Hacks and Attacks: Examples of Electronic Device Compromise

Embedded Systems Conference Silicon Valley 2010 (ESC-343)

Joe Grand[*]
Grand Idea Studio, Inc.

ABSTRACT

Bolstered by the flourishing hobbyist electronics and do-it-yourself movements, easy access to equipment, and nearly realtime information sharing courtesy of the Internet, hardware devices have become a target for both harmless, curious hackers and malicious attackers. Many devices are inherently trusted and taken for granted, though they are actually susceptible to compromise leading to potential financial, social, or legal implications.

As engineers, we have a responsibility to learn from problems of the past and anticipate new ones in order to better equip ourselves for designs of the future. This paper will present a typical hardware hacking process and explore a few real-world attacks against electronic devices.

WHY HARDWARE?

Society thrives on an ever-increasing use of technology. Electronics are embedded into nearly everything we touch. Hardware products are relied on for security-related applications and are inherently trusted, though many are completely susceptible to compromise with simple classes of attacks that have been known for decades.

Contrary to conventional thinking, engineering doesn't only have to be about *design* and hacking doesn't have to be *illegal*. You can combine the best of both worlds - the skills and precision of an engineer with the freewheeling, anti-authoritative mindset of a hacker - to discover, learn about, experiment with, modify, build, break, or improve a product. Whether the goals of a hardware hack or attack are for "good" or for "evil" depends purely on the person or people undertaking the task. Generally, the reasons for "good" hardware hacking can fall into one of the following:

- **Security Competency:** Testing a hardware product or specific hardware security scheme for failures or weaknesses. This is of particular importance to those in charge of recommending security-related products to their company or clients, or to engineers who are trying to verify the strength of their design.

- **Consumer Protection:** Marketing materials and data sheets are typically the only documentation an end user receives when purchasing a product. Hardware hacking may be used to help prove or disprove the vendors' claims of how the product is behaving. Many times, if a discrepancy is discovered, the information is publicly released by the hardware hacker to ensure that all end users are aware of the issue and can take steps to suitably protect themselves.

- **Military Intelligence:** Reverse engineering and forensic analysis of electronic products to help investigators determine how a piece of discovered hardware functions, how it was designed, and by whom.

- **Education and Curiosity:** Tinkering is an important part of the continuing education of engineers. In the process of hardware hacking simply for curiosity's sake, you may learn or discover something that can be added to your "tool box" and used later in another project.

[*] `joe@grandideastudio.com`, `http://www.grandideastudio.com`

Hardware hacking for "evil" could have potentially more nefarious results:

- **Competition (or Cloning):** In this scenario, an attacker (usually in the form of a competitor) would reverse engineer or copy specific intellectual property from the product with a primary goal of gaining an advantage in the marketplace by improving their product using the stolen technology or by selling lower-priced knock-offs.
- **Theft-of-Service:** These attacks aim to obtain a service for free that usually requires payment. Examples include mobile phone cloning, bypassing copy protection schemes, or defeating fare collection systems.
- **User Authentication (or Spoofing):** These attacks are focused on products used to verify the user's identity or store sensitive information intended specifically for the legitimate user, such as a smartcard or authentication token.
- **Privilege Escalation (or Feature Unlocking):** These attacks are aimed at unlocking hidden/undocumented features of a product or to increase the amount of control given to the user without having legitimate credentials (e.g., acquiring administrator access on a network appliance).

ACCESSIBILITY TO HARDWARE HACKING

Easy Access to Tools

Pre-made, entry-level tool packages such as *Ladyada's Electronics Toolkit*[1] or the *Deluxe Make: Electronics Toolkit*[2] provide a fledging hardware hacker with basic equipment, including a soldering iron, soldering and desoldering accessories, and multimeter.

Equipment that had originally cost tens or hundreds of thousands of dollars is now likely to be found used on eBay or surplus markets for a fraction of the price. Digital oscilloscopes, logic analyzers, device programmers, spectrum analyzers, and microscopes used to be out of reach for most hardware hackers except those in academic or government environments. Now, the playing field is leveled, allowing all hardware hackers to take advantage of essentially the same equipment used by production engineering facilities.

