Hacking for Fun and Glucose: Reverse Engineering an Insulin Pump

Alex Bellon, Alex Snoeren, Deian Stefan
University of California, San Diego

***Abstract*—Medical devices such as insulin pumps are physically connected to people's bodies and often have direct control over different organs or bodily systems, yet the software controlling these devices often undergoes little to no security review. The FDA (who approves these devices) only generally suggests to check for security issues, but does not provide any specifications for security functionality. This presents a worrying problem, as bugs in such a depended-upon system could have injurious or even fatal consequences. As a result, the correctness and integrity of these systems is incredibly important, yet current systems for ensuring the firmware on these devices is secure often does not catch all the issues. In fact, we have witnessed first hand an insulin pump operating incorrectly due to logic errors in the software. By manually dumping and reverse engineering the firmware from a Tandem Diabetes t:slim X2 insulin pump, we aim to prove that there already exist flaws in the software that could lead to malfunctioning in the best case and a malicious attack in the worse case. We plan to make a framework that will allow developers for these devices to create verifiably secure firmware to prevent future bugs and vulnerabilities.**

I. Introduction

Hundreds of years ago, a diagnosis of diabetes would mean an inevitable premature death. Nowadays, thanks to advances in medical technology, this is no longer the case, with innovations like the insulin pump allowing diabetic individuals to lead relatively normal lives. Some modern insulin pumps can connect to and communicate with smartphones, allowing data about blood sugar levels to be easily accessible to the user. Soon, it will even be possible for users to administer doses of insulin directly from their smartphone without having to interact with the insulin pump at all [1].

While this progress has greatly improved quality of life for diabetic individuals, the functionality and security of these devices have not been keeping pace. Just within recent years, there have been multiple recalls and safety notices for insulin pumps [1] [2] [3] and accessories [4] due to programming errors and security vulnerabilities. In the United States, the FDA (who regulates insulin pumps and other medical devices) provides general security requirements for insulin pumps and related devices, but these requirements do not specify any technical standards or specifications and leave the implementation up to the individual manufacturers.

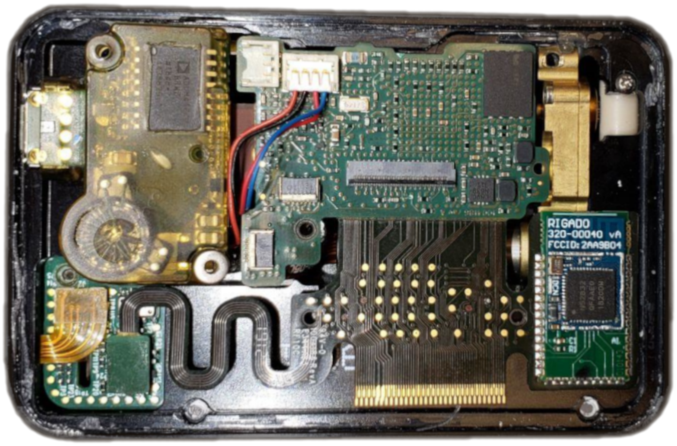


Fig. 1. The insides of a t:Slim X2 insulin pump

Bugs in these critical devices are not just some rare occurrence either: we have witnessed them first-hand, such as an insulin pump continuing to administer insulin even when blood sugar levels were already dangerously low. To more concretely prove the existence of these logic and security flaws, we took apart an insulin pump and extracted the firmware on board so we could examine it.

II. Background

A. Diabetes and Insulin Pumps

Diabetes refers to a group of diseases in which the body cannot regulate blood sugar properly. Normally when you eat food, your blood sugar will begin to rise, and in response your pancreas will release insulin to promote glucose absorption and lower the blood sugar to normal levels. In diabetic individuals, this cycle does

[1] https://www.tandemdiabetes.com/remote-bolus

not work as intended, leading to high blood sugar levels which can cause health complications over time. This lack of control over blood sugar levels can be very dangerous, with continuous high or low blood sugar leading to complications like comas and even death.

Traditionally, a diabetic individual would have to constantly check their blood sugar using a glucose meter (which would require a finger prick), then determine how much insulin they needed to administer. Luckily, medical advancements have largely digitized this process with continual glucose monitors (CGMs) and insulin pumps. CGMs allow continual monitoring of glucose levels through a device that is attached to the individual. Insulin pumps are devices connected to the individual that can automatically administer insulin in adjustable doses.

