Breaking Access Controls with BLEKey

by Mark Baseggio and Eric Evenchick

Abstract

RFID based access controls are ubiquitous in the enterprise today. Across manufacturing facilities, industrial sites and offices of all types if you look towards a door, chances are, in the vicinity you'll see a familiar grey or beige plastic box with a glowing LED. On the surface these systems feel secure but in reality most are not.

In fact, the most popular access control system in use today (HID Prox™) has absolutely no security. The cards, communication between the card and reader, and finally the transmission of data upstream are all unprotected by encryption. While recent generation HID hardware includes additional security features, one common thread has remained: unencrypted data transmission to the backend.

This paper will focus on the upstream communication protocol used by RFID readers, called Wiegand. We'll discuss the potential for and practical abuse of this protocol using a piece of hardware we developed called the BLEKey.

BLEKey is a tiny device that is designed to be embedded into an RFID reader or the wall behind one, and attached to the lines carrying Wiegand data. Once attached the BLEKey monitors the lines for data and stores any cards it reads. Since the device is directly attached to the Wiegand lines, it's also possible to control them, enabling replay attacks and writing of arbitrary data. Control of the BLEKey is achieved wirelessly using Bluetooth Low Energy (BLE), or via specially programmed RFID cards.

The intention of BLEKey is to provide an easy to use device designed to exploit the unencrypted nature of Wiegand. A penetration tester can use this device to illustrate the triviality of abusing the Wiegand protocol to defeat access controls, with the ultimate goal exposing the severity of the issue to card users, encouraging manufacturers to adopt a secure alternative.

Introduction

Some may be surprised to discover that the protocol used to send data from an RFID reader to the backend door controller started life in the early 1970s [1]. John R Wiegand discovered a special wire that would eventually be embedded in a plastic card, which when pulled through a reader, creates a bitstream of ones and zeros based on the presence or lack of wire in the card [2]. The protocol used to transmit this information in early access control devices became known as the Wiegand protocol. Today, the Wiegand protocol remains the predominant method by which RFID access control card readers communicate with upstream devices [3].

Wiegand uses three data lines to transmit binary data to an upstream device (e.g. a door network controller), which handles the authentication and door control function in an access control system. The protocol is relatively simple, using one line to indicate a 0 (DATA0), another for 1 (DATA1) and the remaining line for ground. The data lines are typically coloured green to indicate the 0 line and white for the 1 line, making them easy to distinguish and locate. Data transmission on the lines is similarly simple, with each line normally being held high (at 5v), a single bit on either line is indicated by pulling the appropriate line to ground; see figure 1 for a graphic depiction of two bits being transmitted using the Wiegand protocol.

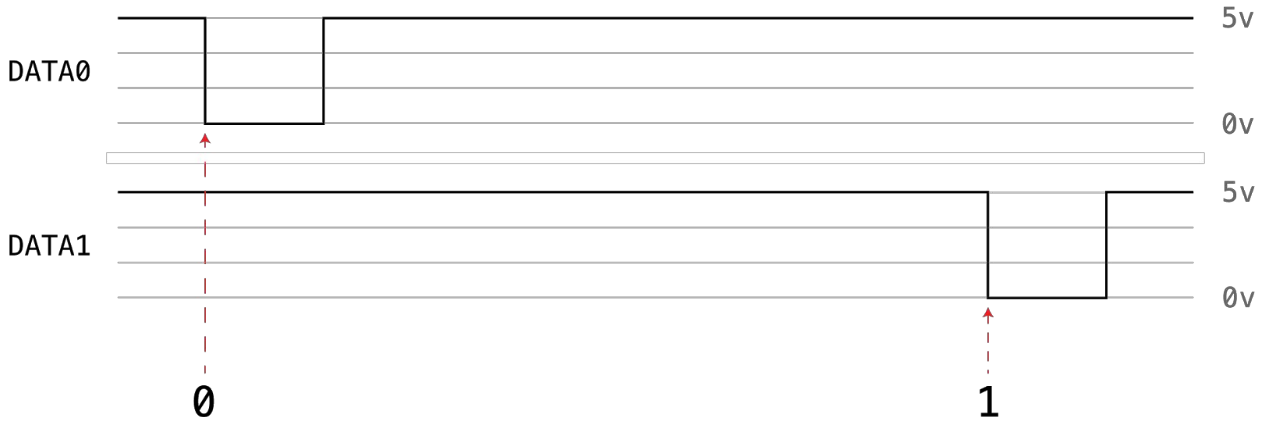


Figure 1: The transmission of two bits (0, 1) using the Wiegand protocol captured using a logic analyzer.