Available now are free, open-source schematic capture and PCB design tools, including gEDA[3] and Kicad[4], that allow more hardware hackers to "get in the game" without significant expense. Captive design tools provided by PCB manufacturing houses such as Sunstone Circuits' PCB123[5] or Express PCBs' ExpressSCH and ExpressPCB[6] are some other options for those with limited budgets to design and fabricate basic circuitry.

Easy Access to Information

The Internet has completely changed how information is disseminated and shared. Over the past few years, open source hardware and do-it-yourself sites, such as Hackaday, Instructables, Harkopen, Nuts & Volts, Circuit Cellar, and MAKE, have been providing a constant flow of electronics-related information and projects. Hardware hackers are publishing new work daily, often including pictures, videos, source code, schematic, and Gerber plots, allowing anyone knowledgeable in the field to recreate their work or use their work as a building block for something new.

[1] www.adafruit.com/index.php?main_page=product_info&cPath=8&products_id=136

[2] www.makershed.com/ProductDetails.asp?ProductCode=MKEE2

[3] http://geda.seul.org

[4] www.lis.inpg.fr/realise_au_lis/kicad

[5] www.sunstone.com/PCB123.aspx

[6] www.expresspcb.com

Easy Access to Other People

If you don't have a specific skill required for a particular hardware hack, it's now easy to find someone with whom you can outsource the task or collaborate. Hackerspaces, essentially clubhouses or work spaces set up for sharing equipment and resources, exist all over the world[7]. Public, membership-based commercial organizations like Techshop[8] provide access to tools, classes, and training. Computer security and hacker-related conferences such as DEFCON, Black Hat, ToorCon, Hackers on Planet Earth, ShmooCon, Chaos Communication Congress, Hacking at Random, Hack in the Box, and CanSecWest, are held almost continually throughout the year, offering an opportunity to join forces with other like-minded individuals and learn the latest techniques and mechanisms for designing or defeating systems.

A GENERAL HARDWARE HACKING METHODOLOGY

The beauty of hardware hacking, like much of engineering design, is that there's never only one correct process or solution. My personal hardware hacking methodology consists of the following subsystems:

Information Gathering

Hardware hacking is all about gathering clues from a variety of sources and then using those clues to determine the most likely avenues for attack. An effective solution is to crawl the Internet for specific information related to the target product or the engineers involved in its development:

- Product specifications
- Design documents
- Marketing materials
- Discussion forums and newsgroups
- Personal and corporate blogs
- Social network sites (Facebook, Twitter, LinkedIn)

The traditional, low-tech approach of "dumpster diving," sifting through the target's trash for discarded materials, is still a very promising avenue to obtain useful information.

Social engineering techniques[9] can be used to manipulate a human target into divulging pertinent information. Many hackers will simply call a vendor's sales or technical engineer directly, feigning interest in a particular product, and ask open-ended questions to obtain as much information as possible.

