How to Secure Consumable Components of Medical Devices and Other Embedded Systems

by Robert Van Rooyen

March 3, 2013

Prepared By
Barr Group
20203 Goshen Road, #324
Gaithersburg, MD 20879
Website: barrgroup.com
Phone: +1 (866) 65-EMBED

Barr Group Contact: Andrew Girson, CEO
Phone: +1 (301) 802-5527
A number of embedded systems, such as medical devices and printers, feature replaceable components designed to be installed new, consumed through one or more cycles of product use, and ultimately disposed. Generally speaking, security is an important issue for designers of such systems. Whether to ensure against the spread of infectious diseases, guarantee a certain level of product reliability, or generate a steady stream of revenues (or some combination), a common security goal is to ensure that third parties and end users cannot clone or recondition consumables. This article describes the security challenges associated with the design of such consumable (a.k.a., disposable) components as well as practical ways to meet these challenges.

One key aspect of the architecture of a system with consumable components is the design of a secure communications protocol between the terminal or “base station” and the consumable. Another important consideration is that a would-be attacker typically has physical possession and control of at least one terminal station and potentially an unlimited supply of consumables with which to experiment and perform reverse engineering activities. Finally, designers must recognize that security is never perfect and always an arms race; upgradeable security should thus be considered so that additional security can be retrofitted in if/when attackers find the “weakest link”. Of course, all of these aspects of security must be addressed in the context of the limited processing power and memory of embedded systems, particularly within a necessarily low-cost disposable component ultimately destined for a landfill.

**Authentication Protocols**

At a minimum, a secure communications protocol must provide the ability of the base unit to distinguish an authorized consumable from a clone, a mechanism to limit usage time and/or count uses of the consumable, and allow the base unit to reject spent consumables. So that we may keep the following discussion concrete, consider the design of a medical device, such as that shown in Figure 1, which consists of a terminal with which a doctor interacts through a touchscreen GUI and a single-use disposable probe that interfaces with the patient.
First, it is critical that the design of secure consumables provide for a mechanism whereby the terminal can identify the consumable as authentic. For example, one aspect of authentication might be that each consumable contains secret information known only to engineers working for the manufacturer of the system. Furthermore, each consumable might contain a serial number or encryption key that is unique. A terminal which verifies that the attached consumable knows certain facts may conclude that the consumable is authentic. Importantly, however, secrets must be carefully protected if they are sent over a communications channel and even if they are merely at rest in memory. A determined (and well-funded) attacker, such as a competitor, can generally observe real-time communications over cables between the terminal and consumable, examine the contents of read-only memory areas, and disassemble executable code.

In addition, once the medical procedure has been completed the attached consumable must be somehow “marked” as used. This either could be completed automatically within the consumable as the medical procedure is performed or via a command from the terminal following the procedure. The first approach is generally more secure, but both approaches have tradeoffs, as, for example, one common scenario with a medical consumable is that the limitation of “single use” may be defined as one successful test
on a single patient—which requires a level of involvement of the device operator in the marking process and also might involve several runs of the medical procedure using the same consumable.

Finally, the number of uses (or, in more general terms, the degree to which the consumable has been spent) must be ascertainable from the consumable so that the terminal is able to reject an authentic product that should be disposed based on its level of use.

The designer of a protocol which supports authentication, marking, and use count detection of consumables must prevent the possibility of certain well-known protocol attacks. For example, in a classic “replay attack” the designer of a clone device could simply record the responses of an authentic and unspent consumable to the various commands from the terminal and replay those same messages from the clone. Alternatively, a third party might attach a spent consumable to the base unit by way of a “man-in-the-middle” device that would intercept and sometimes alter otherwise genuine communications between the two endpoints—as illustrated in Figure 2.

![Figure 2 - Man-in-the-middle device](image-url)
The use of an off-the-shelf Authentication chip can be placed on the consumable to provide a tamper-proof private key management solution. These types of devices typically offer multiple security models, such as a random challenge/response, that are based on open standards and can eliminate the need for a processor on the consumable. Some manufacturers offer implementations that use a single line for communication and power along with a ground reference, which can be a real advantage when pins or size is at a premium on the consumable connection. Application specific data storage is also available and can be configured for “one time programming” mode to track usage. This specific feature can be employed to prevent attackers from resetting the usage count on a spent consumable. Other methods of authentication could include monitoring the power consumption profile of the consumable and looking for uniquely identifiable behaviors.

