Bas les masques

Analyse d’une carte SD sécurisée

À propos

- Chercheur en sécurité, Suisse
- Principalement sur systèmes embarqués

- Organisateur BlackAlps
- Dév. principal Hydrabus

Carte SD ?

- Périphérique de stockage
  - Disponible depuis +20 ans
- Communique via un protocole spécifique
  - Orienté bloc (512 octets)

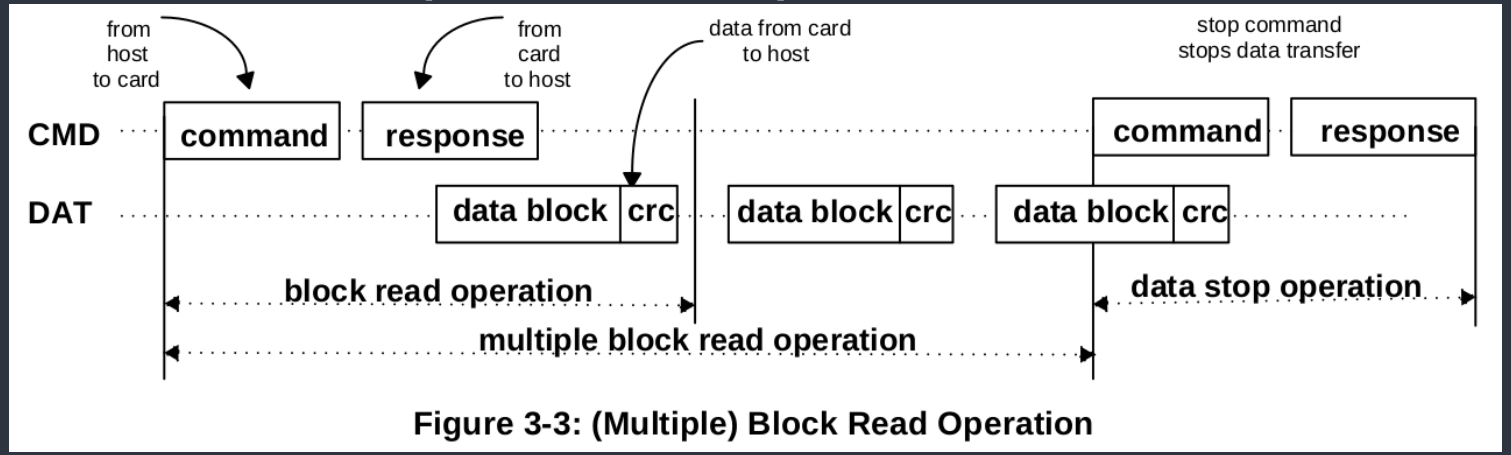


Figure 3-3: (Multiple) Block Read Operation

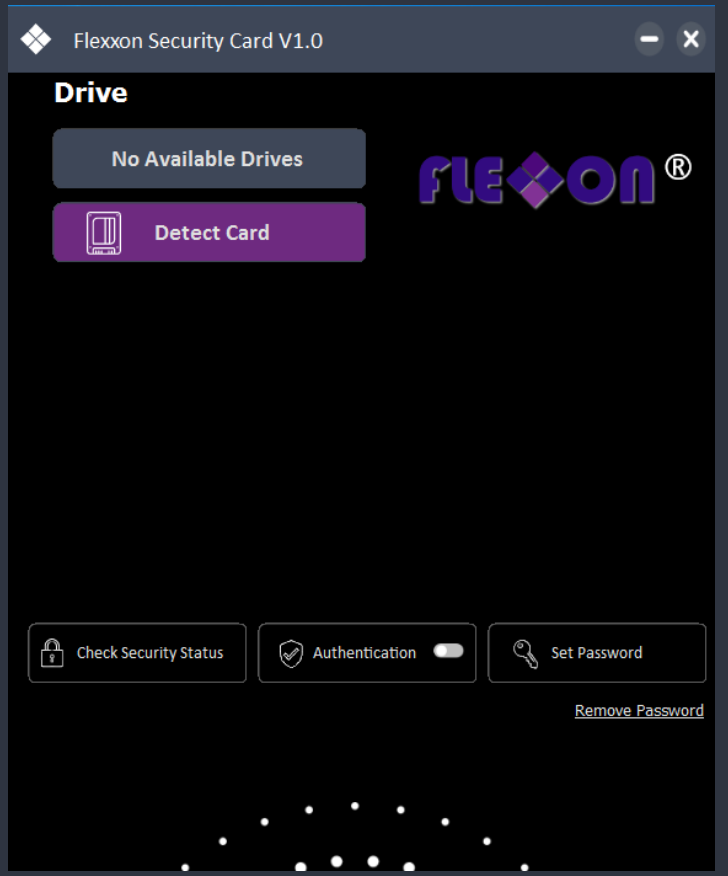
La cible

- Flexxon X-mask
- Données protégées par mot de passe
- Une fois configuré, les données ne sont plus accessibles sans mot de passe (Masque)
  - Lire des blocs retourne un bloc vide (que des 0xff)
  - Une fois le mot de passe saisi, la lecture est possible

https://www.flexxon.com/x-mask/

Utilisation

- La carte est débloquée par défaut
- Un outil est fourni par le fabricant pour la configuration
- L’outil permet aussi de débloquer la carte
- Que pour Windows :(



Analyse des données

- Premier essai : Analyseur logique
  - Difficile de capturer les bonnes commandes
- Deuxième essai : *usbmon*
- Un lecteur de carte USB et Wireshark
  - L’outil tourne dans une VM Windows
- Permet de facilement capturer de longues traces et de filtrer

Analyse des données

No.	Time	Source	Destination	Protocol	Length	Info
1041	2025-12-08 21:51:35.7778…	host	7.21.2	USBMS	95	SCSI Write(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000000
→ 1043	2025-12-08 21:51:35.7799…	host	7.21.2	USB	576	URB_BULK out
1047	2025-12-08 21:51:35.7841…	host	7.21.2	USBMS	95	SCSI Read(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000005
1050	2025-12-08 21:51:35.7981…	7.21.1	host	USB	576	URB_BULK in
1053	2025-12-08 21:51:35.8013…	host	7.21.2	USBMS	95	SCSI Read(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000005
