Guidance for Use of Mobile Devices in Diabetes Control Contexts

The draft of this document was issued on February 7, 2018

Document Number: DTMOST-FEB-2018

Preface

Public Comment

You may submit electronic comments, questions, and suggestions relating to this guidance document at any time to the chairs of the DTMoSt (Diabetes Technology Society Mobile Platform Controlling a Diabetes Device Security and Safety Standard) committee:

David Klonoff (Chair): dklonoff@diabetestechnology.org
David Kerr (Chair): dkerr@sansum.org
David Kleidermacher (Technical Chair): dkleidermacher@google.com

Identify all comments with the document number listed in the title page.

Additional Copies

Additional copies of this document are available from the Internet. You may also send an e-mail request to the contacts listed above to receive a copy of this guidance.

Acknowledgements

The DTMoSt Chairs, David Klonoff (Mills-Peninsula Medical Center), David Kerr (Sansum Diabetes Research Institute), and Dave Kleidermacher (Google), and Assistant Chair, Barry Ginsberg (Diabetes Technology Consultants), would like to thank the members of the steering committee and their organizations for their contributions towards the creation of this guidance, including:

Aiman Abdel-Malek (Insulet), David Armstrong (University of Southern California), Guillermo Arreaza-Rubin (NIDDK / NIH), Joshua Balsam (FDA), Stayce Beck (FDA), Don Boyer (BOYER@RegulatorySolns), Carole Carey (IEEE), Joe Chapman (MITRE), Penny Chase (MITRE), Elvis Chan (FBI), Kong Chen (NIDDK / NIH), Sammy Choi (US Army), Mark Coderre (OpenSky), Barry Conrad (Stanford), Keesha Crosby (Tri-Guard Risk Solutions), Eyal Dassau (Harvard), Sheldon Durrant (MITRE), Anura Fernando (UL), Joseph Fernando (ARM), Justin Fisher (Booz Allen Hamilton), Brian Fitzgerald (FDA), Mike Golden (Samsung), Christian Howell (DHS), Christopher Keegan (Beecher Carlson), Lisa Kerr (Australian Government Department of Health), Mandeep Khera (Consultant), Michael Kirwan (IEEE & Continua), Boris Kovatchev (UVA), Jeffrey LaBelle (ASU), Benjamin Lee (Flex), Luis Malave (EOFlow), Bryan Mazlish (Bigfoot Biomedical), Laurel Messer (University of Colorado), Uwe Meyer (TÜV Rheinland), Thomas Miller (Novo Nordisk), John Oberlin (US Air Force), Irina Nayberg (Mills-Peninsula Medical Center), Dale Nordenberg (MDISS), Yarmela Pavlovic (Hogan Lovells), Matt Petersen

(ADA), Patrick Phelan (UCSF), Gil Porat (Abbott Diabetes Care), Azhar Rafiq (NASA), Kelly Rawlings (Vida Health), Jeffery Reynolds (Ascensia Diabetes Care), J.P. Ribeiro (Insulet), Linda Ricci (FDA), Naomi Schwartz (FDA), Jennifer Sherr (Yale), Christine Sublett (Sublett Consulting), Michael Taborn (Intel), Eugene Vasserman (Kansas State University), Alicia Warnock (US Navy), Tim West (Atredis Partners), Eric Winterton (Booz Allen Hamilton), Michael Wiseman (Australian Government Department of Health), Jonathan Woo (EOFlow), Margie Zuk (MITRE).

Abbreviations

American Diabetes Association (ADA)
Application Programming Interface (API)
Central Processing Unit (CPU)
Consumer Mobile Device (CMD)
Department of Homeland Security (DHS)
Diabetes Technology Society Cybersecurity Standard for Connected Diabetes Devices (DTSec)
Diabetes Technology Society Mobile Platform Controlling a Diabetes Device Security and Safety Standard (DTMoSt)
Federal Bureau of Investigation (FBI)
Food and Drug Administration (FDA)
IEEE (Institute of Electrical and Electronics Engineers)
International Electrotechnical Commission (IEC)
International Organization for Standardization (ISO)
Joint Tactical Radio System (JTRS)
Mobile Device Fundamentals Protection Profile (MDFPP)
National Aeronautics and Space Administration (NASA)
National Information Assurance Partnership (NIAP)
National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK)
National Institutes of Health (NIH)
Personal Area Network (PAN)
Protection Profile for Connected Diabetes Devices (CDD PP)
Real-time operating systems (RTOS)