**Reverse Engineering**

Attackers can employ a number of proven techniques to gain insight into the inner workings of a system with the goal of discovering an exploit. One of these techniques is known as a “chip rip”, whereby they physically remove the top of a chip and analyze its internal structure for data, such as a private security key. Some chip vendors offer “tamper proof” solutions for their products while contract manufacturers can use epoxy and other mechanical defenses against this type of attack. A process called “chip sanding” can also be employed to eliminate any identifiable markings, which would complicate the identification of the specific devices that were used in the system.

Another effective method used by attackers is to connect a debugger to the system processor and step through code execution in order to determine a variety of vulnerabilities. This threat can be reduced by disabling the JTAG\(^1\) port, which can be completed by either burning fuses or writing specific values to locations during device programming, or configuring the appropriate debug registers early in the boot process. It is also important to remove all debug connectors/pads along with any related test

---

\(^1\) **Joint Test Action Group (JTAG)** is the common name for what was later standardized as the IEEE 1149.1 **Standard Test Access Port and Boundary-Scan Architecture.**
points. A method to counter this type of attack is detection through the use of case interlocks, destructive enclosures, clock speed and/or voltage level monitoring, and validating the code image by the use of a digital signature. When an intrusion is detected, the system could potentially log and/or report the attack and optionally cease to operate.

Certain standalone “system on a chip” processors can offer a distinct advantage by not exposing any external memory busses that could be probed with a logic analyzer. If the attacker actually removes an external memory chip by de-soldering it from the circuit board and dumps the contents, data and address lines can be scrambled to further complicate the attacker’s interpretation of information. This can be performed by developing a simple software tool that scrambles the binary code image to match the data and address line assignments on the hardware. It should be noted however that scrambling the address and data lines will not be effective if the attacker is actually able to probe the external bus with a logic analyzer.

Given the level of attacker sophistication, designers must avoid using simple methods like recording serial numbers of each consumable in a text file on internal or removable storage. This type of approach to security is easily exploited by less-sophisticated attackers. Other methods that involve security by obscurity are also generally easy to circumvent by experienced attackers. In contrast, by carefully applying multiple levels of effective security measures, designers can slow down attackers and protect against a single point of entry if a specific tactic is weakened or broken.

**Upgradeable Security**

Consumables are now becoming more and more profitable, increasing the economic incentive for reverse engineering them. If there is sufficient economic incentive, a determined (and well-funded) attacker will find the weakest link. To limit the manufacturer’s risk, it is critical to quickly retrofit additional security. If the exploit can be prevented by modifying only the consumables going forward, that is ideal. If the exploit only can be prevented by upgrading the terminal then this must be planned for in advance. Ideally this could be accomplished through a software update, but this is not foolproof as the updates may not be applied by the end-user in a timely fashion.
Securing a software update requires authenticating the new firmware before it is installed as well as validating it after it has been programmed into flash. Another key component of upgrading the firmware is a secure boot strategy. This is a feature of the processor along with immutable and often multistage authenticated boot images that prevent unauthorized code from executing. Regardless of the software update delivery mechanisms (Internet, USB, SD Card, serial) and whether the end user or field technician performs the upgrade; authentication, validation, and secure boot is critically important to preventing an attack.

The use of code image hashes, public key signatures, and signature authority certificates are common and proven methods to ensure authenticity. There are many providers of tools, processes, and hardware that take full advantage of this standards-based approach. In addition, researchers are constantly developing new and innovative ways to increase the level of security in a variety of settings while leveraging existing standards where possible. An example of this would be the creation of SHA-2\(^2\), which increases the number of output bits in SHA-1 from 160 to 256 bits in a variant based on roughly the same underlying mathematical concepts.

