Nd Neodyme

BSIDES FRANKFURT

Keyless Entry

Hacking SwitchBot smart locks

Kolja Grassmann Security Researcher, Neodyme

MOTIVATION

Imagine you just put a new smart lock on your front door

It has a lot of convenient features and it is a popular brand. How sure are you that it is actually secure?

What we looked at

A best-selling smart lock that can be unlocked via app or via a keypad you put outside your door.

Research question

Does an attacker need a key to enter the home?

In this talk we will go over

Our research process, the issues we found and some takeaways

AGENDA

What we will cover

01 The lock and its attack surface

02 Firmware

03 BLE Communication

04 Defeating the tamper alarm

05 Round two: bypassing the patch

06 Takeaways

01

The target

THE TARGET

SwitchBot Lock Pro Starter Combo

Lock is attached onto your existing lock

Keypad with PIN, fingerprint reader and RF card

Hub is only needed for remote and voice unlock

Keypad and lock communicate over Bluetooth Low Energy

220 €
bundle price

~1 000
Amazon sales last month across models (September 2026)

02

Firmware

Retrieving the firmware

Dumping the firmware from the device

- We opened the unit and looked for storage to read out
- Did not find any external flash
- Unable to dump data from the microcontroller during the week

Android app

- Found URLs to an S3 bucket with update files in the app
- The bucket was publicly accessible
- Downloaded several firmware versions for offline analysis

03 BLE Communication

ARCHITECTURE

Decision to open the door happens on keypad

Access control is done on the keypad. After we enter the right PIN, it sends an unlock command to the lock over BLE.

To stop trivial man-in-the-middle, critical exchanges are encrypted with AES. Two different keys are used.

User key

Created when a user binds the lock to their account. Lets the phone talk to the lock.

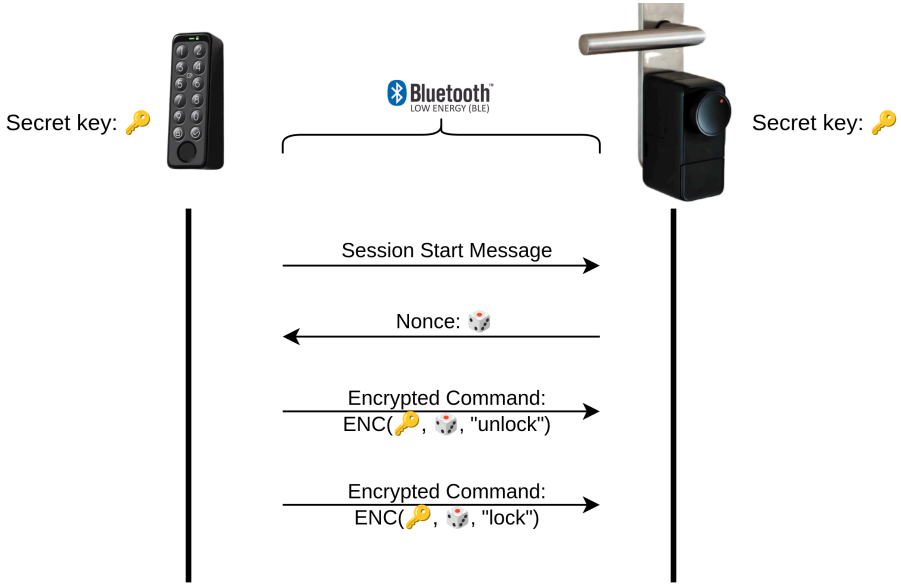
Pairing key

Created when keypad and lock are paired. Allows communication between the keypad and the lock.