1. Introduction

The need to assure medical device functionality and safety has become more challenging with the growing use of wireless and Internet-connected devices. For example, can safe operation of the device be impacted by loss of wireless connectivity due to interference or malicious jamming? Indeed, an important component of safety assurance is security assurance: ensuring that malicious attacks against these devices (e.g. via their network connections) do not adversely impact functionality and safety.

In addition, there is significant increased use of off-the-shelf consumer mobile devices (CMDs), (e.g. iPhones and Android smartphones) in medical contexts. While these contexts have historically been limited to monitoring rather than control of the medical device and its safety functions, there is increasing demand for the use of such mobile devices for control applications. For example, the use of a smartphone app can replace a custom insulin pump remote controller, reducing time-to-market and cost of new treatments while providing for an improved user experience and quality of life for people with diabetes.

In order to realize the potential beneficial uses of consumer digital technology, the medical community, including device manufacturers, regulators, caregivers, and patients must be aware of the risks associated with the use of CMDs and apps in these contexts and follow appropriate regulatory, developmental, lifecycle management, and usage guidelines to ensure that proper functionality and safety are maintained.

This guidance has been developed by a multi-stakeholder community consisting of the FDA, independent cybersecurity experts, consumer technology developers (e.g. smartphone developers, smartphone operating system developers, and smartphone chipset developers), diabetes device developers, medical research funding agencies, physicians, educators, consumers, regulatory experts, liability attorneys, policy experts, and more. This guidance has been developed to identify issues and best practices relating to CMD use in medical contacts. The same stakeholder groups and other applicable interested parties should consider this guidance in the design, development, evaluation, approval, management, deployment, and use of CMDs in medical control contexts.

The recommendations contained in this guidance are intended to supplement existing standards and guidance, including FDA recognized standards such as *ISO/IEC 62304* and FDA guidance such as the *Content of Premarket Submissions for Management of Cybersecurity in Medical Devices*. These guidelines describe current consensus thinking of the DTMoSt committee membership on this topic and should be viewed only as recommendations, unless specific regulatory or statutory requirements are cited. The use of the word ***should*** means that something is suggested or recommended, but not required.

2. Scope

The intent of this document is to provide guidance for the safe use of CMDs in the control of diabetes-related medical devices. While this guidance may be applied for other medical use cases, it has been developed specifically for diabetes related control by a stakeholder community focused on diabetes control use cases. The following two use cases are covered by this guidance:

- Open loop remote control
- artificial pancreas/closed loop control

In general, the guidance herein applies to both use cases unless explicitly clarified.

2.1. Open Loop Use Case

In this use case, one or more mobile applications (apps) running on a CMD are used to perform some command operation, upon request by the CMD user, on a wirelessly connected diabetes device. For example, a diabetes control application may provide a user interface that enables the user to specify the amount of insulin to be dosed by a wirelessly connected insulin pump. The CMD and its diabetes-related apps replace the traditional remote control medical device manufactured by a medical device supplier.

2.2. Closed Loop Control Use Case

In this use case, the CMD is used to host software that performs some portion of a closed loop control system. For example, a continuous glucose monitoring system transmits (via wireless network) sensor readings to a CMD application; the CMD application executes an algorithm to compute treatments of insulin; the CMD autonomously transmits (via wireless network) treatment commands to an insulin pump. The CMD and its diabetes-related apps are executing a continuously repeating algorithm for which each algorithm computation results in a treatment to the patient that must be delivered within some developer-specified time frame in order to maintain safe use.