B. FDA Approval Process

The Food and Drug Administration is responsible for regulating medical devices in the United States and approving relevant devices for market. There FDA categorizes medical devices into 3 classes [5], each with an increasing amount of regulatory control:

- **Class I**: only subject to general controls
- **Class II**: subject to general and special controls
- **Class III**: must obtain premarket approval

Typically, insulin pumps that connect to phones and/or CGMs (called "alternate controller enabled infusion pumps" or more colloquially, "iPumps") are categorized as Class II devices [6]. In the special controls set forth for iPumps, the FDA requires that [6] insulin pumps provide "[s]ecure authentication (pairing) to external device", "[s]ecure, accurate and reliable means of data transmission", and similar requirements. While these requirements are reassuring, the FDA does not define what "secure" entails for device pairing or data transmission, nor does it provide any technical specifications describing, for example, what cryptographic standards to use.

In a news release [7] regarding the Tandem t:Slim X2 insulin pump (the pump which we are studying), the FDA mentioned that while they "...assessed the ability of the pump to communicate with external devices with appropriate reliability, cybersecurity and fail-safe modes", one of risks of using iPumps like the t:Slim X2 "can include incorrect drug delivery as a result of loss of communication between devices, such as the pump misunderstanding commands it receives, or cybersecurity vulnerabilities". Of course, no amount of regulation can completely remove all cybersecurity vulnerabilities from a device, but having more explicit security requirements for iPumps could provide a common starting ground of minimum security.

III. Related Work

There has been a large body of work focused on security for medical devices, beginning with numerous reviews of the status of security attacks and defenses against implantable medical devices (and medical devices in general), as well as possible directions for future work [8] [9] [10] [11] [12] [13] [14].

There have also been works that carry out attacks on medical devices, including pacemakers [15], insulin pumps [16] [17], and other implantable medical devices [18]. In the case of attacks against insulin pumps, to the best of our knowledge these attacks have only leveraged attacks that target the wireless communication protocol between the insulin pump and CGM or remote, and have not analyzed the hardware or firmware running inside of the pump. For other medical devices, there has been work [19] evaluating the security of device firmware, although this was not for an implanted medical device.

Additionally, there have been many proposals for systems to provide confidentiality, integrity and authentication for implantable medical devices while also allowing for possibly insecure but otherwise life-saving safety measures to be performed, striking a balance between security and safety [20] [21] [22] [23] [24] [25] [26]. Some of these works specifically focused on securing insulin pumps [27] [28] [29], although they all took a different approach than the one we intend to pursue (formal verification). Finally, [30] [31] discuss the cost of adding security to medical devices, while [32] offers suggestions for how to implement secure software for medical devices.

IV. The Insulin Pump

The insulin pump used in this work is a t:Slim X2 Insulin Pump manufactured by Tandem Diabetes Care. On its own, it can deliver basal insulin to the user, in addition to boluses controlled through its touch screen.

A. Automatic insulin delivery

When used in conjunction with a CGM, there are two predictive technologies it can additionally provide:

- **Basal-IQ**, which uses glucose level readings from the CGM to prevent low blood sugar by turning off insulin delivery

or, the more advanced

- **Control-IQ**, which uses glucose level readings from the CGM to adjust insulin levels to keep blood sugar levels in range, and predict when to administer a bolus of insulin to prevent the blood sugar from spiking

Only one of the technologies can be used at a time, with Control-IQ acting as an "upgrade" from the less-complex Basal-IQ. Both of these technologies do not require input from the user to confirm the start or stop of insulin delivery (although the user can turn off the technologies entirely if they choose).

B. t:connect

Tandem also provides a smartphone application for iOS and Android, t:connect, which allows users to view information about their pump. They can view the amount of basal insulin being delivered, the time and amounts of the last boluses, as well as their personal settings for target blood sugar levels, amound of insulin to deliver per amount of carbohydrates, and more. When the pump is used in conjunction with a CGM, it will also display a graph of blood sugar levels. Currently the app only *displays* information, but when Mobile Bolus[2] is released, it will also be able to remotely issue commands to the pump to deliver a bolus of insulin.