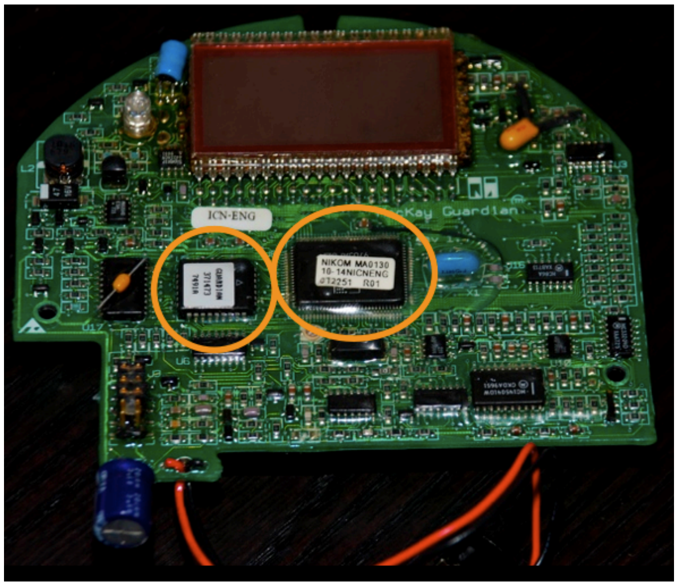
Hardware Teardown

This phase consists of acquiring the target hardware and disassembling it to gather clues about system functionality, design techniques, and potential attack areas. A primary goal of the hardware teardown is to gain access to the circuit board, which will then allow the hacker to identify high-level subsystems, component types and values, and in certain cases, any anti-tamper mechanisms specifically intended to thwart physical attack or tampering. Obviously, having direct access to the circuitry lets the hacker modify, remove, or add components or custom electronics at will.

[7] http://hackerspaces.org

[8] www.techshop.ws

[9] http://en.wikipedia.org/wiki/Social_engineering_(security)



Teardown image from the author's analysis of the San Francisco Smart Parking Meter system[10] showing a meter's main circuit board with the microcontroller and parallel EEPROM highlighted.

External Interface Analysis

Any product interface accessible to the outside world may be an avenue of attack. Programming, debugging, or administrative interfaces are of particular interest, as they may allow the hacker to control the device with the same authority as a legitimate technician or engineer. Common interfaces include smartcard/ISO7816, serial/RS232, USB, JTAG, Ethernet, CAN, and wireless.

This phase of hardware hacking consists of monitoring any communication taking place on an external interface and, if necessary, decoding or emulating the data or protocol. Common engineering tools like a digital oscilloscope or logic analyzer can easily be used to monitor and capture communication. The Bus Pirate[11] is an example of a dedicated, open-source hardware device designed as a universal bus interface specifically to sniff serial protocols, inject data onto the bus, or communicate with new or unknown devices.

Silicon Die Analysis

Using techniques pioneered by integrated circuit designers, publicized by Marcus Kuhn et al[12], and mastered by hackers like Chris Tarnovsky[13] and Karsten Nohl[14], manipulating silicon is no longer black magic. Chip hacking is seen as the next generation of hardware hacking and follows a similar process while being at a microscopic level.

Hackers with access to high-end resources such as chip decapping equipment, probe stations, Scanning Electron Microscopes, and Focused Ion Beam workstations, will be using the same technology that integrated circuit designers use for chip design and debugging. Many hackers are beginning to develop lower-cost, homebrew methods, such as using fuming nitric acid (HNO_3) and acetone to decapsulate and delayer the silicon die, a simple high-powered microscope for die imaging, and a micropositioner with sewing needles to serve as a probe station.

---

[10] www.grandideastudio.com/portfolio/smart-parking-meters/

[11] http://buspirate.com

[12] www.cl.cam.ac.uk/research/security/tamper/

[13] www.flylogic.net

[14] www.cs.virginia.edu/~kn5f/

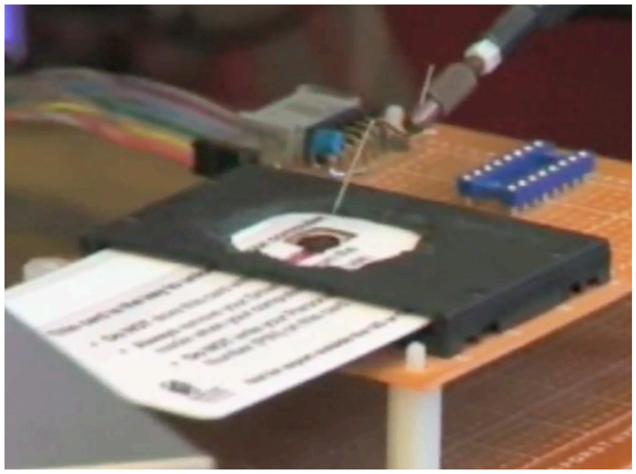
When done right, the decapping process is non-destructive to the silicon die, meaning that the device will still function normally even with the die exposed. Having access to the die can be extremely useful for gathering clues, reverse engineering, or active modifications to further an attack, such as:

- **Imaging:** Take a snapshot of the die at a high magnification to gather general clues about chip manufacturing processes, chip layout, test structures, and possible attack targets.