2.3. Non-Goals

This guidance does not cover standards or guidance already covered in other, pre-existing medical standards and guidance. For example, for the remote control use case, this guidance does not explain how a developer of a remote control solution, which happens to use a CMD and CMD software, follows existing FDA-recommended development standards and obtains FDA approvals to develop and deploy that remote control solution. Rather, this guidance discusses the additional considerations related to the use of CMDs in the context of existing standards and approvals.

3. Definitions

Availability: capability of a system or component to be in a state to execute the function required under given conditions, at a certain time or in a given period, supposing the required external resources are available.

Degradation: strategy for providing safety by design after the occurrence of failures.

Developer: the entity that brings to market a solution to which this guidance applies; while the traditional developer in this sense is a medical device manufacturer, the entity may be some other systems integrator or service provider that is responsible for the safe and secure development and market deployment of the solution.

Failure: termination of the ability of an element to perform a function as required.

PAN: Personal Area Network - the local wireless network used to connect a CMD to one or more medical devices to create an overall medical solution.

Real-time: the actual time during which an activity must take place.

Safety: absence of unreasonable risk.

4. Meeting Security Targets (STs) derived from the CDD PP

This section covers cybersecurity guidance. Cybersecurity requirements for the medical uses defined in this document's scope are covered by the DTSec standard's Protection Profile for Connected Diabetes Devices (CDD PP) and associated Extended Packages (EPs). The specific security requirements for a particular solution (e.g. a standalone product or a system composed of multiple products), whether it incorporates the use of a CMD or not, is defined in an ST, according to the DTSec standard. Such an ST must claim conformance to the CDD PP and one of the EPs: *CDD PP - EP Moderate*, for solutions that require protection against moderate attack potential threats; and *CDD PP - EP Enhanced Basic*, for solutions that require protection against enhanced-basic attack potential threats.

4.1. Guidance for CDD PP - EP Enhanced-Basic

In order to meet the requirements of Enhanced-Basic Attack Potential Assurance, evaluators of solutions that leverage CMD apps ***should*** require the use of CMDs that either are certified against the most recent version of the NIAP Mobile Device Fundamentals Protection Profile (MDFPP) or satisfy the following requirements:

- Hardware-rooted verified boot (provides integrity protection, required by existing CDD PP, but evaluators likely will not need to perform rigorous testing);
- Regular security updates (commitment from CMD manufacturer and previous history of compliance);
- Controls in place to prevent malware-type behavior (for example, ensuring anti-maklware software is embedded within the device and adopting mechanisms to prevent the loading of apps from untrusted sources or from unknown developers.

In addition to these device security attributes, the medical software running on the CMD and the medical software running on a connected device as part of the solution ***should*** perform additional security checks to ensure the medical function is hosted on a CMD that meets the above requirements or at least as much of them that can be attested. Examples of methods for providing this kind of attestation include:

- Ensuring the medical software can only run on known good CMDs (whitelisting via the app store or using mobile device management software).
- Ensuring the CMD software calls operating system attestation APIs to validate that the software is running on known good CMDs.
- Ensuring the connected medical device software uses hardware-backed remote attestation to validate that the CMD software is running on known good CMDs (an example would be the use of Android Oreo's key attestation capability).

While good security often assists in privacy, and while data encryption is recommended for privacy-sensitive medical data, privacy-related requirements are not rigorously considered in the scope of this guidance.

Ultimately, the ability of a solution to meet the requirements of the CDD PP, EPs, or other medical system PPs ***should*** be assessed and determined by an authorized independent laboratory within the appropriate evaluation scheme (e.g. Diabetes Technology Society's DTSec program, DTSec's descendant standard IEEE/UL 2721, etc.) rather than by developers, users, caregivers, or other stakeholders. Developers and regulators ***should*** leverage such standards when determining the safety suitability of CMDs in medical contexts.

4.2. Guidance for CDD PP - EP Moderate

At the time of this writing, meeting the requirements of Moderate Attack Potential Assurance using standard mobile "apps" on CMDs is difficult due to the existence of a frequent stream of exploitable high severity vulnerabilities in various layers of the operating systems managing these apps. Even with the frequent security patching recommended in the preceding section, moderate attack potential attackers have been able to locate exploitable so-called "zero day" vulnerabilities given sufficient resources and effort (applicable to the parameters of moderate attack potential per ISO 18045) dedicated to the task.