C. Tandem Device Updater

The t:Slim X2 can be updated using the Tandem Device Updater desktop application (available for Windows and MacOS). Users plug in their insulin pump over USB and begin the process in the Updater, which will authenticate to Tandem's server, download the update files, and upload them to the insulin pump.

V. Getting the Firmware

The firmware for the t:Slim X2 is, unsurprisingly, closed source and not publicly available to random graduate students who want to try to break it. Since the main goal of this project is to get this firmware and look for vulnerabilites, this means that the only way to get the firmware is through more manual means.

A. Intercepting firmware during an update

Our first attempt to retrieve the firmware entailed intercepting the firmware during the update process. Using `mitmproxy`[3] to capture HTTPS traffic and Wireshark[4] to capture USB traffic, we recorded communications between the Updater app, Tandem servers, and insulin pump as the pump went through the firmware update process. Through our HTTPS traffic capture, we were able to intercept 8 `.bin` files that seemed to correspond to different internal parts of the insulin pump:

- `maryann.bin`
- `gilligan.bin`
- `alphamasks_embalmed.bin`
- `raster_embalmed.bin`
- `tndm_ble.bin`
- `ble_stack_embalmed.bin`
- `ble_bl.bin`
- `language_pack_embalmed.bin`

These files were encrypted, so we decided to look for either decryption routines or keys in the firmware that was actually *on* the insulin pump.

B. Extracting firmware from the insulin pump mainboard

After opening the insulin pump and figuring out the basic functions of the different ICs on the mainboard (detailed further in Section VII), we were able to isolate the IC that performed the main processing, the STM32F103ZG chip. Additionally, there was an unconnected ribbon cable protruding from the main board whose contacts were covered over with polyimide tape. After testing the connections between the contacts on this ribbon cable and the JTAG/SWD/SPI pins on the various ICs, it became clear that this was a cable used for debugging that was not disabled or removed when the pumps were shipped out to consumers.

We created and manufactured a PCB that would connect to this ribbon cable and break out each contact to a header pin that could be connected to with a jumper wire. Using this setup (Fig. 2), we were able to connect different debugging devices to read out the firmware of the various ICs. To get the "main" firmware from the STM32 chip, we connected an STLink2 device and read out the flash memory and system memory (Boot ROM). We were also able to extract the firmware from the nRF52832 chip using a J-Link mini.

VI. Reverse Engineering

After retrieving the STM firmware, we began to inspect it in Ghidra[5]. In addition to looking for cryptography-specific functions, we also tried to figure out the general control flow of the program, as well as look for any interesting data or functions.

[2] https://www.tandemdiabetes.com/remote-bolus

[3] https://mitmproxy.org/

[4] https://www.wireshark.org/

[5] https://ghidra-sre.org/

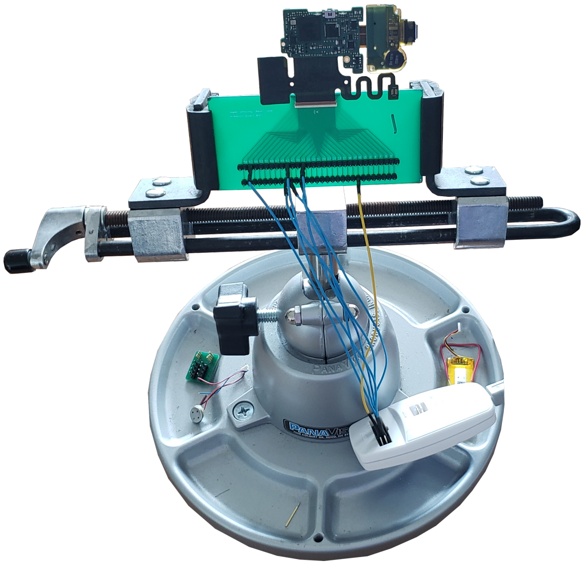


Fig. 2. The STLink2 connected through the breakout board to the insulin pump

A. Cryptographic functions

Using the `findcrypt`[6] plugin for Ghidra, we were able to find AES T-Tables (described in [33] Section 5.2.1) in the firmware memory. Following the references to this data, we were able to find the overarching AES encryption and decryption function, subroutines that performed the `SubBytes` and `ShiftRows` steps of the AES rounds, and functions that handled the key schedule.