THE HANDSHAKE

Communication protocol

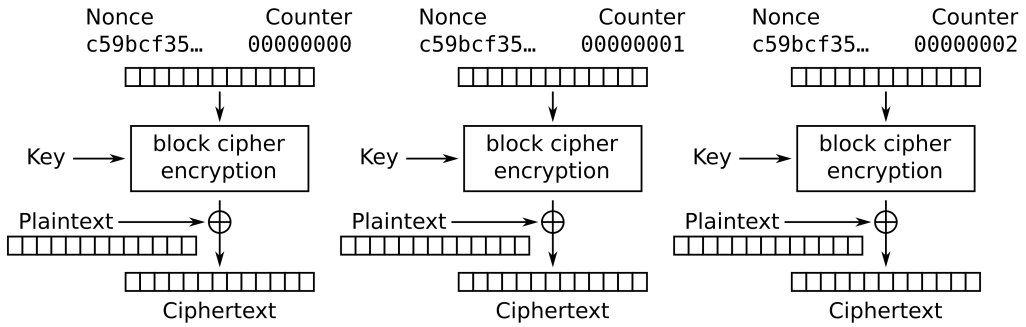
- Session start and the random nonce are transmitted without encryption
- The workflow is identical for keypad and phone
- AES-128 in CTR mode



REFRESHER

CTR mode turns a block cipher into a keystream

The cipher never touches the message. It encrypts nonce plus counter, and the result is XORed onto the plaintext.



Counter (CTR) mode encryption

Keystream depends only on key, nonce and counter

Same key and nonce means the same keystream, every time

No integrity: the mode says nothing about whether the ciphertext was altered

THE COMMANDS

Lock and unlock differ by one bit

```
COMMAND_LOCK   = 57 0f 4e 01 01 00 00 00
COMMAND_UNLOCK = 57 0f 4e 01 01 00 00 80
```

1 bit

No checksum. No HMAC. No integrity check of any kind on the message.

PROBLEM

AES-CTR does not ensure integrity

THE TWO COMMANDS DIFFER BY ONE BIT

```
DIFF           = LOCK_COMMAND ^ UNLOCK_COMMAND
UNLOCK_COMMAND = LOCK_COMMAND ^ DIFF
```

CTR JUST XORS A KEYSTREAM ON TOP

```
ENCRYPTED_LOCK_COMMAND          = KEYSTREAM ^ LOCK_COMMAND

ENCRYPTED_LOCK_COMMAND ^ DIFF = KEYSTREAM ^ LOCK_COMMAND ^ DIFF
                              = KEYSTREAM ^ UNLOCK_COMMAND
                              = ENCRYPTED_UNLOCK_COMMAND
```

LOCKING THE DOOR

Keypad sends a Lock command if we press the lock button

- Does not matter if the door is already locked
- Default behaviour, but it can be disabled in the app
- Gives an attacker a ciphertext to work with, on demand

ATTACK CHAIN

Unlocking the door without knowing the PIN

01	Do a BLE man-in-the-middle attack on the communication between keypad and lock
02	Press the keypad's lock button to provoke a Lock command
03	XOR the static mask with the intercepted ciphertext
04	Forward the manipulated version, which decrypts to an Unlock command and opens the door

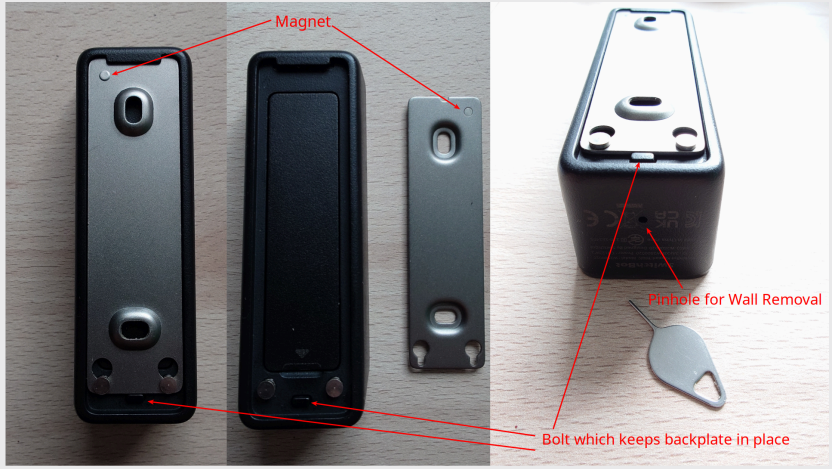
04

Defeating the tamper alarm

PROBLEM

Need distance between lock and keypad

- In a normal installation the keypad sits right next to the lock
- The lock will receive the original transmission, which messes with our attack
- We need to change that by moving the keypad away
- It is held by a backplate and released with a pin
- Removing it while the door is locked triggers the tamper alarm



Keypad, backplate, magnet and release pinhole

TAMPER DETECTION

Details on the tamper detection

Detection

A sensor in the keypad reads a magnet embedded in the backplate.