**Secure Consumable Example**

One method of authenticating a consumable as described in the previous section is to use an off-the-shelf 1-Wire\(^3\) authentication chip. This approach keeps the complexity and cost down on the consumable side and shifts the burden to the terminal side in terms of implementing a random challenge/response protocol. A high-level block diagram shows the primary functional blocks as illustrated in Figure 3.

---

2 SHA-2 is a cryptographic hash function designed by the United States National Security Agency and published by the United States NIST as a U.S. Federal Information Processing Standard.

3 1-Wire is a device communications bus system originally designed by Dallas Semiconductor Corp. that provides low-speed data, signaling, and power over a single signal.
In this example, a private key is permanently stored in the authentication chip during manufacturing and is unreadable from the 1-Wire interface or through physical tampering. A random challenge is generated by the processor on the terminal side and is submitted to the authentication chip over the 1-Wire interface. The consumable provides a response back to the terminal based on the private key. The response is then compared to an expected value that was computed by the terminal’s processor using its protected copy of the same private key, which is a concept commonly referred to as “symmetric keys”. If the response matches the expected value, then the consumable is authenticated as genuine and normal system operation can continue. The terminal may also repeat this authentication process randomly to make sure that the device has not been swapped out with an unauthorized consumable during the course of a given treatment session.

If the selected processor does not provide any inherent security features, then a second authentication chip can be co-located on the terminal to provide the required functionality. Aside from a secure key storage solution, all of the needed features can alternatively be leveraged from off-the-shelf software libraries; however this will have an impact on processor performance and memory footprint.

Once the device has been authenticated and before a treatment session begins, the processor checks that the current usage count on the authentication chip data store is

As the name implies, private keys must be highly protected and only available on a need to know basis in a manufacturing chain of trust.
below a defined threshold. If the threshold is exceeded, the terminal will not begin the treatment session and consequently inform the operator that a spent consumable has been attached. Otherwise, the terminal will increase the usage count on the consumable by using the “one time programmable” feature (bits can only be set to ‘0’ once, so successive bits are used to implement a count) of the authentication chip and the treatment session will begin. On some systems, the threshold may be determined by using a timestamp with a duration limit as opposed to or in addition to a session counter.

In this example, it also may be important to encrypt patient and general communication data to/from the consumable. This can be accomplished with either the same private key, or a different key stored on the authentication chip. Depending on the type of data and rate of communications with the terminal, the consumable would likely require a dedicated local processor and/or custom hardware to implement the encryption and decryption algorithms. Conversely, the terminal would have to have sufficient resources to process the encrypted data in real time.

**Conclusion**

A consumable must be authenticated as genuine through a secure communications protocol in order to prevent unauthorized cloning. In addition, the consumable must have a method for checking and updating its usage count and/or time-stamps so that it cannot be placed in service beyond its intended life, which for most medical devices is a single “valid” use.

The use of counter-measures like removing debug connectors, tamper detection, and using tamper-proof components and/or other mechanical defenses can dramatically increase the level of difficulty for physical attacks.

Preventing unauthorized code from executing on the system requires both hardware and software support. A secure boot capability coupled with code image authentication and verification will help prevent malicious code from being executed.

Embedded system security is a continuum that requires due diligence on the part of designers to keep up with the constant barrage and ever increasing sophistication of
attacks by highly motivated groups and individuals. In this instance, the cost associated with preventing unauthorized cloning always should be weighed against the cost of a security breach. Consider all of the applicable risks such as safety, litigation, quality, terminal damage, support, reputation, and lost revenue. Can you afford to take this level of risk with your consumable?

About the Author

Robert Van Rooyen is a senior engineer at Barr Group. He earned his Master of Science in Computer Science degree at California State University Chico, and his Bachelor of Science in Computer Engineering degree at California State University Sacramento. He has over 28 years of experience developing electronics and embedded software for products including medical devices. Barr Group specializes in the design of reliable and secure embedded systems and provides training, consulting, and engineering services to product makers large and small. Mr. Van Rooyen can be reached by email at rvanrooyen@barrgroup.com.

References


Acknowledgements

The author wishes to thank Michael Barr from Barr Group for providing considerable input and guidance throughout the development of this paper as well as Andrew Girson and Dan Smith from Barr Group for their invaluable feedback.