Thanks to the Wiegand protocol's simplicity, and the proliferation of affordable electronics prototyping platforms such as the Arduino, today the protocol is remarkably simple to capture and decode. A quick Google search for "wiegand arduino" returns a wealth of resources including a complete Wiegand protocol library for the Arduino [4], as well as example code from PageMac [5] which the authors used to understand the Wiegand protocol.

The initial motivation for building BLEKey came after one of its creators built another RFID security tool, a long range RFID thief (largely influenced by the *Tastic RFID Thief* [6]). The long range thief can capture card data from a distance of approximately 3 feet, it works by recording Wiegand data from an RFID reader designed for parking garage applications. While this attack is very effective, it requires a different RFID reader for each unique card technology--and it isn't always practical to bring along three or four different hardware configurations while conducting a physical penetration test. While it's possible to circumvent protections on newer encrypted card technologies such as HID's iClass™ [7] it's much easier to read and store Wiegand data, like the Tastic RFID thief does. With this in mind, it was clear that taking advantage of the Wiegand output on readers already installed on a wall somewhere, protected by only a few screws, could yield convincing results.

The idea for the BLEKey was born. A few key parameters were laid out for the device during the planning phase: it needed to be small, easy to install (no soldering or cutting wires), must have the ability to read and write the Weigand protocol and to be controlled wirelessly. A device with these characteristics could be used in several attack scenarios; it could be surreptitiously installed in an RFID reader during a physical penetration test, or used inside the assessor’s own RFID reader to create a Bluetooth Low Energy enabled portable RFID skimmer. Either attack would yield the desired result, impressing upon a client not only that their RFID systems are insecure, but that the attack is very real and relatively easy to accomplish.

Background

After some research it was discovered that a device called the Gecko was presented at Blackhat USA in 2008 by Zac Franken. The Gecko device also connects to Wiegand lines in order to collect card data and perform replay attacks–however lacking wireless, it used command cards for control functions [8]. It is unclear if the device was ever released as open source hardware, or made available through other means. Gecko however did not go unnoticed, being mentioned in marketing materials published by 3M [3] and Borer [9], US patent 8358783 B2 filed in 2009 [10], and even in a DoD paper [11].

Despite the previously mentioned attack vectors against the Wiegand protocol, it seems that its use in the enterprise hasn’t significantly changed since Franken’s release of Gecko. Through this author's own experience during physical penetration tests, the vast majority of systems encountered across North America, Europe and Asia were using the protocol. Moreover, modern readers such as HID’s iClass and iClass SE lines continue to support the insecure Wiegand protocol–negating the use of the systems’ encrypted RFID cards. To make matters worse, legacy HID Prox compatible devices are still being sold for new installations; to put this in terms of the personal computer, if software lasted this long we’d all still be using DOS 1.0.

It seems that either organizations are unaware of the inherent security risk of using Wiegand, or that they have simply chosen to accept the risk. Based on anecdotal evidence gathered during the course of physical testing, the authors believe that most vulnerable organizations are under the impression that their access control system are secure. The BLEKey is designed to be an affordable turnkey device intended to provide these businesses a real world example of the potential implications of using insecure access controls. The choice to release both the hardware and firmware as open source helps to make the device accessible, and allows security professionals to build upon its current functionality to include new features and support for other systems that utilize the Wiegand protocol.

BLEKey Hardware and Firmware

The BLEKey design and layout was dpone using the open source EDA tool KiCad (http://www.kicad-pcb.org/). It uses a pre-built Bluetooth Low-Energy module from Raytac, which uses the Nordic nRF51822 SoC (nRF51). The nRF51822 soC has 2.4GHz radio, with a 32-bit ARM Cortex M0 CPU core along with 256kB of flash memory and 16kB of RAM.

Nordic Semiconductor provides an SDK that is largely compatible with ARM GCC, however some code examples are only easily usable in the Keil uVision IDE. BLEKey's firmware was written for and compiled with the ARM GCC compiler, which is available for Microsoft Windows, Linux and Mac [13]–with the exception of its bootloader, which due to compatibility issues [14] was built using Keil uVision.

