ARCHIVED STORY

McAfee Enterprise ATR Uncovers Vulnerabilities in Globally Used B. Braun Infusion Pump

Douglas McKee and Philippe Laulheret · Aug 24, 2021

Overview

As part of our continued goal to provide safer products for enterprises and consumers, we at McAfee Advanced Threat Research (ATR) recently investigated the *B. Braun Infusomat Space Large Volume Pump* along with the *B. Braun SpaceStation*, which are designed for use in both adult and pediatric medical facilities. This research was done with support from Culinda – a trusted leader in the medical cyber-security space. Though this partnership, our research led us to discover five previously unreported vulnerabilities in the medical system which include:

1. CVE-2021-33886 – Use of Externally-Controlled Format String (CVSS 7.7)
2. CVE-2021-33885 – Insufficient Verification of Data Authenticity (CVSS 9.7)
3. CVE-2021-33882 – Missing Authentication for Critical Function (CVSS 8.2)
4. CVE-2021-33883 – Cleartext Transmission of Sensitive Information (CVSS 7.1)
5. CVE-2021-33884 – Unrestricted Upload of File with Dangerous Type (CVSS 5.8)

Together, these vulnerabilities could be used by a malicious actor to modify a pump's configuration while the pump is in standby mode, resulting in an unexpected dose of medication being delivered to a patient on its next use – all with zero authentication.

Per McAfee's vulnerability disclosure policy, we reported our initial findings to B. Braun on January 11, 2021. Shortly thereafter, they responded and began an ongoing dialogue with ATR while they worked to adopt the mitigations we outlined in our disclosure report.

This paper is intended to bring an overview and some technical detail of the most critical attack chain along with addressing unique challenges faced by the medical industry. For a brief overview please see our summary blog here.

Table of Contents

Background

The most important part of any product assessment is a solid understanding of the purpose and function of the product under test. Without this it is simply too easy for research to produce less than meaningful results. Therefore, for this research it is first important to answer these few simple questions. What are infusion pumps? What security research has already been performed?

What are Infusion Pumps?

To start with the basics using a trusted resource – fda.gov says "An infusion pump is a medical device that delivers fluids, such as nutrients and medications, into a patient's body in controlled amounts." The FDA goes on to explain they are typically used by a "trained user who programs the rate and duration". Infusion pumps can be simple, administering a single intravenous (IV) medication in the home setting, or complex, delivering multiple medications simultaneously in the ICU setting. From the 1960's to 2000 infusion pumps were mostly electromechanical devices with some embedded electronics, but the turn of the century delivered "smarter" devices with better safety mechanisms and the possibility to program them, which slowly opened the door to information security challenges. Cross referencing the specific product we have chosen to look at, the Infusomat® Space® Large Volume Pump (Figure 1), we see that this pump is meant only for a medical setting and not designed for a home user. Infusion pumps exist mostly to remove the need to perform manual infusion, which requires dose conversion into drops per minute and visually counting drops to set a rate which is both time consuming and unreliable. It is estimated that there are over 200 million IV infusions administered globally each year, and 2020 sales of IV pumps in the US

were at $13.5 billion. Clearly infusion pumps have cemented their place in the medical world.

Figure 1: B. Braun Infusomat Pump

What Security Research has Already Been Performed?

Since infusion pumps are such a large part of the medical field and there are several different types, it is reasonable to expect our team is not the first to inquire about their security. As expected, there have been many different research projects on infusion pumps over the years. Perhaps the most well-known research was presented in 2018 at Blackhat by Billy Rios and Johnathan Butts. The infusion pump portion of their research was focused on the Medtronic insulin pumps. They found they were able to remotely dose a patient with extra insulin due to cleartext traffic and the ability to issue a replay attack. Even earlier, in 2015 research was published on the Hospira Symbiq Infusion Pump showing that it was possible to modify drug library files and raise dose limits through "unanticipated operations", although authentication was required.

Of course, for our purpose, the most important question remains – is there any previous research performed on our specific device. Initially the answer was no; however, during our research project a very large study, ManiMed, was released under the aegis of German authorities to examine the security of network-connected medical devices produced or in use in their country. This included research done on the B. Braun Infusomat pump. This is a fantastic piece of work which covers many network-connected devices. We will reference this study and talk about their findings where appropriate throughout this document, as we additionally explore our enhancements to this research and demonstrate a new attack that was previously called impossible.

Project Motivation

If we consider the Background section earlier, it becomes apparent there is still a large amount of critical research to be performed in this space. Infusion pumps are a prominent and continuously developing area within the medical device space, where previous research has only scratched the surface. Due to the potential critical impact and the state of medical device security, many previous projects didn't need to dig very deep to find security issues or concerns. The infusion pump industry has numerous devices which have not been researched publicly at all, and even more that only received a cursory analysis from the information security community. For these reasons, we decided to have an in-depth look at one of the largest infusion pumps vendors, B. Braun, and specifically focus on one of their devices used worldwide to analyze it at a depth never seen before. Tackling every aspect of this pump, we wanted to answer the basic question: In a realistic scenario, leveraging original security vulnerabilities, could a malicious attacker impact human life?