Unfortunately, none of these functions we found were referenced by any other functions in the firmware. This led us to believe that these AES functions were part of a library that was included but never used, or that the only way these functions are being called are using function pointers, or some other method that Ghidra would not be able to resolve as references.

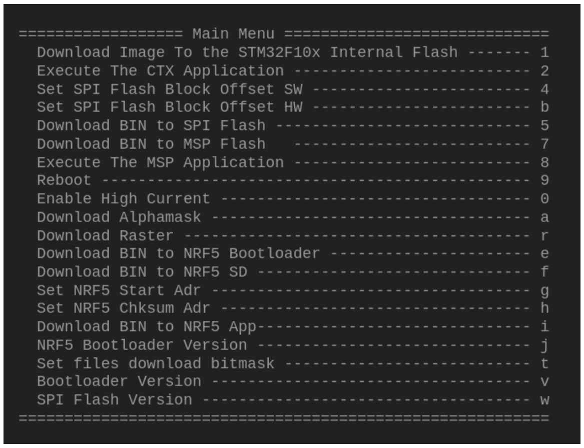
B. Control flow

In Ghidra, through manual inspection, we were able to find the function that is called when the insulin pump is first booted, as well as some ancilliary functions that write binaries to the flash of different ICs and perform CRC checks on the binaries. We matched up the boot function by looking at strings that were printed to the console when the insulin pump was booted while connected over serial to a computer.

C. Interesting data/functions

We found lots of strings that are displayed to the user on the pump during normal operation (regarding blood sugar levels, inslin boluses, etc.), which is what led us to believe that the STM chip is handling the main insulin pump logic.

Additionally, we found a set of strings that seemed to be part of some sort of debug or administrative menu that could write to the flash of different ICs, change IC settings, set addresses, and more. We believe that this was probably used in conjunction with the ribbon cable during development, but we have been unable thus far to figure out how to access this menu when connected to the insulin pump.



```
================== Main Menu =============================
  Download Image To the STM32F10x Internal Flash ------- 1
  Execute The CTX Application -------------------------- 2
  Set SPI Flash Block Offset SW ------------------------ 4
  Set SPI Flash Block Offset HW ------------------------ b
  Download BIN to SPI Flash ---------------------------- 5
  Download BIN to MSP Flash   -------------------------- 7
  Execute The MSP Application -------------------------- 8
  Reboot ----------------------------------------------- 9
  Enable High Current ---------------------------------- 0
  Download Alphamask ----------------------------------- a
  Download Raster -------------------------------------- r
  Download BIN to NRF5 Bootloader ---------------------- e
  Download BIN to NRF5 SD ------------------------------ f
  Set NRF5 Start Adr ----------------------------------- g
  Set NRF5 Chksum Adr ---------------------------------- h
  Download BIN to NRF5 App------------------------------ i
  NRF5 Bootloader Version ------------------------------ j
  Set files download bitmask --------------------------- t
  Bootloader Version ----------------------------------- v
  SPI Flash Version ------------------------------------ w
==========================================================
```

Fig. 3. The debug menu found in the firmware.

VII. Insulin Pump Teardown

The mainboard of the insulin pump has 4 main ICs, a ribbon cable with contacts that connect to different debug pins on the ICs, and many other smaller components. In the below photos, other peripherals, such as the battery, speaker, screen, pump motor, etc. have been removed.

Detailed below are the main ICs on the insulin pump's mainboard, and details about their use, debug interfaces, etc.

A. STM32F103ZG

The STM32F103ZG is a microcontroller with an ARM®Cortex®-M3 CPU, 1MB of flash memory and 96KB SRAM. It provides a JTAG interface (using the TMS, TCK, TDI, TDO and TRST pins) and a SWD interface (using the SWDIO and SWCLK pins). This IC handles the main logic of the insulin pump.

[6] https://github.com/TorgoTorgo/ghidra-findcrypt

B. MSP430F2370

The MSP430F2370 is a microcontroller with a MSP430™ CPU, 32KB of flash memory and 2KB RAM. It provides a JTAG interface (using the TMS, TCK, TDI, TDO and TRST pins). Currently we do not know what this IC is used for, as we were unable to get the TI MSP-FET programmer to connect to the IC.