The BLEKey code can be programmed on to the nRF51 via P1 on the BLEKey with a Serial Wire Debug (SWD) adapter such as the Segger J-Link, or by doing a Firmware Over the Air (FOTA) flash. In order to provide an easy and accessible method to update the firmware on BLEKey, a FOTA compatible bootloader is pre-programmed on the device. This allows the BLEKey to be updated with Nordic's Firmware update applications, available for most major mobile platforms [15]. When new features are added to the firmware it can be updated by simply downloading or compiling a firmware file from source, shorting two pins on the device, and loading the new firmware with a mobile phone.

In 2008 the Gecko's stated cost was approximately $10USD [8], today the BLEKey can be produced and assembled in small quantities at the same cost. While keeping the cost to $10, the BLEKey design removes barriers to use found in the Gecko–primarily thanks to the rapid advancement of technology. BLEKey's design eliminates the need to physically cut wires, introducing an insulation displacement connector which allows for a quick and relatively easy installation. Thanks to its wireless technology, the use of command cards is no longer necessary–data can now be extracted wirelessly, rather than having to remove the device and risk being caught during penetration testing.

For a complete bill of materials and schematic of the BLEKey version 1.1 refer to Appendix A.

Installation

In order to operate, the BLEKey must be physically attached to the Wiegand transmission lines of an RFID reader. Most RFID readers are attached to a mounting surface first and then have a cover attached to them using between 1-4 Phillips screws. Upon removing a reader's plastic cover, its connection terminals are typically well labelled; or in the case of a "pigtail" type wire connection from the rear of the reader, the Wiegand lines can be located by their colours: green (DATA0), white (DATA1) and black (GND).

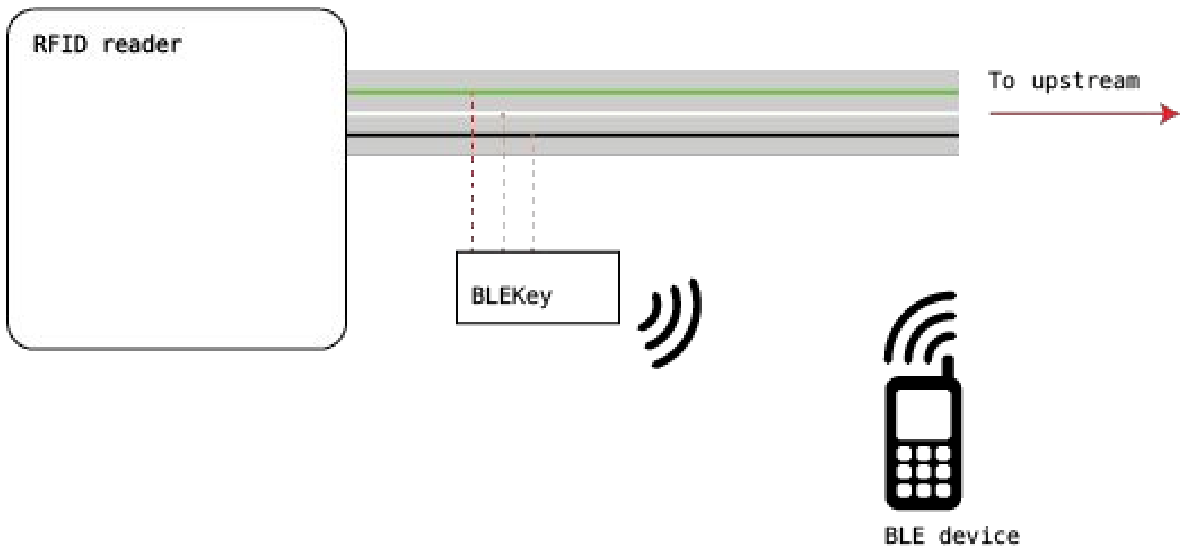


Figure 2: A diagram illustrating the typical BLEKey deployment.

The BLEKey can be connected to the Wiegand lines of a reader in several ways, first via the insulation displacement connector (IDC) located at the bottom of the board (below the GRN, WHT, GND screen printing on the board in the figure below). The IDC connector requires that each wire be placed in the connector and pushed in using a “punchdown” tool to make contact. The versatile P2 connector with standard 0.1” pin headers can also be used in the case that an alternate method of connection is required.