System Description

For this research project our system consisted of three main components– a B. Braun Infusomat Large Volume Pump Model 871305U (the actual infusion pump), a SpaceStation Model 8713142U (a docking station holding up to 4 pumps) and a software component called SpaceCom version 012U000050. These models and the corresponding software for the B. Braun Infusomat system were released in 2017. In industries such as consumer electronics, this would be considered obsolete and

therefore less relevant to research. However, as discussed above, in the medical field this is simply not the case. Since older devices are still widely used and perhaps originally developed with a less emphasis on security, it increases the importance of investigating them. For due diligence, we consulted and confirmed with our industry partners that this specific model was still actively being used in hospital systems across the country.

SpaceCom is an embedded Linux system that can run either on the pump from within its smart-battery pack or from inside the SpaceStation. However, when the pump is plugged into the SpaceStation, the pump's SpaceCom gets disabled. We performed most of our research with the pump attached to the SpaceStation as we found this was the most common use case. If a SpaceStation was compromised, it could potentially affect multiple pumps at once. SpaceCom acts as the external communication module for the system and is separated from the pump's internal operations, regardless of where it is running from.

If we consider the pump attached to the SpaceStation as one system, it has three separate operating systems running on three distinct chipsets. SpaceCom running on the SpaceStation runs a standard version of Linux on a PowerPC chipset. The WIFI module for the SpaceStation also runs a standard version of Linux on an ARM chipset and communicates over a PCI bus with SpaceCom. Lastly, the pump runs its own custom Real Time Operating System (RTOS) and firmware on a M32C microcontroller. An additional microcontroller is used to monitor the M32C microcontroller, but this goes beyond the scope of our research. Due to this modular and isolated design, the Spacecom communication module and the pump need a dedicated path for exchanging data. This is resolved via a CAN bus, shared throughout the SpaceStation, where it allows pumps and accessories to communicate with each other. This is what SpaceCom and any pump docked into the Space Station rely on for their exchange. An architecture diagram below helps demonstrates the system layout and design when a pump is present in the docking station.

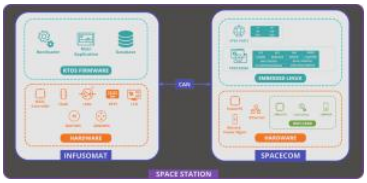

Figure 2: System Architecture

SpaceCom Functions and Software Components

SpaceCom contains many different pieces of propriety software and applications to support the many functions of the larger B. Braun and medical facility ecosystem. Our team spent time analyzing each one in great detail; however, for the purpose of this paper we will only touch on key components which are important to the most critical findings mention in the opening summary.

An important function of SpaceCom is to be able to update the drug library and pump configuration stored on the pump. The drug library contains information such as ward and department, a list of pre-configured drugs with their default concentrations, information messages to be printed on the screen when selected, and more importantly, soft, and hard limits to prevent medication error. One of the biggest selling points of the smart infusion pumps is their ability to prevent incorrect dosing of drugs, which is partly done through the limits in the drug library. Another risk the drug library helps mitigate is human error. By having the most common dosage and infusion lengths preprogrammed into the pump, it eliminates errors associated with rate calculations, and drop counting previously mentioned, associated with manual infusion therapy.

The pump RTOS contains a database of over 1500 key/value pairs used during operation. This data consists of everything from status about current components, battery life, motor speed, alarms and values used for tube calibration. As such, this data would be considered extremely sensitive in the context of the pump's operation and is not intended to have direct user interaction, nor is it presented to the user. A subset of the keys can be indirectly modified via a dedicated servicing software by certified technicians.

To interact with both the drug library and pump configuration on the pump from SpaceCom, a propriety binary called PCS is used. The PCS binary uses the *canon* binary to interface with the CAN bus to send commands to the pump's system for both reading and writing values based on the drug library or pump configuration provided to it. The main interface to accomplish this task is via a propriety TCP networking protocol, which by default is sent over port 1500. This protocol is both unauthenticated and unencrypted and we relied heavily on these weaknesses for our research and attacks. Additionally, this resulted in the filing of CVE-2021-33882 and CVE-2021-33883 as stated in the overview above.

Critical Attack Scenario Details

Goals

What could be the goal of a malicious attacker? Realistically speaking, most attacks have been proven to be financially motivated. When translating this to our infusion pump, the question becomes: What would medical executives, without hesitation, pay large sums of money for? If we look at recent events, in May of 2021, Colonial Pipeline paid hackers 4.4 million dollars to get their oil pipeline running again from ransomware attacks. Attacks on healthcare settings are increasing with the FBI estimating a cyberattack using "Ryuk" ransomware took in $61 million over a 21-month period in 2018 and 2019. Attacks are now showing potential for patient harm with one example beginning on October 28th, 2020. The University of Vermont Health Network was part of a larger coordinated attack on multiple US healthcare which resulted in a complete loss of their electronic medical record system for weeks. The results of the ransomware-based attack led to 75% of active chemotherapy patients being turned away, rerouting of ambulances, and delays in testing and treatment. Considering IV pumps are directly supporting human life in some cases, it is easy to suggest an attacker could demand any "ransom" amount leveraging threats to actual patients. To accomplish this an attacker would therefore need to control the operation of the pump.

