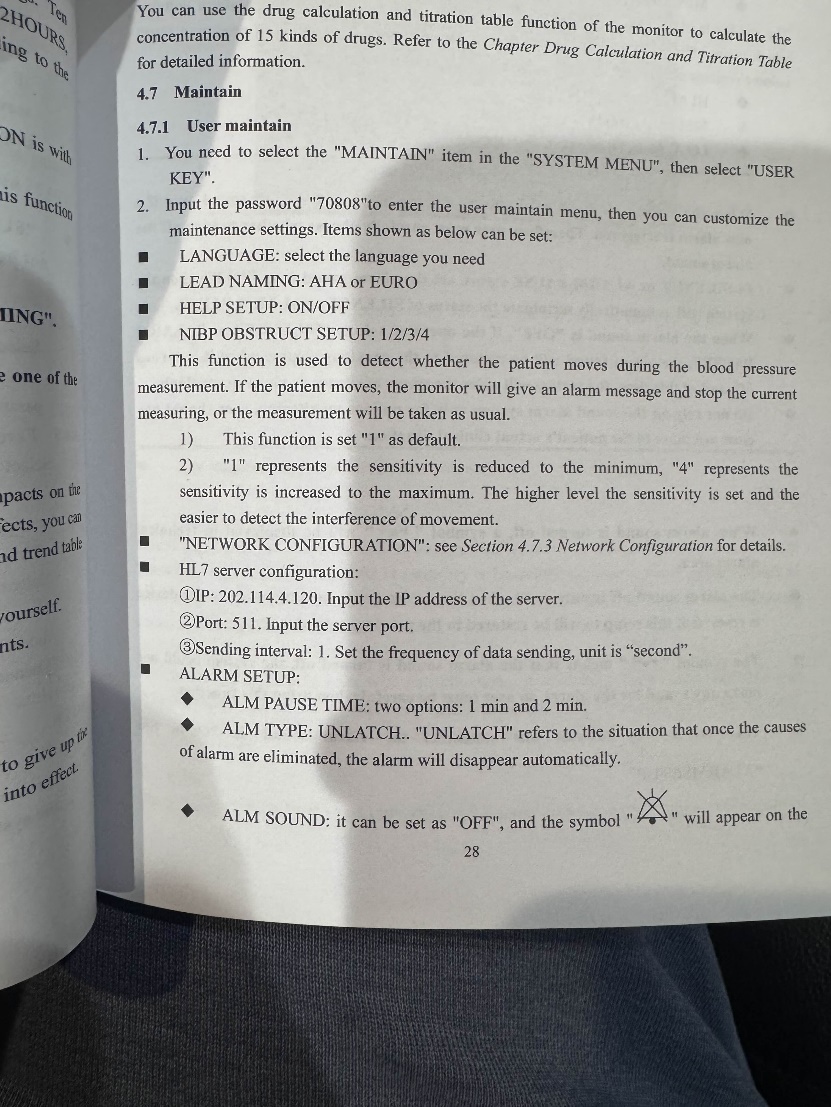


Figure . Default HL7 data IP address in manual

A black background with many small colored lines

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

Figure . Streamed HL7 data being intercepted

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure . Reconstructed base64 encoded pulse wave using ChatGPT

# Device internals

The device firmware is stored on a DRAM chip labeled SN00SA1T1. The chip manufacturer is Xi'an UnilC Semiconductors Co., Ltd. (<http://en.unisemicon.com/>). There is unfortunately no datasheet available for this specific device that we were able to find.

Using the Xeltek SuperPro 6100n, we successfully extracted the firmware from the chip. The firmware binary blob is formatted in Yet Another Flash File System (YAFFS), which is specifically designed for NAND flash memory and created in 2002 by Charles Mannning. We used the yaffs2 utility to extract the filesystem from the firmware blob.

This extraction resulted in a total of 211 files. Many of these files are random configuration and image files used by the device. The files appear to be organized in directories that correspond to their functionality, as follows:

* **Root directory:** Contains the start scripts that are executed when the monitor is powered on.
* **Wireless:** Contains one single installation script that is responsible for installing WiFi adapter drivers as well as the drivers themselves.
* **ppp:** Contains binaries and scripts used for Point-to-Point internal communication.
* **bin:** Contains all executable binaries as well as configuration files for internal and external device operations.

A screen with white text on it

AI-generated content may be incorrect.A silver rectangular electronic device with a yellow label

AI-generated content may be incorrect.

Figure . Chip set as well as SuperPro 6100n

A screen shot of a computer screen

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

Figure . Binary blob and extraction of the filesystem

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Figure . Device internal folder structure

# Device communication

The device communicates using the Point-to-Point (PPP) protocol to its external extensions. This protocol is a layer2 protocol used for direct communication over serial connections. This is an unusual approach however it makes sense given the nature of the device.