The BLEKey requires a single CR1632 lithium coin-cell type battery for operation, which is inserted on the back-side of the device. The battery holder is pictured in figure 3 below, on the right hand side of the image.

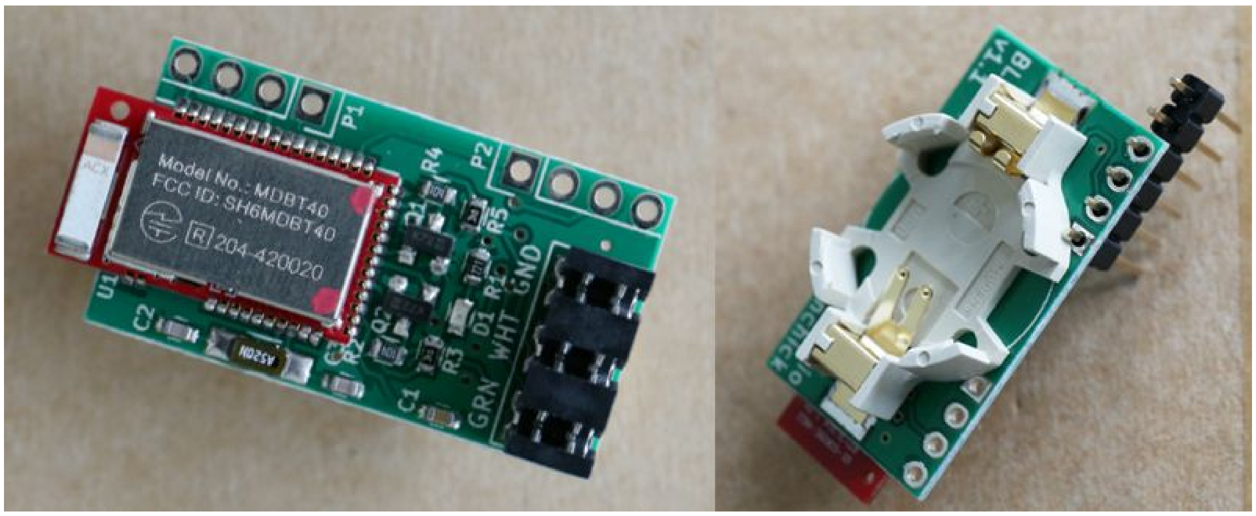


Figure 3: BLEKey v1.1 sample images provided by the production facility, left: front, right: back.