Therefore, in order to leverage CMDs for moderate attack potential assurance requirements, the full operating system attack surface area ***should*** be avoided, using one of many possible risk reduction schemes that are technically feasible, albeit not widely deployed on CMDs at time of this writing. For example:

- Host critical functions on a separate security co-processor or other hardware partitioned environment running an independent operating system that is less susceptible to attack due to lower code complexity, lack of attackable surface area (e.g. inability to run arbitrary apps directly on the co-processor), or both.
- Lock down the CMD using a policy enforcement engine (such as that used by enterprises for corporate liable, fully managed operation) to only allow a whitelisted set of highly trusted applications, limit the methods and peers for wireless connections, and employ potentially other controls, thereby making it more difficult for attackers to leverage mobile operating system vulnerabilities.
- Do not depend on the CMD alone for safety and security; for example, a remote control command from the CMD may be double-checked by the user on an insulin pump equipped with its own display.

While the commercial availability of these risk reduction schemes is not widespread at the time of this writing, increased demand for CMDs in medical contexts will help to encourage CMD manufacturers and other service providers to build and leverage such approaches. Any solution approach taken by a developer ***should*** be evaluated by authorized independent testing labs for security and compliance against the CDD PP and CDD PP - EP Moderate.

5. Real-Time Control and Resource Availability

In the use of CMDs for medical control, we are concerned about the ability of CMD medical software operations, - working alone or in combination with one or more medical devices - , to complete reliably and within an expected time-frame, and to obtain access to the required resources to complete their function. For example, when a remote control operation is initiated by the user, does the remote control app running on a CMD (relative to a traditional purpose-built remote controller) successfully transmit the control information wirelessly to the controlled medical device within a human-discernible timeframe? In closed loop control, is a CMD-hosted control algorithm that needs to execute at some fixed periodic interval able to do so without fail (obtaining adequate CPU time), as well as having access to other required resources such as memory, communication, etc.? The ability of a system to complete a required task within some specified deadline is sometimes referred to as real-time, although the computing world often disagrees on the precise meaning of this term. Note that in order to complete a task, access to finite resources other than computing time is also required.

The importance of real-time reliability varies on the application, the ramifications of a missed deadline, and the resilience of the system/function to missed deadlines. For example, if the remote control operation fails to be transmitted to an insulin pump in response to the user's direction (failure of timely access to communication, e.g. radio), the operation may still be safely completed by retrying the transmission or by falling back to manual input on the pump itself. Similarly, a closed loop algorithm that fails to execute within its developer-specified real-time window may cause an alarm on the insulin pump, (driven by the pump itself) , that alerts the user to fall back to manual treatment via the insulin pump. Similar arguments can be made for other forms of failure, such as loss of battery power or loss of wireless connectivity, which may prevent the CMD from completing its operation.

The ability of a medical device to meet its safety requirements is covered by existing medical device manufacturing and regulatory approval processes. For example, a remote controller or dedicated closed loop controller may also lose battery power or connectivity for a variety of reasons, and developers already take such hazards into account in making their safety cases for approval. Therefore, this section covers only additional concerns specific to the use of CMDs in these contexts.

5.1. CMD Real-Time Performance Considerations

CMDs do not run traditional real-time operating systems (RTOS), and therefore some stakeholders may view the use of CMDs in real-time contexts as incurring additional risk. While there may be additional risk, the characteristics of the operating systems themselves arguably contribute less to that risk than the arbitrary workloads that may share compute resources with the medical software. Even traditional RTOS's are rarely able to make mathematically proven response time guarantees under any arbitrary, theoretical workload. Rather, real-time assurance is generated from some combination of proven-in-use (an RTOS has been used for many other real-time projects and is therefore less risky than an operating system that has not

been used in real-time projects), the use of well-understood and well-contained static workloads, the employment of fallback or graceful degradation mechanisms to reduce the impact of missed deadlines, and a heavy dose of empirical testing of the real-time software under a variety of workloads (including intentionally stressful workloads). For example, Linux, an operating system that forms the workload management foundation of Android, has been successfully used in a wide range of real-time systems and has relied more on these other assurance methods than the ability to prove determinism.