1056	2025-12-08 21:51:35.8035…	7.21.1	host	USB	576	URB_BULK in
1065	2025-12-08 21:51:35.8138…	host	7.21.2	USBMS	95	SCSI Write(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000000
1067	2025-12-08 21:51:35.8159…	host	7.21.2	USB	576	URB_BULK out
1077	2025-12-08 21:51:35.8264…	host	7.21.2	USBMS	95	SCSI Write(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000000
1079	2025-12-08 21:51:35.8284…	host	7.21.2	USB	576	URB_BULK out
1097	2025-12-08 21:51:35.8471…	host	7.21.2	USBMS	95	SCSI Write(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000000
1099	2025-12-08 21:51:35.8492…	host	7.21.2	USB	576	URB_BULK out
1109	2025-12-08 21:51:35.8609…	host	7.21.2	USBMS	95	SCSI Write(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000000
1111	2025-12-08 21:51:35.8629…	host	7.21.2	USB	576	URB_BULK out
1121	2025-12-08 21:51:35.8734…	host	7.21.2	USBMS	95	SCSI Write(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000000
1123	2025-12-08 21:51:35.8755…	host	7.21.2	USB	576	URB_BULK out
1133	2025-12-08 21:51:35.8859…	host	7.21.2	USBMS	95	SCSI Write(10) LUN: 0×00 512 bytes (1 blocks) at LBA: 0×00000000

```
▸ Frame 1043: Packet, 576 bytes on wire (4608 bits), 576 bytes captured (4608 bits) on interface usbmon7, id 0
▸ USB URB
  Leftover Capture Data […]: 3639363638333031000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
0000  80 90 f9 03 4c 8f ff ff  53 03 02 15 07 00 2d 00   ····L··· S·····-·
0010  57 3a 37 69 00 00 00 00  ca e6 0b 00 8d ff ff ff   W:7i···· ········
0020  00 02 00 00 00 02 00 00  00 00 00 00 00 00 00 00   ········ ········
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  36 39 36 36 38 33 30 31  00 00 00 00 00 00 00 00   69668301 ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
```

Qu’apprend-t-on ?