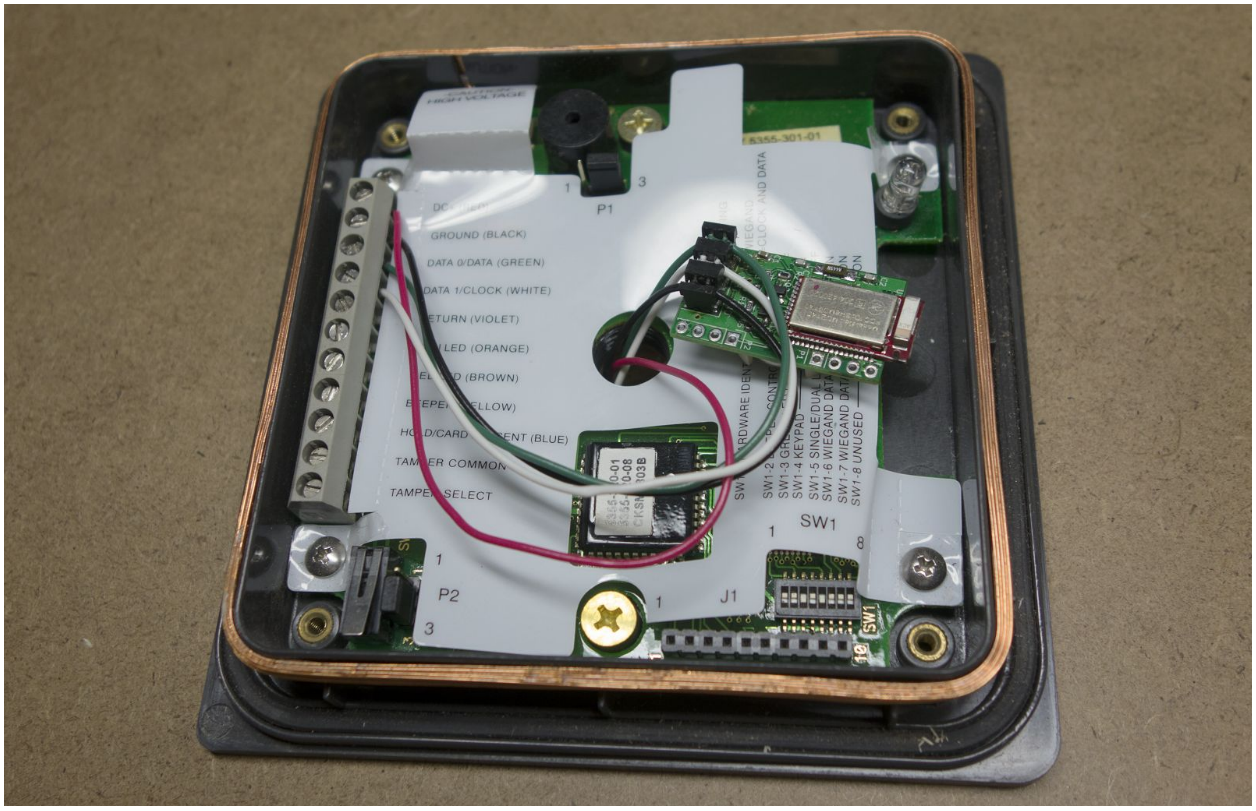
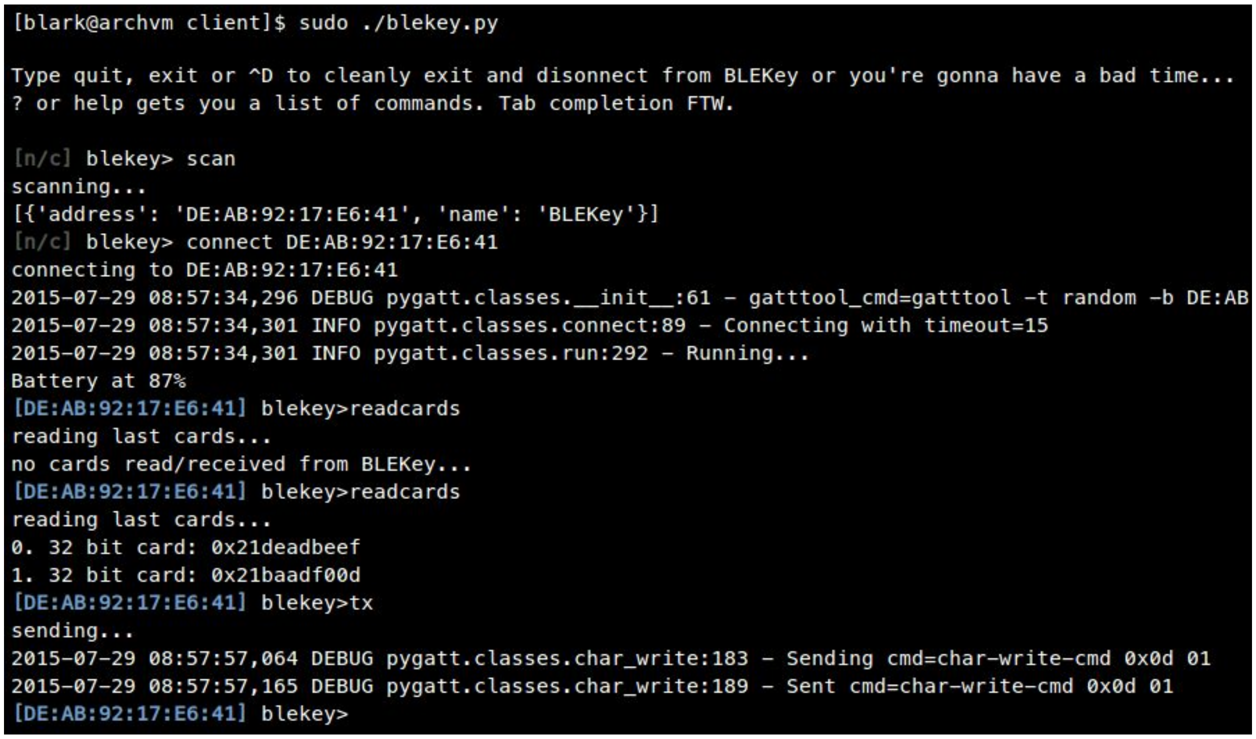


Figure 4: BLEKey installed in a popular RFID reader.