This task is easier said than done when considering the design of the pump as outlined above. The traditional "getting root" on the network component (SpaceCom) proves ineffective. To make any changes to the pump itself, an attacker needs to interact with the pump's RTOS, which is not network connected. In this section we provide an outline on how we were able to accomplish this goal by using the five reported CVEs.

Initial Access

Even though getting root access on SpaceCom will not provide us everything we need to accomplish the ultimate goal, it is still the first step. During our reconnaissance and enumeration of the system we discovered a remote interface listening at https://{ipaddress}/rpc. This interface was connected to a common open source service referred to as "json-dbus-bridge". As described on GitHub, this service "is a fast-cgi application that provides access to D-Bus. It accepts JSON-RPC calls and translates these into D-Bus calls. Any response is converted back to JSON and sent to the client." This piqued our interest since external access to the D-Bus subsystem could provide us

access to internal communication, which may have a different level of security than typical external networking.

When doing any type of vulnerability research, product security assessment or evaluation it is critical to not forget to search for existing issues in any third-party components. This is even more important since we are working on a software released in 2017. While scouring GitHub pages for the json-dbus-bridge, we noticed a format string vulnerability that was patched in 2015. Of course, we had to test if the version we encountered had the existing vulnerability.

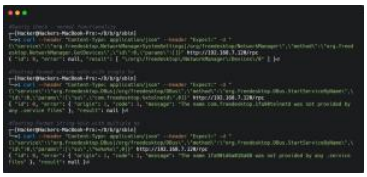

Figure 3: Format String Vulnerability Testing

The tests in Figure 3 confirmed the existence of the format sting vulnerability. While this format string vulnerability had been publicly discovered in 2015 in the json-dbus-bridge code, the update was never included in B. Braun's software and hence satisfied the condition for a vendor specific zero-day vulnerability disclosure. This was filed as CVE-2021-33886 and was our first reported discovery to B. Braun. Over the next several weeks we were able to leverage this vulnerability and create a working exploit to gain *www* user level shell access to the device. Due to the potential impact to unpatched devices, the exact technical details of our exploit have not been included.

Privilege Escalation

Although user access is the first step, root access will be needed in order to interact with the CAN bus to communicate with the actual pump. A good target and well-known process for privilege escalation is to find a binary owned by root with the setuid bit enabled. We could not find one ready to use; however, the web interface has an option to backup and export settings which relies on *tarring* a folder containing a handful of files and encrypting it with AES using a user-provided password. The backup archive can then be downloaded for later restore of the settings. When restoring this backup, root is the user doing the *untarring* in such a way that file permissions are being preserved from the provided tar file. Thus, if we can tamper with the archive, we might be able to create a privilege escalation scenario.

To use this to our advantage we need to embed a binary in the backup archive owned by root with the "setuid" bit set so we can use it to elevate privileges. Ironically, the code responsible for the import/export of settings is already doing most of the work for us. The "configExport" binary located on the filesystem is a wrapper to call setuid/setgid (and sanitize inputs) which then calls execve on the script "/configExport/configExport.sh." We can use a hex editor to change which script the "configExport" binary is running and replace "configExport.sh" with an attacker-controlled script, while also patching out the input sanitizing. We could absolutely have compiled our own binary instead, but this approach saves us from a couple of hours of PPC cross-compiling fun.

While we were working through this component of our attack chain, researchers working on the ManiMed project, in coordination with B. Braun, published a report which included this finding, listed as CVE-2020-16238 on B. Braun's website. As described in section 4.6.2.2 of their report "An authenticated arbitrary file upload vulnerability combined with an unvalidated symbolic link and local privilege escalations enables attackers to execute commands as the root user." We commend the ManiMed researchers for also discovering this vulnerability and practicing responsible disclosure.

Crossing Systems

The real work begins once root access is obtained. The challenge becomes how to affect change on the pump RTOS with root access on the SpaceCom communication module. One common approach would be to continue to look for vulnerabilities in the pump's RTOS that would lead to code execution within its system. This method poses many challenges during black box testing and could lead to damaging our limited number of test devices.

Another approach which we have leveraged in past projects is hijacking the standard functionality of the device to further the attack. This can be more manageable, but it first requires a deep understanding of how the device works and the desired outcome. This also tests the device's defense in depth and can prove to be very difficult depending on the security measures in place. In our case, this would force the question of how well-protected the area is surrounding the communication between the pump and SpaceCom.

As mentioned in the system description section above, the PCS binary is responsible for communicating with the pump's system for two critical operations – updating the drug library and updating the pump config. These are key functions that would likely be of interest to an attacker. There are several different approaches which could be taken by an attacker to interact with these key operations, especially given root access. Considering the various alternatives, we chose to leverage our root access on SpaceCom to inject code into PCS's memory and use existing functions and objects to communicate with the pump's internal system.