- **Probing:** Monitor data transfer across a bus to determine system operation, extract secret keys or protected algorithms, retrieve contents of Flash, ROM, or other non-volatile devices, or inject a glitch into the system to cause an intentional failure.

- **Modification:** Modify (cut or repair) gates and other silicon structures, such as adding jumpers to bypass an area of active mesh.

While still a specialized field of hardware hacking, chip hacking has played a pivotal role in helping researchers discover numerous security vulnerabilities and weaknesses in microcontrollers, smartcard implementations, and the highly touted Trusted Platform Module[15].



Probing a smartcard die using a micropositioner and angled sewing needle (from Wired's Hack a Sat-TV Smart Card video[16]).

Firmware Reverse Engineering

This phase consists of extracting program code or data from on-board memory devices (Flash, ROM, RAM, EEPROM) and disassembling the contents. With access to the firmware, a hacker will be able to obtain full insight into product operation and functionality and possibly modify, recompile, and reprogram the code back into the product in order to bypass security mechanisms or inject malicious routines. Disassembly and decompilation tools exist to greatly aid in the reverse engineering process, such as IDA Pro[17], which supports more than fifty families of embedded processors. Many times, a simple pass of the contents through a program like *strings*[18] to display any printable ASCII constants and text strings will yield useful information without the time and effort required for a full firmware analysis.

The prolific adoption of embedded systems has led to a blurring between hardware and software. A software hacker can now use his familiar skills, tools, and techniques in an embedded environment against firmware running on a general-purpose microcontroller without needing to understand hardware-specific constructs.

[15] www.usatoday.com/tech/news/computersecurity/2010-02-08-security-chip-pc-hacked_N.htm

[16] www.wired.com/video/hack-a-sattv-smart-card/1813637610

[17] www.hex-rays.com/idapro/

[18] http://en.wikipedia.org/wiki/Strings_(Unix)