- Quand on initialise le mot de passe :
  - Écriture bloc 0 (donnée fixe)
  - Lecture bloc 5 (deux fois)
    - La seconde fois, le bloc lu contient des données fixes et des données aléatoires
  - Écriture bloc 8 (donnée aléatoire)
  - Écriture bloc 0 (donnée fixe)

Qu’apprend-t-on ?

- Quand s’authentifie :
  - Écriture bloc 0 (donnée fixe)
  - Lecture bloc 5 (deux fois)
    - La seconde fois, le bloc lu contient des données fixes et des données aléatoires
  - Écriture bloc **6** (donnée aléatoire)
  - Écriture bloc 0 (donnée fixe)

Qu’apprend-t-on ?

- Quand s’authentifie :
  - Écriture bloc 0 (donnée fixe) <= Entrée commande
  - Lecture bloc 5 (deux fois)
    - La seconde fois, le bloc lu contient des données fixes et des données aléatoires
  - Écriture bloc **6** (donnée aléatoire)
  - Écriture bloc 0 (donnée fixe)

Qu’apprend-t-on ?

- Quand s’authentifie :
  - Écriture bloc 0 (donnée fixe)
  - Lecture bloc 5 (deux fois) <= Challenge
    - La seconde fois, le bloc lu contient des données fixes et des données aléatoires
  - Écriture bloc **6** (donnée aléatoire)
  - Écriture bloc 0 (donnée fixe)

Qu’apprend-t-on ?

- Quand s’authentifie :
  - Écriture bloc 0 (donnée fixe)
  - Lecture bloc 5 (deux fois)
    - La seconde fois, le bloc lu contient des données fixes et des données aléatoires
  - Écriture bloc **6** (donnée aléatoire) <= Réponse
  - Écriture bloc 0 (donnée fixe)

Qu’apprend-t-on ?

- Quand s’authentifie :
  - Écriture bloc 0 (donnée fixe)
  - Lecture bloc 5 (deux fois)
    - La seconde fois, le bloc lu contient des données fixes et des données aléatoires
  - Écriture bloc **6** (donnée aléatoire)
  - Écriture bloc 0 (donnée fixe) <= Sortie commande

Qu’apprend-t-on ?

- Quand s’authentifie :
  - Écriture bloc 0 (donnée fixe)
  - Lecture bloc 5 (deux fois)
    - La seconde fois, le bloc lu contient des données fixes et des données aléatoires
  - **Écriture bloc 6 (donnée aléatoire)**
  - Écriture bloc 0 (donnée fixe)
- Le numéro de bloc définit la commande !

Méthode de chiffrement ?

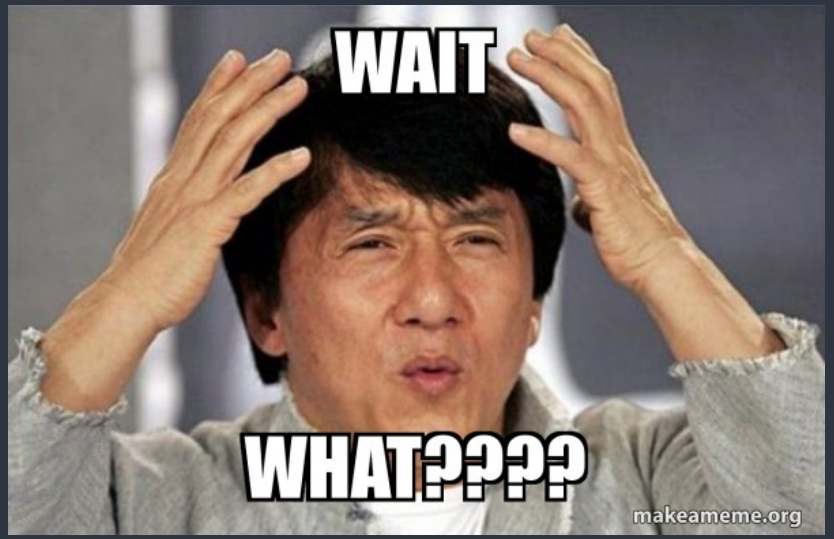
- Analyse du challenge-response
  - Rétro-ingénierie avec Ghidra
  - La librairie contient quelques symboles utiles
- Une fonction intéressante
  - Prend l’argument à envoyer et le paquet contenant le *challenge*

Méthode de chiffrement

- La fonction commence par initialiser un *Mersenne Twister* avec la date du PC
  - La graine n’est jamais transmise à la carte