Alarm

If the magnet is gone an alarm is triggered, which continued until we removed the batteries.

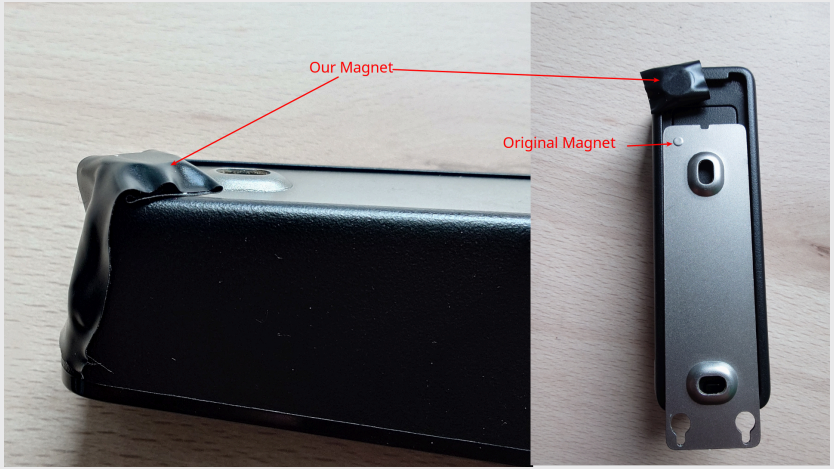
Notification

A push message is sent to the owner via the hub, or the next time the app is in range.

BYPASS

Bring your own magnet

- We can replace the magnet with our own while removing the keypad
- A magnet is still present, so the alarm is never triggered



Our magnet, taped in beside the original

Demo

05

The patch

THE PATCH

Truncated hash over the intended command

- SwitchBot patched the vulnerability and gave us the patched version to verify
- Commands now carry an MD5 hash over command, key and IV, truncated to the first two bytes

```
def calculate_checksum(command, key, iv):
    data = command + key + iv
    return hashlib.md5(data).digest()[:2]

full_command = COMMAND_UNLOCK + \
    calculate_checksum(COMMAND_UNLOCK, key, iv)
send_encrypted_bluetooth_command(full_command)
```

TWO WAYS TO BYPASS THE PATCH

Problems with the patch

65 536

possible tags. The session keeps the same IV and the connection allows retransmission for BLE timeouts, so the tag can simply be brute-forced.

App excluded

Only keypad commands carry the checksum. App commands stayed untouched, and the lock never checks which peer a message came from.

BYPASS

Changing a keypad command to an app command

Diff a keypad “Lock” command against an app “Unlock” command, apply the new static mask and drop the checksum. We can do this using the same bit-flip trick we did before.

06

Takeaways

TAKEAWAYS

What to take home

Encrypt and authenticate

Encryption does not ensure integrity. Use the right cryptographic primitives for the task at hand.

Unneeded functionality can enable attacks

There is no need to lock the door again if it is already locked. This functionality enabled our attack.

Decision logic should be on the lock

The keypad is in a hostile environment and can be tampered with, so the access decision should happen on the lock.

Shipping a fix takes time

Embedded devices update slowly and have to stay compatible with older peers. This makes fixing bugs hard.

Nd Neodyme

Thank you

Any questions?

Kolja Grassmann

Security Researcher, Neodyme

E-MAIL

kolja.grassmann@neodyme.io

BLOG

neodyme.io/blog

There will be a post on this bug soon

WE DO

Trainings · Pentests · Security research