Using BLEKey

The firmware released at Blackhat USA 2015 is designed to work with HID Prox compatible systems that use 26-40 bit formatted cards. Once the BLEKey is installed and powered on it will begin to record card data. A Python utility is included in the BLEKet git repository that is capable of reading data from and controlling the BLEKey.

In figure 5 below the BLEKey client is *scan* command used to scan for BLE devices, then connect to a BLEKey. Once connected the client returns the device's current battery status and awaits a command from the user. The *readcards* command is issued reading back 0 cards in the first instance because the BLEKey has received no card data, after two cards are read by the BLEKey the user again issues the *readcards* command which returns the two 32 bit cards read (0x21DEADBEEF and 0x21BAADF00D). Finally, the *tx* command is issued, which by default replays the last card read by the BLEKey. The client implements rudimentary help, most of its functions should be fairly self-explanatory.

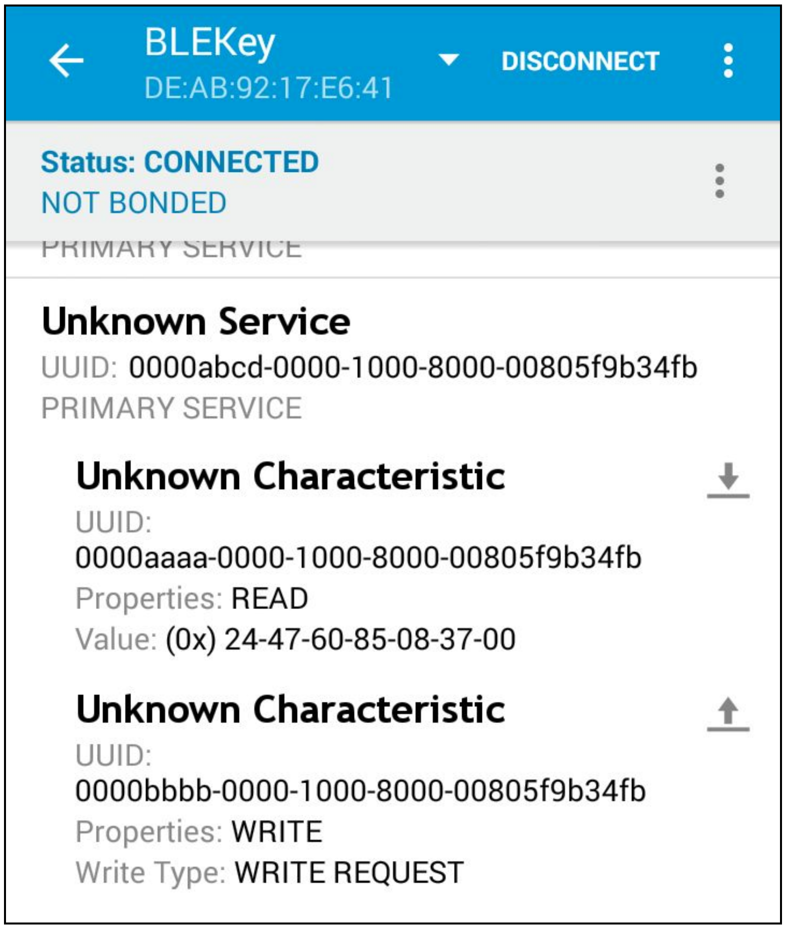


```
[blark@archvm client]$ sudo ./blekey.py

Type quit, exit or ^D to cleanly exit and disonnect from BLEKey or you're gonna have a bad time...
? or help gets you a list of commands. Tab completion FTW.

[n/c] blekey> scan
scanning...
[{'address': 'DE:AB:92:17:E6:41', 'name': 'BLEKey'}]
[n/c] blekey> connect DE:AB:92:17:E6:41
connecting to DE:AB:92:17:E6:41
2015-07-29 08:57:34,296 DEBUG pygatt.classes.__init__:61 - gatttool_cmd=gatttool -t random -b DE:AB
2015-07-29 08:57:34,301 INFO pygatt.classes.connect:89 - Connecting with timeout=15
2015-07-29 08:57:34,301 INFO pygatt.classes.run:292 - Running...
Battery at 87%
[DE:AB:92:17:E6:41] blekey>readcards
reading last cards...
no cards read/received from BLEKey...
[DE:AB:92:17:E6:41] blekey>readcards
reading last cards...
0. 32 bit card: 0x21deadbeef
1. 32 bit card: 0x21baadf00d
[DE:AB:92:17:E6:41] blekey>tx
sending...
2015-07-29 08:57:57,064 DEBUG pygatt.classes.char_write:183 - Sending cmd=char-write-cmd 0x0d 01
2015-07-29 08:57:57,165 DEBUG pygatt.classes.char_write:189 - Sent cmd=char-write-cmd 0x0d 01
[DE:AB:92:17:E6:41] blekey>
```