Our chosen path required a deep understanding of the data structures and functions used to facilitate this communication. The key is to find the perfect place in a larger operation call stack where we can modify or inject the data we want, while still utilizing lower-level functions to avoid the need to unnecessarily create objects and data from scratch. To illustrate this point, consider if we want to send a simple signal to power off the pump from within PCS's memory space. The fact that all data sent from SpaceCom to the pump's RTOS is done through CAN messages, with root access meant that we could send CAN messages directly on the CAN bus. This would require an extensive knowledge and breakdown of the CAN message structure as the underlying protocol is designed by B. Braun and would have to be reverse engineered. Although possible, it is very difficult, especially with CAN's data frame field having a lack of strict specifications. Inside PCS there is a call chain which builds this message. If we were to inject and utilize functions very low in the call chain, such as the *trySend* function which sends a CAN message (as seen in figure 4) , we would need to understand all of its arguments and the data format it uses. We'd essentially have the same problem as before.

Figure 4: trySend function

If we look higher in the call stack for a function that performs the operation we are interested in, switching off the device, we can instead let the rest of the call chain do the heavy lifting for us. Notice in Figure 5 below there is a function for just this purpose, which only requires one parameter to be passed.

Figure 5: switchOffDevice

Leveraging this concept, we are able to use the functions within PCS in a manner similar to an API to perform read

and write operations to the pump's database and force a change.

Understanding Critical Data

If we want to send and write data such as the drug library and pump config, we first need to understand the format of the data, how it is processed and any security measures in place which need to be accounted for. Our team spent extensive time reversing both the drug library and pump configuration data. A portion of the pump configuration is referred to as calibration and disposable data. Both can be modified through our attack chain; however, for this paper we will just touch on the more critical of the two the calibration and disposable data.

The calibration and disposable data are usually seen in the form of files that are living in SpaceCom. At a more granular level, they are a collection of key/value pairs that are meant to be read or written to the pump's database. Each file can also be a large blob of data living on the pump flash. The physical location of each key within this blob is hardcoded in the pump and sometimes in PCS. This representation is relevant when it comes to computing various CRCs that operate on blobs of data rather than key pairs. These checksums are used heavily throughout the pump's infrastructure with critical data to ensure the integrity of the data. This goes to ensure the safety of patients by ensuring data can't be accidently modified or corrupted. Figure 6 shows an example of disposable data as contained in files on SpaceCom.

Figure 6: Disposable Data

Looking at the variable names inside the disposable data file and relevant code in the pump firmware led us to one key/value pair that specifies the "head volume" of the tube, which can be seen in the figure above. After extensive analysis, we determined that "head volume" is the parameter dictating the amount of medication being delivered per cycle to the patient. We determined that if this value was to be changed, it could be potentially harmful. We detail this analysis in section "Unique Consideration for Infusion Pump Hacking" below.

With a target key/value pair in mind, the next step would be to understand how to calculate the CRCs. Since the system is constantly checking the integrity of the data, if an attacker wanted to modify any value, they would also need to modify the CRCs which validate the changed data. Through reverse engineering we determined the CRC was a custom implementation of a CRC16, where the initial value is 0xFFFF and relies on a hardcoded polynomial table. We were able to extract this algorithm and write custom python scripts to compute the CRC needed for the disposable data.

With a basic understanding of the critical operational data and the ability to compute the CRCs, we are able to leverage the PCS binary, in an API fashion to send commands to the pump to modify this data. This holds true for both the drug library and the pump configuration data. Although CRCs are great for integrity checking, they provide no security or level of trust of the where the data is coming from. This lack of origin verification is what led to the filing of CVE-2021-33885.

Final Attack Chain

If we review our attack chain, we can gain user-level access to the device without authentication or authorization. We can then escalate our privileges to root and leverage the existing functionality of the PCS binary to make modifications to the pump's disposable data. Conceptually, the process is complete; however, we can do some additional housekeeping in order to make our attack chain slightly more realistic and efficient.

Since the proprietary protocol for the PCS binary is unauthenticated, there are certain configuration options which can be modified for an attacker to make their job even easier. One of these configuration options tells the pump which server is "trusted" to receive operational data from (such as the drug library). An attacker can send a command to SpaceCom which clears the current trusted server configuration and rewrites it to an attacker-controlled server. This is not required for this attack when leveraging the format string and privilege escalation path outlined above; however, it does provide alternative methods and simplifies the attack process.

Lastly, the pump has an audible and visual notification when any configuration or drug information has been modified on the pump. Once again in the spirit of a realistic attack, a malicious attacker is going to want to be as stealthy as possible. To accomplish this, it was worth determining a method in which to clear these notifications. This process turned out to be as simple as restarting the pump after our modifications were complete. The reboot operation happens in a matter of seconds, so by using this technique, all alerts to the end user were quickly cleared. The complete attack process can be seen outlined in the diagram below.