Mobile operating systems are subjected to a wide range of workloads across their user populations. Mobile operating system developers go to great lengths to ensure that a single app, either accidentally or maliciously, is unable to dramatically degrade the user experience. For example, both iOS and Android limit the amount of execution resources available to background apps, ensuring that the user's foreground activity remains responsive. It is increasingly difficult for any single app (either accidentally or maliciously) to starve other apps of computing resources. Finally, the response time of the diabetes use cases in the scope of this guidance (usually measured in minutes) are far less stringent than the sub-millisecond response times required in other industrial real-time environments and well within the computing capabilities of modern CMDs.

5.2. Guidance for Open Loop Remote Control

For use case #1, remote control, performance risk is deemed minimal for CMDs. It is advisable that the solution provide some out-of-band (distinct from the primary mobile operating system), assured feedback of the integrity of the remote control operation to the user. For example, the insulin pump may provide audible and/or visual feedback to the user that confirms the remote command, or the CMD may offer an alternative operating environment (e.g. hosted on a co-processor with exclusive display) to provide user confirmation of the command. Such an approach may provide additional security assurance as well.

5.3. Guidance for Closed Loop Control

Use case #2, closed loop control, is a traditional real-time safety-critical application. The inability of CMD software to execute within the solution's required timeframe, without any additional failover mechanism, would render the solution unsafe. If a medical application were to utilize hardware-based secured execution environments, then the integrity of the operating system (and its scheduler) may be reduced as a source of risk as a hazard. The developer ***should*** provide guidance to the user in the form of product documentation that can reduce risk of problems, such as (but not limited to) the avoidance of loading apps from untrusted sources or from unknown developers.

Response time requirements will vary across implementations. For example, one implementation may require an autonomous treatment decision every five minutes. Another may

require a thirty-minute interval. At of this writing, response time windows are not less than a minute and well within the capability of modern CMDs, even under substantial load. However, because the workload of CMDs may vary dramatically from user to user and be subjected to malicious denial of service attack by malware, one or more of the following risk reduction mechanisms are advisable:

- Developer ***should*** stress test and clinically test all supported CMDs and publish to all stakeholders the specific list of CMDs with configurations and operating systems that are deemed safe, even under anomalous load, for closed loop use. Solutions ***should*** not be used on arbitrary, untested mobile devices unless the manufacturer informs the users of the risks of using such systems.
- Developer ***should*** provide guidance to the user in the form of product documentation that can reduce risk of real-time problems, such as (but not limited to) the avoidance of loading apps from untrusted sources or from unknown developers.
- Solutions ***should*** provide a failover mechanism such that missed real-time deadlines will be detected by one or more of the solution's constituent regulated medical devices (e.g. insulin pump) and as a response to such failures, offer a method to exit autonomous operation and perform manual treatment.

6. Availability of the PAN

This section pertains only to closed loop control.

Ambulatory networks provide an increased quality of life to patients but connectivity risks can be translated into patient risk if those networks are not resilient. CMDs in the context of this guidance are expected to be used within a wireless PAN. Interconnectivity of component parts of the PAN and the connectivity of the PAN as a system to other networked entities like cloud services, electronic medical record systems, etc. are achieved by a number of communication~~s~~ transports and modalities. Industry standard radio frequency transports include Bluetooth, Wi-Fi, Zigbee, and others, which exhibit great convenience during normal use but are susceptible to jamming, eavesdropping, and interference immunity threats. Some of these modalities provide some resilience features, such as frequency hopping, automatic reconnection after a service break, localized paired environment, etc. Generally speaking, however, consumer PANs are not currently designed to withstand sophisticated malicious attack of the physical network transport, in contrast to the resilient protocols used in some military wireless networks, such as those developed in the Joint Tactical Radio System (JTRS) program.