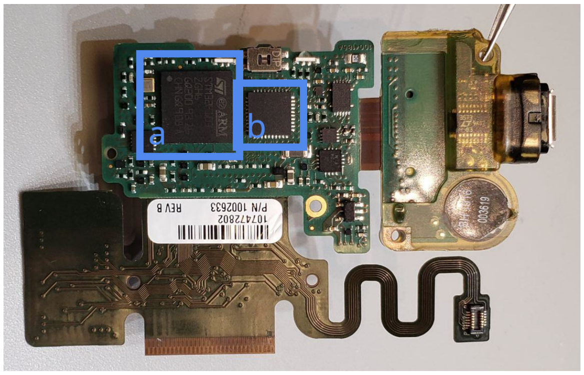


Fig. 4. The back of the mainboard, (a) STM and (b) MSP chips

C. MX25L25645G

The MX25L25645G is a flash memory chip with 256Mb of memory. It provides a SPI interface (using the CS, SO, SI, SCLK). We do not know what specifically is stored on this IC, as the flash chip does not seem to be connected to the debug ribbon cable, and we are otherwise unable to connect to the pins of the chip without removing it entirely and breaking the insulin pump.

D. nRF52832

The nRF52832 is a microcontroller with an ARM®Cortex®-M4 CPU, 512KB of flash memory and 64KB RAM. It provides a SWD interface (using the SWDIO and SWCLK pins). This IC handles Bluetooth connections.

VIII. Limitations

The most glaring limitation is, of course, the fact that this work in incomplete. The intercepted firmware has yet to be decrypted, and there is still the need to automate some sort of vulnerability finding (such as fuzzing).

Speaking specifically of the work that has been done, we present a (non-exhaustive) list of limitations:

- **Firmware incompleteness/inaccuracy**: Since the firmware that is being reverse-engineered was read directly off of the insulin pump using a fairly “hacky” setup, it’s possible that there may have been bit errors when reading the firmware from memory.

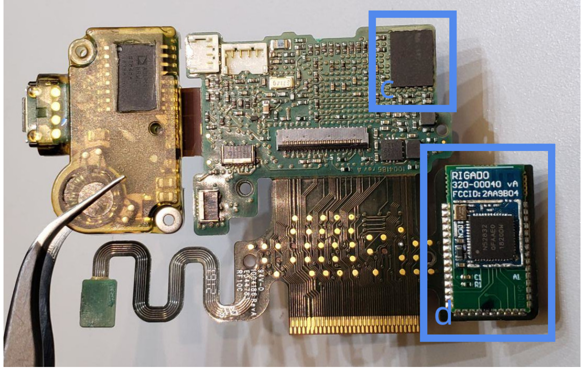


Fig. 5. The front of the mainboard, (c) flash and (d) nRF (Bluetooth) chips

- **Missing firmware**: We have been unable thus far to recover the firmware/data from the flash chip and the MSP chip. It is possible that the cryptographic keys may be stored in the flash chip, and we believe that the MSP chip may be controlling the motors and other mechanical peripherals the insulin pump uses.
- **Static, offline analysis**: All of the reverse engineering that has taken place has been with static, off-device firmware that was read from the insulin pump when it was powered off. Without being able to interact with the firmware as it runs, on device, in real time, there are many address references in the firmware that cannot be resovled.

In summary, since there is no source of truth for how the different parts of the insulin pump mainboard work, which ICs handle which functions, what the firmware looks like, etc. it is difficult to make any sure statements about how the pump does/does not work. Some of this can be remedied by getting more firmware, but some of these issues are just inherent to reverse engineering.

IX. Future Work

We are currently in the process of rehosting the extracted firmware using HALucinator [34]. HALucinator allows users to manually define handlers to be run when the firmware would normally call to a hardware-dependent function (that otherwise could not be replicated off the hardware). This greatly reduces the difficulty of rehosting embedded firmware, and should allow us to run the insulin pump firmware without needing any hardware. Once we are succesfully able to rehost the firmware, we can connect the HALucinator setup to a fuzzer to begin automating the vulnerabilty search.

After searching for vulnerabilities and logic bugs in the firmware, the next major goal for this work is to design and implement a framework that can allow firmware developers working on high-risk systems like health devices to write verifiably secure code.