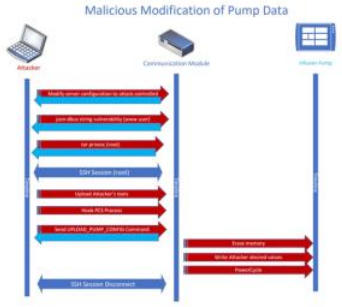

Figure 7: Complete Attack Chain

Attack Prerequisites

Although this attack chain presents a complete method to modify critical pump data, it is important to recognize the conditions required for this attack to be successful. These pumps are designed to be network connected to a local internal network. Therefore, under normal operating conditions an attacker would need to have found a method to gain access to the local network. Could this attack take place over the internet? Technically speaking, yes; however, it would be very unlikely to see a setup where a pump is directly internet-connected.

In addition to being on the local network, the pump does have safeguards in place to ensure no modifications can occur while the pump is operational. From what we discovered during our research, if the pump is actively administering medication, it ignores any request on the CAN bus to modify library or configuration data. This means the attack can only be successful when a pump is idle or in standby mode in between infusions.

Impact

The prerequisites for this attack are minimal and are not enough to mitigate the overall threat. In today's world there are a wide range of documented and utilized methods for attackers to gain access to local networks. If we also consider that hospital or medical facilities are generally public places with little to no barriers to entry, it is easy to see how someone malicious can go unnoticed and obtain network access. Pumps are also not always

actively administering mediation. Even in the busiest of hospitals there is downtime between patients or times when pumps are simply not in use.

With the ability to modify disposable and configuration data on the pump, there are a wide range of possibilities for which an attacker could choose to have an impact. An attacker could simply put the device in an unusable state or write arbitrary messages on the screen. We chose to focus on the disposable data, specifically the key/value pair labeled "TUBE_HEADVOLUME_A" since we determined it would demonstrate the greatest impact, bringing harm to a patient. In the below video you will first see the pump under normal operation. After demonstrating the system working as intended, we modify the configuration remotely using the attack chain explained above and then illustrate its effect on the pump when administering medication.

Demo

Unique Considerations for Infusion Pump Hacking

An interesting characteristic of this project is that its impact and consequences are inherently grounded in the physical world. Where common software hacks end with the ability to get root access or kernel privileges, in this project, the way the device is used by medical staff and how it can affect patient safety is crucial to the outcome. The next few sections will focus on various aspects of the project that fall under this umbrella.

Why we modified TUBE_HEADVOLUME

As described previously, our attack relies on modifying the disposable data that governs the way the pump is used to deliver medication. But why and how did we decide to go investigate this? An interesting side-effect of the pump being built to be safe is that most of the inputs and outputs it receives from the CAN bus are extensively checked against out-of-range access. From an attacker's perspective who has already compromised SpaceCom, this would usually be the prime target for memory corruption bugs. Fuzzing and emulating the M32C architecture is cost-heavy in terms of upfront work, so instead, we started looking for a path of least resistance and searched for blind spots in the secure design.

In our case, we wanted to be able to affect the amount of drug being dispensed, preferably without having something on screen as that would indicate a malfunction or abnormality. Our original plan was to tamper with the device drug library, but it turns out that data we could alter would be displayed on screen, which could raise concern as medical staff verify the prescribed drug and rate against the order before, and immediately after starting the infusion. This would not be ideal for an attacker, so we kept investigating. The other files we could modify were the calibration data and the disposable data. These files are interesting as they describe internal parameters; the calibration one specifies the physical parameters of the device itself, while the disposable one is for the specifics regarding the tubing going through the pump. Anyone familiar with precision tools know how important a good calibration is. If the calibration is off it will lead to improper operations or results. From an operational

standpoint this makes sense, but from an attacker perspective this has a strong likelihood of fitting the bill for the attack we had in mind: modifying an internal value so the pump thinks it is dispensing the right amount of drug, while it is actually incorrect in its calculations.

Looking at the variable names inside the disposable file and relevant code in the pump firmware led us to one that specifies the "head volume" of the tube. From our understanding, each time the pump pumps, it compresses the IV tubing thereby pushing a small quantity of drug towards the patient. Overall, there are many physical parameters that would govern this volume -the internal tube diameter, the length of the compressed region, how much the tube is being compressed, etc.—but in the end, it seemed that all these values were summed up in one variable. Cutting this value in half would make the pump believe it is pushing half the actual amount, and therefore would have to pump twice as fast to deliver it. We tried our hypothesis, and by doing so, the amount of drug dispensed doubled while the pump assumed everything was normal.