Figure 5: the BLEKey client example interaction with a target device.

The BLEKey client utility runs exclusively on Linux and requires version 1.1.0 of the pygatt [16] module and corresponding Linux bluetooth tools.

Alternately, it is possible to read data from the device using one of the many available tools designed to work with BLE devices. The information required to use the BLEKey with a custom tool or generic BLE application is available in Table 1.



BLEKey
DE:AB:92:17:E6:41
DISCONNECT

Status: CONNECTED
NOT BONDED

PRIMARY SERVICE

Unknown Service
UUID: 0000abcd-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE

Unknown Characteristic
UUID:
0000aaaa-0000-1000-8000-00805f9b34fb
Properties: READ
Value: (0x) 24-47-60-85-08-37-00

Unknown Characteristic
UUID:
0000bbbb-0000-1000-8000-00805f9b34fb
Properties: WRITE
Write Type: WRITE REQUEST

Figure 6: Nordic's Master Control panel application (Android) connected to a BLEKey.

Description	**UUID**	**Purpose**
Cards read	0000aaaa-0000-1000-8000-00805f9b34fb	Reading this characteristic will return the last 73 cards read by the BLEKey (unless page is configured). The field will return up to 511 bits of non-delimited data, each card is 7 bytes and is 0xLLCCCCCCCCCCCC where L is the bitlength of the card and C is the card data in Proxmark3 format
Write Wiegand	0000bbbb-0000-1000-8000-00805f9b34fb	Write a number to this characteristic to replay that card. A value of 255 will cause the BLEKey to replay the last card, while a value of 254 will write arbitrary data supplied by the user.
Custom data	0000cccc-0000-1000-8000-00805f9b34fb	7 bytes of data can be supplied to the BLEKey in the format 0xLLCCCCCCCCCCCC where L is the bitlength of the card and C is the card

		data in Proxmark3 format.
Page	to be implemented	Configure the card page to be read from flash memory
Cards Read	to be implemented	Returns the number of unique cards read and stored by the BLEKey

Table 1: BLE characteristics implemented and planned for the BLEKey.

BLEKey v1.1 outputs a 3.3v TTL serial debug on the P2 port’s first pin (with the square solder mask). The serial debug can be used to monitor the BLEKey’s operation, and for troubleshooting while adding new features.

Firmware Updates

The firmware (application) on the BLEKey will be outdated before Blackhat 2015 due to ongoing work on the application. The latest firmware for the device will be available on the github repo under the *firmware* folder, entitled *application.bin*. In order to perform an upgrade simply take a paperclip and short the first and last pins of P2 while powering on the device. It will enter the bootloader and advertise as “DfuTarg,” at this point the NRF Master Control Panel application can be used to perform a software update. For more details see the readme.md available in the root directory of the project’s git repository.

Conclusion

The BLEKey was created and released as an open source tool in effort to illustrate the potential for abuse of the Wiegand protocol. We believe the device provides an effective mechanism for physical penetration testers to easily and impactfully illustrate practical attacks to their clients. We hope that the device and media coverage surrounding its release at Blackhat USA 2015, helps to end the use of Wiegand as an upstream protocol.