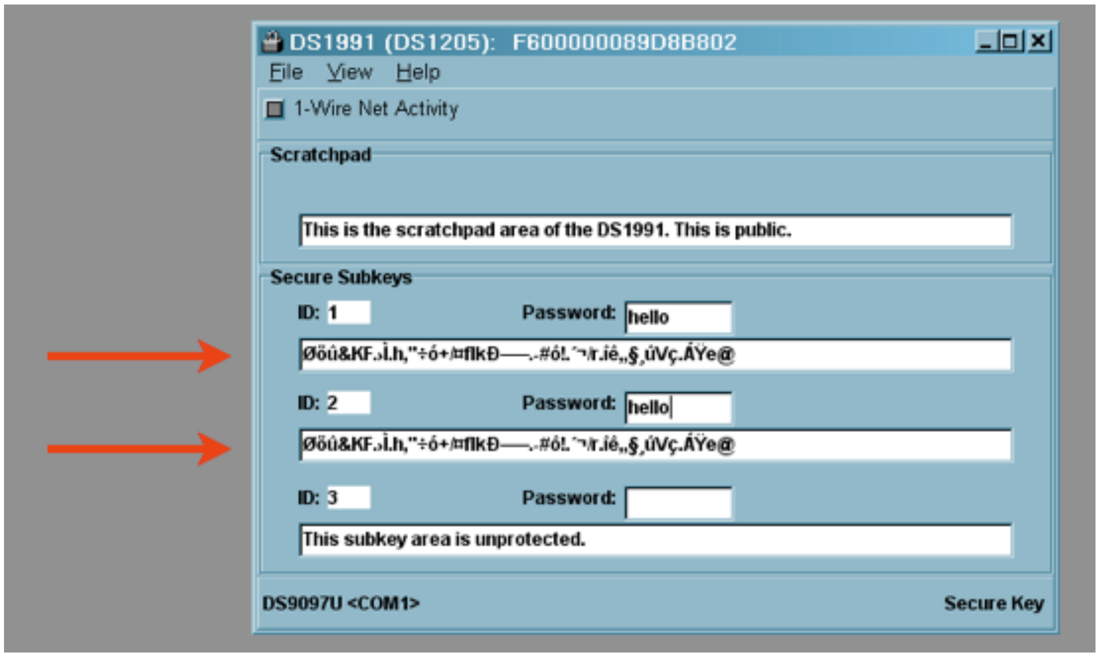
ATTACK EXAMPLES

iButton DS1991

Originally discovered by the author in 2001, this example demonstrates the importance of independently verifying claims made by product marketing literature[19].

iButton devices[20] are hardware tokens deployed globally in applications such as cashless transactions, software copyright protection, user authentication, and access control. The DS1991 contains 1,152 bits of non-volatile memory split into three 384-bit (48-byte) containers known as "subkeys." Each subkey is protected by an independent 8-byte password. Only the correct password will grant access to the data stored within each subkey area and return the 48-bytes of data. As stated in the product marketing literature, if an incorrect password is given, the DS1991 will return 48-bytes of "random" data.

It was determined that the data returned on an incorrect password attempt is not random at all and is calculated based on the input password and a block of data stored within the DS1991 device and constant across all DS1991 devices. By precomputing the 48-byte return value expected for an incorrect password, it is possible to know if a correct password was entered, as the data returned by the DS1991 device will be the actual subkey data, not the precomputed value.



This is the scratchpad area of the DS1991. This is public.

ID: 1 Password: hello

Øöû&KF.›İ.h,"÷ó+/¤flkÐ——.-#ó!.˜¬/r.iê„§¸úVç.ÅŸe@

ID: 2 Password: hello

Øöû&KF.›İ.h,"÷ó+/¤flkÐ——.-#ó!.˜¬/r.iê„§¸úVç.ÅŸe@

ID: 3 Password:

This subkey area is unprotected.

The iButton Viewer application[21] showing that the "random" response for an incorrect password is actually based on the input password. In this case, subkey areas 1 and 2 return the same string for the incorrect password of "hello"

[19] www.grandideastudio.com/portfolio/ds1991-ibutton-dictionary-attack/

[20] www.ibutton.com

[21] www.ibutton.com/software/tmex/index.htm

Microchip PIC Configuration Fuses

This example highlights the "cat and mouse" game played between engineers and hardware hackers as security mechanisms are implemented and then defeated. It is arguably much harder to design a secure mechanism than it is to break it, giving the hacker a significant advantage over engineers, who often have constraints, such as time-to-market and manufacturing cost, to contend with.

Andrew "bunnie" Huang discovered that the Microchip PIC-family of microcontrollers stores its configuration "fuse" settings in Flash memory, which has a floating-gate transistor structure similar to that found in the old UV-erasable EPROM technologies. Of particular interest were the various Code Protection bits that prevent modification or reading of certain regions of memory. His correct hypothesis was that if he could access the die and reset the fuses with UV light, he would effectively disable any asserted Code Protection bits and fully extract any program code stored in the device[22].