Operations in Hospitals and Consequences of Over-Infusing Drugs

Now that we have an idea of what happens to the device when we alter its internal configuration, we can consider how this could play out in the real world. As mentioned previously, medical staff are expected to be extra-careful when using these devices, ensuring the numbers match the doctor's order. In the United States, both the Centers for Medicare and Medicaid Services (CMS) and the American Society of Clinical Oncology require standard of practice be followed with high risk or hazardous infusions like blood or chemotherapy. This standard requires two appropriately trained people (usually nurses), one who will be infusing the medication, and the other to verify the order and configuration prior to administration. Looking internationally, we were also able to find this same protocol in use at an Irish hospital. It confirms the attention to detail and the requirement to double-check each value is correct. However, another document describing the adoption of a smart pump system in a Swedish hospital hints at concerns (p. 47) that invalid drug protocols might be followed if a nurse picked the wrong default settings on the pump. These documents are anecdotal, but the overall feeling is that strong checks are in place. Under pressure or with multiple infusions, mistakes can be made, which smart pumps should prevent.

One of our industry partners, Shaun Nordeck, M.D. is an Interventional Radiology Resident Physician at a Level 1 Trauma Center and prior, served as an Army Medic and Allied Health Professional. Leaning on more than 20 years in the medical field. Dr. Nordeck states "A high-pressure environment such as the ICU may be at increased risk for infusion errors since these critical and often medically complex patients have multiple infusions which are being adjusted frequently. Errors, however, are not limited to the ICU and may just as easily occur in the inpatient ward or outpatient settings. Essentially with each increase in variable (patient complexity or acuity, number of medications, rate changes, nurse to patient ratio, etc.) there is an increased risk for error."

As a measure of safety, it is important to keep in mind that one can visually count the number of drops to verify the infusion rate (there's even an optional module to do it automatically). However, depending on the parameters, a minor change of speed (e.g., halved or doubled) might not be immediately obvious but could still be deleterious. Dr. Nordeck further stated that "something as routine as correcting a person's high blood sugar or sodium level too quickly can cause the brain to swell or damage the nerves which can lead to permanent disability or even death." The FDA's MAUDE database keeps track of adverse events involving medical devices and can be used to see what type of problems actually occurred in the field. Certain

drugs are particularly potent, in which case the speed at which they are delivered matters. In this instance, an over-sedation at 4 times the intended rate led to the death of a patient a few hours after the incident occurred. Under-dosing can also be problematic as the required medication does not reach the patient in the appropriate quantity. These examples highlight that a pump not delivering the correct amount of drug occurs in the field and may remain unnoticed for multiple hours, which can lead to injury or death.

Common Pitfalls

Let's now take a step back and consider some generic shortcomings that became apparent while looking at the infusion pump ecosystem. We believe these problems are not specific to a brand or a product but rather may be found across the entire medical field. This is because throughout the years, this vertical has only received a limited amount of attention from both malicious actors and the cybersecurity industry. With the increased rate of cyber threats and the constant additions of new smart devices in private networks, new attack surfaces are being exposed and the hardening of many systems may turn into low hanging fruits for the ones lagging. The slower life cycle of smart medical devices means that best security practices and mitigations take longer to be adopted and deployed in the field. Awareness of this may help healthcare organizations, and their supporting IT administration have a more critical eye on the technology deployed in their environments while medical device vendors should remain vigilant of their "legacy" technologies and continually reassess the risk profile associated with legacy products in the current cybersecurity landscape.

Patching is Costly

Consumer products, both hardware and software are often nimbler than their counterparts in the medical industry. Your web-browser or operating system on your personal computer will auto-update immediately after a patch is released which come on a regular basis. This is radically different for medical devices which are often directly linked to patient safety and therefore need to undergo a more rigorous vetting process before applying updates. This often leads to the need to immobilize devices during updates, perform follow up tests and recalibrations. It is often very expensive and challenging for medical facilities to update products, resulting in deployed devices with firmware that is several years old. Because of this, "table stakes" security measures may never be fully adopted, and corresponding vulnerabilities may have a larger impact than in other industries.

Designed for Safety Rather than Security

When looking at the general architecture of the pump, it is obvious that it was designed with safety in mind. For instance, it relies on an application processor for the main processing but also has a control processor that makes sure nothing unexpected occurs by monitoring sensors output along with other components. Everything is CRC checked multiple times to flag memory corruption and every range is bounds-checked. All of this suggests that the design was intended to mitigate hardware and software faults, data accidentally being corrupted over the wire, and the flash module degrading which aligns with a high priority on safety.

However, it looks like preventing malicious intent was not given as much attention during the design process. Sometimes the difference between safety and security might be a little blurry. Preventing accidental memory corruption and out of bounds access due to faulty hardware will also make exploitation harder, yet an attacker will always attempt to escape these mitigations. Along the same lines, logic bugs that would be extremely unlikely to occur by chance might be the "keys to the kingdom" for an attacker. Internal audits and offensive security exercises can highlight the attacker mindset and

bring valuable insights as how to harden existing safeguards to protect against intentional threats.