Communication is established by creating serial ports during the boot process and waiting for external connections to be connected. Once an extension is plugged in the device will begin a “chat” with the extension using the PPP protocol by executing a corresponding script. These scripts are Code Division Multiple Access (CDMA) modules. We believe that ETCOM is the manufacturer.

This communication method presents what seem to be significant security risks. There are hardcoded credentials contained in the communication scripts, allowing anyone who finds them the ability to send custom commands to the serial ports. Furthermore, the use of the PPP protocol opens the device up for man in the middle attacks and potential remote access vulnerabilities.

The presence of USBC communication within one of the scripts may reveal undocumented functionality within the device. However, it is most likely proprietary communication with the serial ports on boot. We unfortunately were unable to determine the exact reason for the USBC communication.

A screenshot of a computer program

AI-generated content may be incorrect.A computer screen shot of white text

AI-generated content may be incorrect.

Figure . Point-to-Point chat scripts

A screenshot of a computer program

AI-generated content may be incorrect.A screen shot of a computer code

AI-generated content may be incorrect.

Figure . USBC device communication

The backdoor  
  
Inside the bin folder there is an ELF binary file named “cmddog”. This file functions similarly to a command-and-control mechanism that passively sits and waits for commands sent through a UNIX sock. The exact purpose of this binary is unclear. Speculation suggests that it may have been used for remote debugging during development. However, it seems implausible (although not impossible) that an engineer would knowingly introduce a shell within a production environment.   
  
The “cmddog” binary is launched in the background during startup originating from a script called “startmonitor”. This startup script is responsible for launching the monitors scripts, updating the modules, installing the wireless adapter drivers at runtime, and starting the monitor itself. Interestingly the second to last command run starts “cmddog” in the background so that it is always listening.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure . "cmddog" being launched in the startup script

The “cmddog” binary is an ELF 32- bit executable built for ARM architecture. Upon execution it performs the following actions:

* Detects and removes any existing temporary files named “/tmp/temp.unix”.
* Sets up custom signal handling for commands ranging from 0x26 to 0x29.
* Creates a socket, if this socket fails to be created the program will exit.
* Configures the socket to use the UNIX sock file: “/tmp/temp.unix”.
* Binds the socket allowing all local processes to communicate with it.
* Sets the files permissions to octal 0777 granting all users execute access to the UNIX sock.
* Begins to listen for incoming connections.
* Accepts the connections and checks the data length, if 0 exits the program.
* Transfers control to a subroutine called “sub\_88ac”.

Interestingly the signal commands do not contain any functional code which effectively makes them no-ops or no operation functions, this suggests that the program is incomplete or may contain anti-analysis functionality.

You may have noticed that the socket binds to 1 as well, this is because the disassembler interpreted AF\_UNIX as a literal without recognizing the socket structure type as sockaddr\_un. As a result, this program is designed to persistently run in the background.

Additionally, the program has multiple unused functions that seem to serve no purpose other than to obfuscate analysis. Once we have cleaned up the code a bit it becomes obvious that the execution of commands is dependent on the following format:

* Look for the string “LEN\_” inside of the sent command.
* Check if the command length is less than 255 characters (this check appears to be unimplemented).
* Searches for the string “CMD\_” and retrieves the portion of the command after that string.
* Executes the extracted command in the context of the current user.
* Sends back the commands response if it was executed successfully or sends back the commands error if the execution failed.

For example, the following command will be executed successfully and return the contents of the file “/etc/passwd”: **LEN\_0011 CMD\_cat /etc/passwd**

Interestingly the validation of the command length does not appear to be properly implemented. This appears to be an oversite from the developer.

A computer screen shot of a program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

Figure . Socket assignment and associated functions

A screen shot of a computer program

AI-generated content may be incorrect.

Figure . The “sub\_88ac” function that the main function passes control to

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

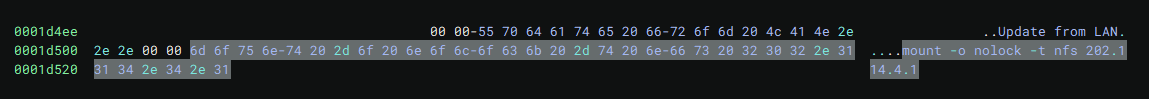
Figure . The functions responsible for analyzing the sent commands

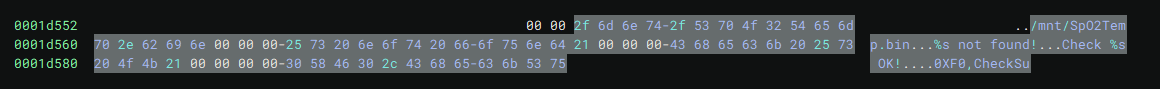
# Insecure updating mechanism

The installation script contains 3 update mechanisms called “update\_module”, “update\_sp02”, and “update”. Within “update\_spo2” there is an IP address (202.114.4.1) that is mounted as a NFS share. A binary file named “SP02Temp.bin” is pulled from this mount point, its purpose is unknown because we are unable to access the mount point at the time of writing.