X. Conclusion

We have presented our progress and findings from the process of reverse engineering the hardware and software of the Tandem Diabetes t:Slim X2 insulin pump. We have recovered firmware for two of the main ICs on the insulin pumps mainboard, as well as encrypted firmware from the firmware update process. We were able to find cryptographic methods in the firmware, as well as the presence of (what seem to be) debug/development menus, although we have not yet been able to get the insulin pump to access them while running. We also presented a breakdown of the main components present on the insulin pump's mainboard and how they are connected to a ribbon cable used for debugging. We plan to continue our work through fuzzing the rehosted insulin pump firmware, and eventually create a framework that allows for the creation of verified firmware.

Acknowledgment

We thank Pat Pannuto, Aaron Schulman, and Nishant Bhaskar for their help with the process of extracting and reverse engineering the firmware from the insulin pump. This work was supported in part by Semiconductor Research Corporation (SRC).

References

[1] U. F. . D. Administration, "Medtronic recalls minimed insulin pumps for incorrect insulin dosing," *FDA.gov*. [Online]. Available: https://www.fda.gov/medical-devices/medical-device-recalls/medtronic-recalls-minimed-insulin-pumps-incorrect-insulin-dosing

[2] Medtronic, "Urgent medical device correction new notification: Basal setting programming," *Medtronic*. [Online]. Available: https://www.medtronicdiabetes.com/customer-support/product-and-service-updates/notice17-letter

[3] B. fuer Arzneimittel und Medizinprodukte, "Urgent field safety notice for t:slim x2 insulin pump by tandem diabetes care, inc," *BfArM.de*. [Online]. Available: https://www.medtronicdiabetes.com/customer-support/product-and-service-updates/notice17-letter

[4] U. F. . D. Administration, "Medtronic recalls remote controllers used with paradigm and 508 minimed insulin pumps for potential cybersecurity risks," *FDA.gov*. [Online]. Available: https://www.fda.gov/medical-devices/medical-device-recalls/medtronic-recalls-remote-controllers-used-paradigm-and-508-minimed-insulin-pumps-potential

[5] *Code of Federal Regulations - Title 21 Chapter I Subchapter H Part 860 Subpart A § 860.3 Definitions*. [Online]. Available: https://www.ecfr.gov/current/title-21/chapter-I/subchapter-H/part-860/subpart-A/section-860.3

[6] *Code of Federal Regulations - Title 21 Chapter I Subchapter H Part 880 Subpart F § 880.5730 Alternate controller enabled infusion pump*. [Online]. Available: https://www.ecfr.gov/current/title-21/chapter-I/subchapter-H/part-880/subpart-F/section-880.5730

[7] U. F. . D. Administration, "Fda authorizes first interoperable insulin pump intended to allow patients to customize treatment through their individual diabetes management devices," *FDA.gov*. [Online]. Available: https://www.fda.gov/news-events/press-announcements/fda-authorizes-first-interoperable-insulin-pump-intended-allow-patients-customize-treatment-through

[8] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and Privacy for Implantable Medical Devices," *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 30–39, Jan. 2008, conference Name: IEEE Pervasive Computing.

[9] M. Rushanan, A. D. Rubin, D. F. Kune, and C. M. Swanson, "SoK: Security and Privacy in Implantable Medical Devices and Body Area Networks," in *2014 IEEE Symposium on Security and Privacy*, 2014.

[10] A. J. Burns, M. E. Johnson, and P. Honeyman, "A brief chronology of medical device security," *Communications of the ACM*, vol. 59, no. 10, pp. 66–72, Sep. 2016. [Online]. Available: https://doi.org/10.1145/2890488

[11] H. Rathore, A. Mohamed, A. Al-Ali, X. Du, and M. Guizani, "A review of security challenges, attacks and resolutions for wireless medical devices," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Jun. 2017, pp. 1495–1501, iSSN: 2376-6506.

[12] J. A. Hansen and N. M. Hansen, "A taxonomy of vulnerabilities in implantable medical devices," in *Proceedings of the second annual workshop on Security and privacy in medical and home-care systems*, ser. SPIMACS '10. New York, NY, USA: Association for Computing Machinery, Oct. 2010, pp. 13–20. [Online]. Available: https://doi.org/10.1145/1866914.1866917

[13] W. Burleson, S. S. Clark, B. Ransford, and K. Fu, "Design challenges for secure implantable medical devices," in *DAC Design Automation Conference 2012*, 2012.