Everything is Trusted

When looking at how the pump and its communication module handles communication and file handling, we observed that critical files are not signed (CVE-2021-33885), most of the data exchanges are done in plain-text (CVE-2021-33883), and there is an overall lack of authentication (CVE-2021-33882) for the proprietary protocols being used. There are a few password-protected areas for user facing systems, but not as many for the behind-the-scenes internal systems. This might be because a login page on a website is an "obvious" necessity, along with having a proper authentication mechanism for FTP and SSH, while ad-hoc protocols designed more customized uses are not as obvious. There is also an evolving landscape at play and its related threat assessment; the risk of an unauthorized person tampering with a configuration file (calibration data, drug library, etc.) is fairly low if it also requires dedicated software and physical access to the device. However, if suddenly the device becomes network-connected, the attack surface is extended and the original assumptions may not be refreshed. Defense-in-depth would dictate that in any case, important files should not be easy to tamper with. However, security vs functionality comes with legitimate compromises and when it comes to embedded devices, limited resources and usability also need to be factored into the equation.

CAN gets Connected to WIFI

Originally, the CAN bus was reserved for communication between trusted components such as a Servicing PC used for maintenance or for connecting multiples devices within an older model of the Space Station that did not have SpaceCom built in. The latter would come as an optional module that could be plugged into the Space Station to offer external connectivity. Hence, the CAN bus was used for "internal" communication between trusted components and an external module, the SpaceCom, could be added for data reporting over the network. Over the following decade, technology improved and miniaturized to the point where everything got merged, so that even a battery module could provide WIFI connectivity and the SpaceCom functionalities. This opened new possibilities, such as having the built-in SpaceCom module provide similar capabilities as the servicing PC. From a user perspective this is great as it simplifies operations, but from a security perspective, this created a situation where a "trusted" internal network suddenly became bridged to an external network that could even be accessed wirelessly. What might have been an acceptable risk, where only a few proprietary devices with physical access could perform privileged operations, became much more questionable when a WIFI-connected Linux device started to offer the same capabilities.

This kind of problem has been faced by nearly every industry vertical that evolved from reliance on trusted physical networks which suddenly got connected to the internet or other untrusted networks. Smart connected devices are a double-edged sword: in the same way they offer greater flexibility and synergy between systems, they can also lead to emergent security issues that need to be considered holistically.

Technical Debt

When developing custom protocols and ad-hoc systems it's natural to incur technical debt. This is even more true when the life cycle of a device is many years and when it is complicated and expensive to deploy patches and upgrades, leading to a heterogeneous customer base and multiple hardware revisions to support. This can cause situations where more obscure features are not looked at for years and their ownership might be lost or perfunctory.

An example of this is the format string vulnerability affecting the json-dbus module. Its usage is obscure, and it was forked from an open-source project many years ago. The original repository fixed bugs that were security bugs but were not flagged as such which led them to fly under the radar for multiple years. Likely, at the time it was forked, the code served its purpose and was never revisited afterwards, leaving the security bug unnoticed. The same can be said for custom-designed protocols and file formats. It may be difficult to evolve them in line with the improvement of best security practices while avoiding breaking "legacy" deployments. In this scenario, mitigations might be the way to go; making sure the systems are isolated, unnecessary features can be disabled and their privilege and access limited to what's needed. Future-proofing a system is a difficult challenge. If anything, transparency on how the system functions and the components it relies on, coupled with regular audits (code source review or black box audit) can help prevent components from falling in the cracks where they're not checked against best practices for many years.

Conclusion

This concludes a research project which took two senior researchers a significant amount of time to showcase a life-threatening risk of a medical device being taken over by a remote attacker. For the time being, ransomware attacks are a more likely threat in the medical sector, but eventually these networks will be hardened against this type of attacks and malicious actors will look for other lower-hanging fruits. Given the lifespan of medical devices and the difficulties surrounding their updates, it is important to start planning now for tomorrow's threats. We hope this research will help bring awareness to an area that has been a blind spot for far too long. Dr. Nordeck affirms the importance of this research stating: "The ability to manipulate medical equipment in a way that is potentially harmful to patients, without end-user detection, is effectively weaponizing the device and something only previously conceived by Hollywood yet, McAfee's ATR team has confirmed is plausible. Device manufactures clearly aim to produce safe and secure products as evidenced by built-in safeguards. However, flaws may exist which allow the device to succumb to a ransom attack or potentially cause harm. Therefore, manufactures should collaborate with security professionals to independently test their products to detect and correct potential threats and thereby preserve patient safety and device security."

Performing regular security audits, making it easier for medical professionals to keep their devices up to date and offering solid mitigations when this is not possible should really be on every medical vendor's list of priorities. Medical professionals, policy makers and even the general public should also hold accountable the medical vendors and have them clearly articulate the risk profile of the devices they sell and demand better ways to keep their device secure. We recognize even with this mindset and a holistic approach to security, there will always be flaws that cannot be predetermined. In these cases, vendors should encourage and even seek out industry partners, embrace responsible disclosure and communicate broadly with researchers, stakeholders and customers alike.

From a security research perspective, it is crucial to understand how a device works at a holistic system level, and how each component interacts with each other, which components they can talk to, and so on. For manufacturers, it is important to read between the lines; something may not be in a design document or in the specifications, but sometimes emergent properties will occur as a side-effect of other design decisions.