This IP is linked to multiple possible locations. Most indicators point the location to a Chinese university and research facility. If this is indeed functions as the update module, it presents a vulnerable update mechanism that could allow unauthorized device updates. This opens an attack vector that allows attackers the ability to upload their own binaries to the device.

Interestingly the startup script runs the update mechanisms every time the device is turned on without any periodic updates being done. If the device remains on for long periods of time, no updates will be done on the device.





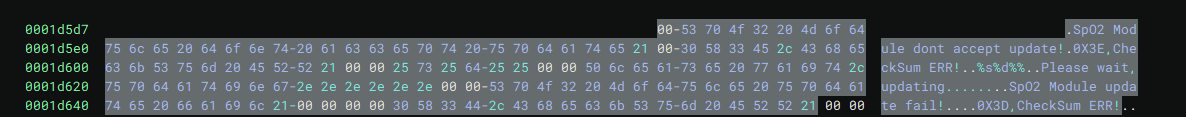


Figure . Update mechanism mounting the IP address and attempting to pull the binary file

A screenshot of a computer

AI-generated content may be incorrect.

Figure . IP address with multiple locations associated with it

A screenshot of a computer

AI-generated content may be incorrect.

Figure . IP address mount seen in another tool

# Chinese Communist Party

During our initial investigation we purchased the device from 3 different vendors on eBay. Performing this approach (we thought) would allow us to compare the devices together to determine if different vendors were selling different versions of the device.

However, upon review of the seller information of each vendor we noticed that each vendor had the same information: “Kangtaiyixuexitong Qinhuangdao Gufenyouxiangongsi”. This translates to “Kangtai Medical Systems Qinhuangdao Co, Ltd” which is directly linked to Contec Medical Systems. Our research revealed that chairman Hu Kun is the president and chairman of Kangtai Medical from since October 2010.

This discovery prompted us to start investigating the board of Contec Medical Systems to determine any interesting connections within the board. We uncovered that Mr. Yang Changdong, a Chinese nationalist with no permanent residency abroad, is a member of the Communist Party of China (CCP), serves as an independent director on the board.

An independent director is a board member who is neither a shareholder nor part of the company’s executive team. They provide unbiased oversight, strategic guidance, as well as corporate governance to steer the company’s direction.

This information establishes a direct link to Contec Medical Systems and the CCP. The extent of this connection is uncertain at the time of writing. We highly recommend the reader to independently verify these claims as well.

A close up of a number

AI-generated content may be incorrect.

Figure . Seller information

A screenshot of a computer

AI-generated content may be incorrect.

Figure . Selling information translated

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a website

AI-generated content may be incorrect.**A screenshot of a delivery

AI-generated content may be incorrect.

Figure . Seller information and pricing of all purchased devices

A screenshot of a computer

AI-generated content may be incorrect.

Figure . Contec Medical Systems board members

# Conclusion

The Contec CMS8000 Patient monitor appears to exhibit significant security vulnerabilities, and interesting data transmission behaviors. The key takeaway from this report is that the device appears to present a major security risk due to its insecure data practices: insecure data transmission, a potential backdoor, unverified update mechanism, and hard coded credentials. These vulnerabilities and practices can lead to patient data exposure, remote device control, and malicious device upgrades. Additionally, the verified connection to the Chinese Communist Party raising further concerns about data privacy and device integrity.

# Recommendations

* If this device is present in an area, you are familiar with, perform immediate network monitoring and block suspicious outgoing connections.
* Sandbox the device or modify the firmware to prevent unauthorized access.
* Investigate the regulatory compliance of the device. HIPPA and GDPR.
* Further research into the corporate affiliations should be investigated.
* Further research into other devices that are offered by Contec Medical Systems.

# Other information

You can find all the information in this report on Github here: <https://github.com/PenetrumLLC/Contec-CMS8000>. Below is a list of IP addresses discovered during analysis and the Malcore share link associated with them.

* 202.114.4.119
  + <https://app.malcore.io/share/637b72e6d2815d7e836350da/67aa2aa5da3e8886f5e41856>
* 202.114.4.115
  + <https://app.malcore.io/share/637b72e6d2815d7e836350da/67aa2b71782e1dfbf8eb6004>
* 202.114.4.120
  + <https://app.malcore.io/share/637b72e6d2815d7e836350da/67aa2f0f782e1dfbf8eb66db>
* 202.114.4.110
  + <https://app.malcore.io/share/637b72e6d2815d7e836350da/67ab7938da3e8886f5e53c1f>

All hashes of the files can be found on Github also.