SYNACKTIV

Sniffing DESFire authentication

Without a valid tag

PTS 2025 - RUMP

2025/07/02

Cloning tags

- **Many known issues**
  - Unprotected low frequency technologies
  - Mifare Classic vulnerabilities or hardcoded key (backdoor?)
  - Cleartext traffic
- **But why cloning them?**
  - Accessing facilities during physical intrusion
- **Tooling**
  - Proxmark (https://2019.pass-the-salt.org/files/slides/06-proxmark.pdf)

Examples

- **Social engineering: The technician**
  - Pretend to be a technician checking the tag
  - Cloning them
- **The lucky man**
  - After accessing offices, found a empty box of HID tagss
  - HID tag is based on two values, written on the box

Mifare DESFire

- **Authentication & encryption**
  - No known vulnerability on the latest versions
- **More security**
  - Proximity Check (since EV2)

Known attacks

- **UID based**
  - This is not a protocol vulnerability!
- **Relay**
  - But Proximity Check (since EV2)
- **Bruteforce keys**
  - Interesting if there are default keys
- **Encryption disabled**
  - Can be sniffed
  - But traffic is still authenticated

My goal

- **Sniff an authentication on a reader**
  - Without a valid tag
  - Check for weak/default keys

Mifare DESFire

- **How does it work?**
  - Create an application (imagine a folder) with different keys (read, write…)
    - You can choose your encryption (AES, 3DES…)
  - Put files with data in the application
- **How does authentication work?**
  - Pretty similar with the different algos
  - Sources
    - https://raw.githubusercontent.com/revk/DESFireAES/master/DESFire.pdf
    - Proxmark source code ;)

Mifare DESFire

SYNACKTIV

- **AES Authentication (simplified version)**
  - The tag creates a 16-byte random number (rndB) and encrypts with its key
  - The reader
    - decrypts the received 16-byte, using its AES key
    - generates its own 16-byte random number (rndA)
    - concatenates rndA and rndB (rotated to the left) together to make a 32-byte value
    - encrypts the 32-byte value with its key
    - sends to the tag
  - I'm not interested in the next steps ;)

Let's sniff

- **Tools**
  - Proxmark
  - Mifare DESFire EV2 tag

Let's go!

SYNACKTIV

- **Start sniffing mode on the proxmark**

```
[usb] pm3 --> hf 14a sniff
```

- **Put the tag on the proxmark and the proxmark on the reader**
- **Check the trace**

```
[usb] pm3 --> hf mfdes list
[...]
2018496 |    2026720 | Rdr |02  5A  00  00  00  66  1F           |  ok | SELECT APPLICATION (appId 000000)
2084256 |    2092480 | Rdr |03  5A  4F  49  42  FB  A8           |  ok | SELECT APPLICATION (appId 53894e)
```

- **Create an app with the AID (Application ID)**
  - Default is DES

```
[usb] pm3 --> hf mfdes createapp --aid 53894e
[+] Desfire application 53894e successfully created
```

Let's go!

- **Retry**
  - In order to get the key number and the authentication mode

```
5688080 |    5696304 | Rdr |02  5A  00  00  00  66  1F          |  ok | SELECT APPLICATION (appId 000000)
5748000 |    5756224 | Rdr |03  5A  4F  49  42  FB  A8          |  ok | SELECT APPLICATION (appId 53894e)
5784672 |    5790528 | Rdr |02  64  01  10  03                  |  ok | GET KEY VERSION (keyNo 1)
5813760 |    5819616 | Rdr |03  AA  01  76  09                  |  ok | AUTH AES (keyNo 1)
```

- **Delete your app**

```
[usb] pm3 --> hf mfdes deleteapp --aid 53894e
[+] Desfire application 53894e deleted
```

- **Recreate your app with the right algo**
  - For example for AES

```
[usb] pm3 --> hf mfdes createapp --aid 53894e --dstalgo aes
[+] Desfire application 53894e successfully created
```

Let's go!

- **Sniff again**

```
2161344 |    2167200 | Rdr |03  AA  01  76  09        |  ok | AUTH AES (keyNo 1)
2226548 |    2249716 | Tag |03  AF  CD [...]          |     |
        |            |     | 82  9B                   |  ok |
2268736 |    2310368 | Rdr |02  AF  C4  [...]         |     |
        |            |     |8F  09  A3  [...]         |  ok | AUTH FRAME / NEXT FRAME
2340484 |    2345220 | Tag |02  AE  64  61
```

- **Last steps**
  - Write a small bruteforcer using proxmark source code
  - Check for default keys

Conclusion

- Works also on Ultralight C/AES (similar authentication)
- No reader were harmed during this talk
- Questions?

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com