[14] N. Ellouze, M. Allouche, H. B. Ahmed, S. Rekhis, and N. Boudriga, "Security of implantable medical devices: limits, requirements, and proposals," *Security and Communication Networks*, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.939

[15] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. Oakland, CA, USA: IEEE, May 2008, pp. 129–142, iSSN: 1081-6011. [Online]. Available: http://ieeexplore.ieee.org/document/4531149/

[16] C. Li, A. Raghunathan, and N. K. Jha, "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system," in *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*, Jun. 2011, pp. 150–156.

[17] "Hacking Medical Devices for Fun and Insulin: Breaking the Human SCADA System (Black Hat USA 2011) - InfoconDB." [Online]. Available: https://infocondb.org/con/black-hat/black-

hat-usa-2011/hacking-medical-devices-for-fun-and-insulin-breaking-the-human-scada-system

[18] E. Marin, D. Singelée, F. D. Garcia, T. Chothia, R. Willems, and B. Preneel, "On the (in)security of the latest generation implantable cardiac defibrillators and how to secure them," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ser. ACSAC '16. Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2991079.2991094

[19] S. Hanna, R. Rolles, A. Molina-Markham, P. Poosankam, K. Fu, and D. Song, "Take Two Software Updates and See Me in the Morning: The Case for Software Security Evaluations of Medical Devices."

[20] T. Denning, K. Fu, and T. Kohno, "Absence Makes the Heart Grow Fonder: New Directions for Implantable Medical Device Security," p. 7.

[21] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu, "They can hear your heartbeats: non-invasive security for implantable medical devices," in *Proceedings of the ACM SIGCOMM 2011 conference*, ser. SIGCOMM '11. Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2018436.2018438

[22] X. Hei, X. Du, J. Wu, and F. Hu, "Defending Resource Depletion Attacks on Implantable Medical Devices," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, 2010.

[23] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li, "IMDGuard: Securing implantable medical devices with the external wearable guardian," in *2011 Proceedings IEEE INFOCOM*, 2011.

[24] K. Nomikos, A. Papadimitriou, G. Stergiopoulos, D. Koutras, M. Psarakis, and P. Kotzanikolaou, "On a Security-oriented Design Framework for Medical IoT Devices: The Hardware Security Perspective," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020.

[25] K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun, "Proximity-based access control for implantable medical devices," in *Proceedings of the 16th ACM conference on Computer and communications security*, ser. CCS '09. Association for Computing Machinery, 2009. [Online]. Available: https://doi.org/10.1145/1653662.1653712

[26] N. Ellouze, M. Allouche, H. Ben Ahmed, S. Rekhis, and N. Boudriga, "Securing implantable cardiac medical devices: use of radio frequency energy harvesting," in *Proceedings of the 3rd international workshop on Trustworthy embedded devices*, ser. TrustED '13. Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2517300.2517307

[27] E. Marin, D. Singelée, B. Yang, I. Verbauwhede, and B. Preneel, "On the Feasibility of Cryptography for a Wireless Insulin Pump System," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '16. Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2857705.2857746

[28] U. Ahmad, H. Song, A. Bilal, S. Saleem, and A. Ullah, "Securing Insulin Pump System Using Deep Learning and Gesture Recognition," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018.

[29] X. Hei, X. Du, S. Lin, I. Lee, and O. Sokolsky, "Patient Infusion Pattern based Access Control Schemes for Wireless Insulin Pump System," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 3108–3121, Nov. 2015, conference Name: IEEE Transactions on Parallel and Distributed Systems.

[30] K. Fu, "On the technical debt of medical device security," in *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2015 Symposium. National Academies Press*, 2016.

[31] M. A. Siddiqi, A.-A. Tsintzira, and G. Digkas, "Adding Security to Implantable Medical Devices: Can We Afford It?"

[32] T. Haigh and C. Landwehr, "Building Code for Medical Device Software Security."

[33] J. Daemen and V. Rijmen, "Aes submission document on rijndael, version 2," 1999. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[34] A. A. Clements, E. Gustafson, T. Scharnowski, P. Grosen, D. Fritz, C. Kruegel, G. Vigna, S. Bagchi, and M. Payer, "HALucinator: Firmware re-hosting through abstraction layer emulation," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1201–1218. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/clements