Appendix A

BLEKey v1.1 Schematic

BLEKey v1.1 Bill of Materials (BOM)

#	Designators	Mfr.	Part Number	Supplier P/N	Description	Qty.
1	K1	AVX	9176003022006	478-4619-1-ND	IDC Connector	1
2	U1	Raytac	MDBT40	MDBT40	Bluetooth Module (nRF51822)	1
3	Q1, Q2	Diodes Inc	2N7002-7-F	2N7002-FDICT-ND	FET	2
4	R2, R4	Panasonic	ERJ-3GEYJ104V	P100KGCT-ND	Res 100k	2
5	R3, R5	Panasonic	ERJ-3EKF9312V	P93.1KHCT-ND	Res 93.1k	2
6	R1	Panasonic	ERJ-3GEYJ221V	P220GCT-ND	Res 220	1
7	C1	Murata	GRM188R71C104KA01D	490-1532-1-ND	Cap 0.1 uF	1
8	C2, C3	Murata	GRM1885C1H120JA01D	490-1405-1-ND	Cap 12 pF	1
9	X1	Epson	FC-135	-	32.768 KHz Xtal	1
10	BATT1	MPD	BU1632SM-JJ-GTR	BU1632SM-JJ-GCT-ND	Battery Holder	1
11	D1	Lite-on	LTST-C191KGKT	160-1446-1-ND	LED, Green	1
12	P1, P2	-	-	-	0.1" pin header (1x4)	2

Reference Documents

[1]	"Patent US3820090 - Bistable magnetic device - Google ..." 2011. 21 Jul. 2015 <http://www.google.com/patents/US3820090>
[2]	"Brushing Up on Wiegand: The man, the effect, and the wire ..." 2014. 21 Jul. 2015 <http://machinedesign.com/engineering-essentials/brushing-wiegand-man-effect-and-wire-changed-engineering>
[3]	"Access Control in the 21st Century - 3M." 2014. 21 Jul. 2015 <http://multimedia.3m.com/mws/media/833804O/beyond-wiegand-access-control-in-the-21st-century.pdf>
[4]	"monkeyboard/Wiegand-Protocol-Library-for-Arduino · GitHub." 2013. 22 Jul. 2015 <https://github.com/monkeyboard/Wiegand-Protocol-Library-for-Arduino>
[5]	"Arduino Wiegand Decoder for HID RFID Reader - PageMac." 2012. 22 Jul. 2015 <http://www.pagemac.com/azure/arduino_wiegand.php>
[6]	"Tastic RFID Thief - Bishop Fox." 2013. 22 Jul. 2015 <http://www.bishopfox.com/resources/tools/rfid-hacking/attack-tools/>
[7]	Meriac, M. "Heart of Darkness - exploring the uncharted backwaters of ..." 2010. <http://www.openpcd.org/images/HID-iCLASS-security.pdf>
[8]	"Zac Franken BlackHat DC 2008." 2008. 23 Jul. 2015 <https://www.blackhat.com/presentations/bh-dc-08/Franken/Presentation/bh-dc-08-franken.pdf>
[9]	"Wiegand's Had Its Day! White Paper - Borer Data Systems Ltd." 2008. 23 Jul. 2015 <http://borer.co.uk/pages/case_studies/wiegand_white_paper_oct_2007.pdf>
[10]	"Patent US8358783 - Secure wiegand communications ..." 2013. 23 Jul. 2015 <http://www.google.com/patents/US8358783>
[11]	"Review of the Open Supervised Device Protocol (OSDP™)." 2015. 23 Jul. 2015 <http://www.acq.osd.mil/ncbdp/nm/pseag/news-references/references/SEIWG_OSDP%20Review_Public%20Release_20140801_v1.0.pdf>
[12]	"OSDP_V2 1_5_2014 - Security Industry Association." 2014. 23 Jul. 2015 <http://www.siaonline.org/SiteAssets/SIAStore/Standards/OSDP_V2%201_5_2014.pdf>
[13]	"GCC ARM Embedded - Launchpad." 2011. 23 Jul. 2015 <https://launchpad.net/gcc-arm-embedded/+download>
[14]	"nrf51822 DFU Bootloader w/ gcc and SDK 6.0 - Nordic ..." 2014. 23 Jul. 2015 <https://devzone.nordicsemi.com/question/14463/nrf51822-dfu-bootloader-w-gcc-and-sdk-60/>
[15]	"nRFready Demo Apps - Nordic Semiconductor." 2015. 23 Jul. 2015 <https://www.nordicsemi.com/eng/Products/nRFready-Demo-Apps>
[16]	Python wrapper for gatttool

	https://github.com/ampledata/pygatt