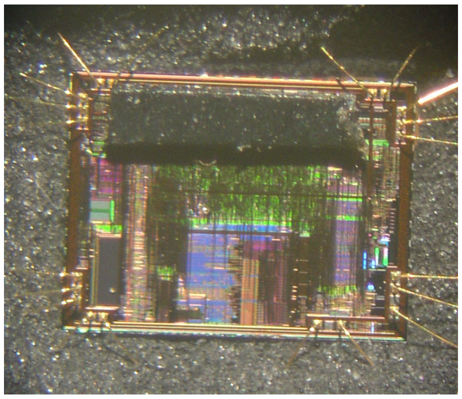
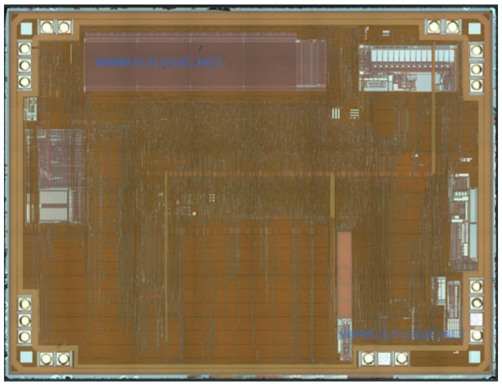
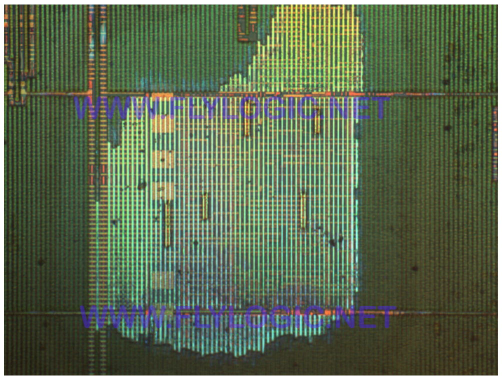


Image of the exposed PIC18F1320 device with electrical tape covering the Flash memory region to prevent the firmware contents from being erased while UV light is shined onto the configuration "fuses."

Likely due to Huang's research, Microchip revised the die and added a metal fill covering almost the entirety of the chip[23]. While this additional layer of security made the attack slightly more time consuming, Chris Tarnovsky was able to selectively remove the metal at the locations above the configuration fuses using a wet-etching technique and reset the Code Protection bits as previously accomplished[24].





Left: Second-generation PIC18F1320 with metal fill patterns placed over open areas
Right: Close-up of the PIC18F1320 with the metal fill removed at the desired location

[22] www.bunniestudios.com/blog/?page_id=40

[23] www.flylogic.net/blog/?p=9

[24] www.flylogic.net/blog/?p=12

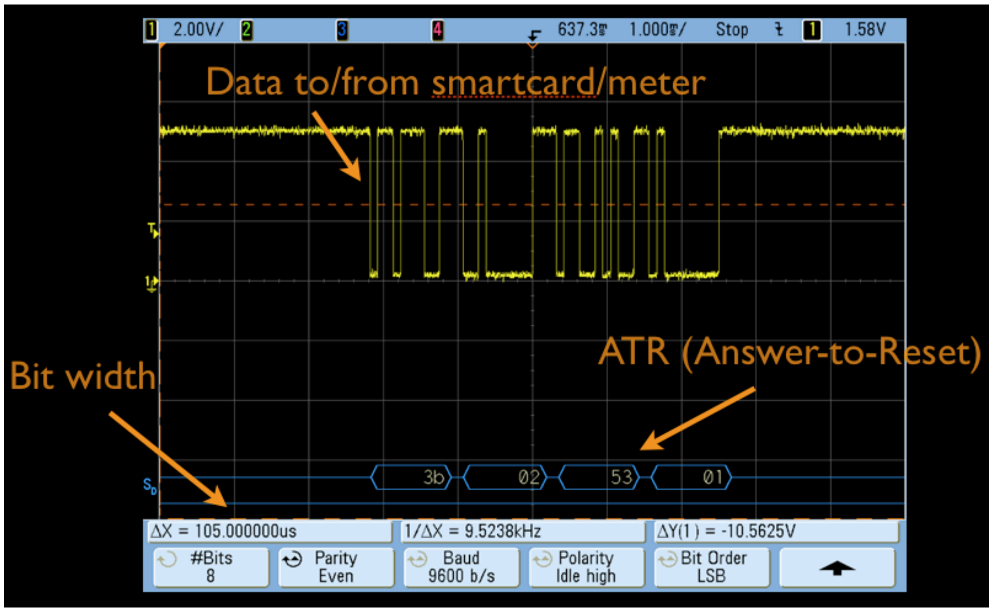
San Francisco Smart Parking Meter

The City of San Francisco contains approximately 23,000 “smart” electronic parking meters manufactured by MacKay Meters[25] that boast tamper resistance, payment via smartcard, and usage auditing capabilities. The author, in collaboration with Jacob Appelbaum and Chris Tarnovsky, evaluated San Francisco’s electronic parking meter implementation, which was installed at a cost of $35 million, and successfully compromised the meter via its smartcard interface[26].