PAN denial of service ***should*** be considered in the context of medical use. Examples of failures that the PAN would be resilient to include, but are not limited to:

- Deliberate jamming of PAN radio frequencies;
- Failure of PAN radio transmissions due to hardware component failure;
- Eavesdropping of PAN radio transmissions;
- Radiated immunity threats from adjacent environments.

Sufficient resilience, in the medical context, would be defined as permitting the remediation of situational risk to the patient. At a minimum, the patient ***should*** be alerted and advised if possible when the system detects a risk to safety because of a failure of PAN communications.

6.1. Use Cases for CMDs in PANs

This guidance considers three use cases where a CMD is used to form part of a PAN. Many other use cases can be composed from combinations of these:

1. CMD acts as a “dumb terminal”. It does nothing other than present data to the patient. The CMD does not act upon sensor input nor does it directly control medical operation. The CMD’s disconnection from the PAN or failure to function properly is tolerable for some time, and functionality can be replaced with little or no patient risk using a replacement CMD or backup display built-in to some other component of the PAN. In this case, the CMD is not an essential component of the closed loop control system. Thus, loss of the CMD creates little or no risk to the patient.
2. CMD acts as a “headless” network element, passing through or routing communications, (e.g. from sensors to actuators elsewhere in the PAN) or enabling transmission of data from the PAN to a secure cloud service and back. Medical software is not resident on the CMD, and the CMD’s failure to function properly or disconnection from the PAN can be tolerated for some well-defined time, depending on the clinical environment. In case of such a failure, the medically relevant sensors and/or actuators in the PAN ***should***

failover to an alternate, possibly degraded, mode without incurring significant patient risk. Such a mode may be invoked autonomously or require user intervention. In either case, the solution ***should*** make it clear to the patient that the solution is in a degraded configuration due to loss of the CMD. Full functionality can be regained with re-establishment of the CMD's operation within the PAN.

3. CMD acts as a smart controller through one or more dedicated software applications, and the PAN's sensors and actuators are merely authenticated components, possibly assembled through open procurement and communicating across standards-based interfaces and protocols. This open system may consist of best of breed components selected by the system designer to create a PAN. The CMD and its smart medical software applications are responsible for critical communications and algorithmic control. While failed connection to the cloud may be tolerable, failure of the CMD within the PAN in this use case may be much more difficult to manage safely. In addition, malicious network-borne or malware-borne attacks add more risk to patient safety because the software's medical operation as well as failure detection (e.g. alarms that may alert the patient of a failure) may all be at risk of corruption and denial of service. Due to this increased risk, developers ***should*** avoid relying on the CMD exclusively for safe operation and consider employing redundant safety systems.

Additional recommendations:

1. The developer ***should*** specify the amount of time necessary for the user to avoid, evade, and remediate any denial of service that can create an unacceptable risk to the user, and the PAN-based solution ***should*** remain safe for that specified time period even during denial of service condition.
2. The medically-relevant nodes at each end of an interrupted communication pathway within a PAN ***should*** announce their degraded communication to the user and/or other connected nodesof the PAN such that the user is alerted to a need to take action to remediate a loss of service that poses a risk to patient safety. Remediation may include (but is not limited to) replacing the interrupted communication pathway or seeking help from a service provider.

When considering the preceding use cases, stakeholders ***should*** not assume that any node of a PAN-based solution can be safely replaced with anything other than a node of the exact same manufacture, even if the new node is able to communicate within the PAN. In other words, while the concept of a fully open, interoperable, pluggable, and safe PAN is desirable, stakeholders ***should*** not assume this to be the case unless the safe and secure operation of arbitrary nodes has been evaluated and confirmed by the appropriate community of developers, independent evaluators, regulators, and users. Such a solution does not exist at time of this writing, which is why the DTSec standard currently requires that any composed solution of evaluated and approved nodes must still be re-evaluated, in any deployed configuration, in order to achieve a successful evaluation of the composed solution. The act of evaluating the safety and security of constituent nodes, as well as the use of open interoperable communications protocols, is expected, nevertheless, to dramatically reduce the cost and time of safety and security validation for composed solutions.