An offensive project like ours is really meant to highlight structural weaknesses and point out risks. Now, defensive work is necessary to address these concerns. For instance, manufacturers should leverage cheaper and more powerful microcontrollers to implement proper authentication mechanisms. However, it is even more important to study and address the challenges hospitals face when it comes to keeping their devices up to date.

This should come as both technical solutions from the vendors and advocacy to promote secure practices and raise awareness on the underlying risks associated with critical devices having outdated software. The FDA tried to lead the way in 2018 with its CyberMed Safety (Expert) Analysis Board (CYMSAB), but so far little progress has been made. The work the German BSI did with the ManiMed project is also extremely encouraging. We see this as an area of cybersecurity with lots of potential and need for attention and look forward to the information security industry taking on this challenge to make this critical sector always more secure.

One goal of the McAfee Advanced Threat Research team is to identify and illuminate a broad spectrum of threats in today's complex and constantly evolving landscape. As per McAfee's vulnerability public disclosure policy, McAfee's ATR team informed and worked directly with the B.Braun team. This partnership resulted in the vendor working towards effective mitigations of the vulnerabilities detailed in this blog. We strongly recommend any businesses using the B.Braun Infusomat devices to update as soon as possible in line with your patch policy and testing strategy.

CVE Details

CVE: CVE-2021-33882

CVSSv3 Rating: 6.8/8.2

CVSS String: AV:N/AC:H/PR:N/UI:N/
S:C/C:N/I:H/A:N/CR:H/IR:H/AR:M/MAV:A

CVE Description: Missing Authentication for Critical Function vulnerability in BBraun SpaceCom2 prior to 012U000062 allows a remote attacker to reconfigure the device from an unknown source through lack of authentication on proprietary networking commands.

CVE: CVE-2021-33883

CVSSv3 Rating: 5.9/7.1

CVSS String:
AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:H/A:N/CR:H/IR:H/AR:M/MAV:A

CVE Description: Cleartext Transmission of Sensitive Information vulnerability in BBraun SpaceCom2 prior to 012U000062 allows a remote attacker to obtain sensitive information by snooping the network traffic. The exposed data includes critical values for the pumps internal configuration.

CVE: CVE-2021-33884

CVSSv3 Rating: 7.3/5.8

CVSS String:
AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L/CR:M/IR:M/AR:L/MAV:A

CVE Description: Unrestricted Upload of File with Dangerous Type vulnerability in BBraun SpaceCom2 prior to 012U000062 allows remote attackers to upload any files to the /tmp directory of the device through the webpage API. This can result in critical files being overwritten.

CVE: CVE-2021-33885

CVSSv3 Rating: 10.0/9.7

CVSS String:
AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N/CR:H/IR:H/AR:M/MAV:A

CVE Description: Insufficient Verification of Data Authenticity vulnerability in BBraun SpaceCom2 prior to 012U000062 allows a remote unauthenticated attacker to send malicious data to the device which will be used in place of the correct data. This results in execution through lack of cryptographic signatures on critical data sets

CVE: CVE-2021-33886

CVSSv3 Rating: 8.1/7.7

CVSS String: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N/RL:O/RC:C

CVE Description: Improper sanitization of input vulnerability in BBraun SpaceCom2 prior to 012U000062 allows a remote unauthenticated attacker to gain user level command line access through passing a raw external string straight through to printf statements. The attacker is required to be on the same network as the device.

RECENT NEWS

Dec 7, 2022
Trellix Predicts Heightened Hacktivism and Geopolitical Cyberattacks in 2023

Nov 30, 2022
Trellix Expedites Delivery of XDR with AWS

Nov 16, 2022
Ransomware Activity Doubles in Transportation and Shipping Industry

Sep 28, 2022
Trellix Expands XDR Platform to Transform Security Operations

Sep 28, 2022
Trellix Empowers Next Generation of Cybersecurity Talent at Xpand Live

RECENT STORIES

Dec 7, 2022
The Bug Report – November 2022 Edition

Nov 30, 2022
Trellix Enhances Zero Trust with AWS Verified Access

Nov 29, 2022
Go Jump in the Lake: Trellix leverages Amazon Security Lake for its Extended Detection and Response (XDR)

Nov 22, 2022
Yanluowang Ransomware Leaks Analysis: Organization, Collaboration with HelloKitty, Babuk and Conti

Nov 19, 2022
Email Cyberattacks on Arab Countries Rise in Lead to Global Football Tournament

Featured Content

PERSPECTIVES
Our CEO On Living Security
By Bryan Palma · January 19, 2022

Trellix CEO, Bryan Palma, explains the critical need for security that's always learning.

Read More

XDR

Time to Drive Change by Challenging the Challengers

By Michelle Salvado · January 19, 2022

Dynamic threats call for dynamic security – the path to resiliency lies in XDR.

Read More

THREAT LABS

2022 Threat Predictions

By **Trellix** · January 19, 2022

What cyber security threats should enterprises look out for in 2022?

Read More

Get the latest

We're no strangers to cybersecurity. But we are a new company.
Stay up to date as we evolve.

Email

Submit

Zero spam. Unsubscribe at any time.