The research process encompassed many of the hardware hacking steps described in this paper and the results serve as a prime example of why electronic devices used in security or financial applications cannot and should not be fully trusted without thorough analysis and testing.

One feature of San Francisco’s implementation is supporting the use of a stored value smartcard. These non-refillable, disposable cards can be purchased online or at selected locations within San Francisco in $20 and $50 values. When the card is inserted into the meter, the meter first calculates the remaining value on the card and then, every few seconds, deducts a “unit,” corresponding to $0.25. When the meter displays the desired amount of time, the user removes the smartcard.

While there were many avenues available for attacking the smart parking meter, effort was focused on the easily accessible, external smartcard interface. An unpopulated “shim”[27] placed between the smartcard and parking meter was used to break out the requisite signals and the communication was then captured using an Agilent DSO7054A digital storage oscilloscope. After determining the communication settings (half-duplex, asynchronous serial protocol at 9600bps, 8 bit, even parity, 1 stop bit), the serial decoding function of the DSO was used to display the actual data bytes being transmitted from the meter to the card and received by the meter from the card.



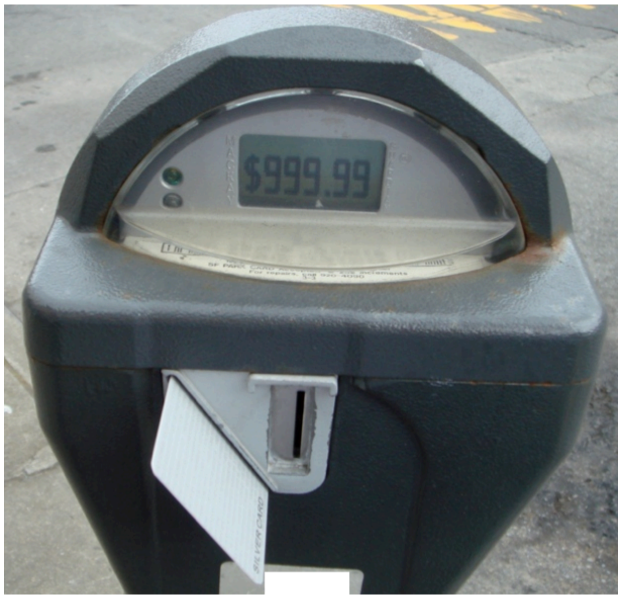
Oscilloscope screenshot showing a captured data stream and its corresponding data bytes. In this case, the smartcard is responding to the meter with an Answer-to-Reset, the first bytes sent by an ISO-7816 compliant smartcard after coming out of reset.

[25] www.mackaymeters.com

[26] www.grandideastudio.com/portfolio/smart-parking-meters/

[27] http://interesting-devices.com/asp/product.asp?product=160

The entire conversation of a card communicating with the meter to deposit "coins" was analyzed offline and, through continued trial and error with multiple cards holding varying remaining values, was successfully reverse engineered in a matter of days. Using a custom designed Microchip PIC16F877-based smartcard emulator, it was possible to replay the exact transaction of a legitimate card, as well as modify specific data contents within the transaction to fool the meter into believing the card held a value well above the standard $20 and $50 denominations sold by the City of San Francisco.



Results of the author's analysis of the San Francisco Smart Parking Meter system.

CONCLUSION

With easy access to tools, manufacturing, information, and other people with common interests, hardware hacking is now more accessible to hackers than ever before. The goal of this paper was to introduce the reader to the hardware hacking process and demonstrate, through real-world examples, attacks against electronic devices covering a wide range of methodologies and levels of technicality. It is crucial to learn from history and understand the details of prior hardware attacks in order to better design products for the future.