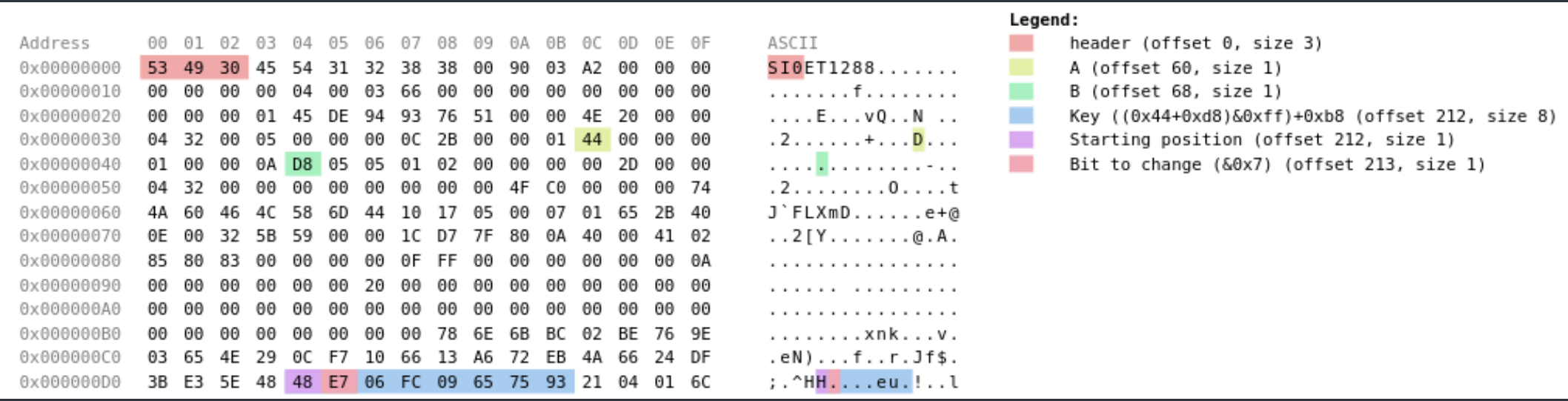
Méthode de chiffrement

- La fonction commence par initialiser un *Mersenne Twister* avec la date du PC
  - La graine n’est jamais transmise à la carte



“Chiffrement”

- Données prise dans le *Challenge*



Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
0x00000000	53	49	30	45	54	31	32	38	38	00	90	03	A2	00	00	00	SI0ET1288.......
0x00000010	00	00	00	00	04	00	03	66	00	00	00	00	00	00	00	00	.......f........
0x00000020	00	00	00	01	45	DE	94	93	76	51	00	00	4E	20	00	00	....E...vQ..N ..
0x00000030	04	32	00	05	00	00	00	0C	2B	00	00	01	44	00	00	00	.2......+...D...
0x00000040	01	00	00	0A	D8	05	05	01	02	00	00	00	00	2D	00	00	....`.........-..
0x00000050	04	32	00	00	00	00	00	00	00	00	4F	C0	00	00	00	74	.2........O....t
0x00000060	4A	60	46	4C	58	6D	44	10	17	05	00	07	01	65	2B	40	J`FLXmD......e+@
0x00000070	0E	00	32	5B	59	00	00	1C	D7	7F	80	0A	40	00	41	02	..2[Y.......@.A.
0x00000080	85	80	83	00	00	00	00	0F	FF	00	00	00	00	00	00	0A	....
0x00000090	00	00	00	00	00	00	20	00	00	00	00	00	00	00	00	00	....
0x000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0x000000B0	00	00	00	00	00	00	00	00	78	6E	6B	BC	02	BE	76	9E	........xnk...v.
0x000000C0	03	65	4E	29	0C	F7	10	66	13	A6	72	EB	4A	66	24	DF	.eN)...f..r.Jf$.
0x000000D0	3B	E3	5E	48	48	E7	06	FC	09	65	75	93	21	04	01	6C	;.^HH...eu.!..l

Legend:

- header (offset 0, size 3)
- A (offset 60, size 1)
- B (offset 68, size 1)
- Key ((0x44+0xd8)&0xff)+0xb8 (offset 212, size 8)
- Starting position (offset 212, size 1)
- Bit to change (&0x7) (offset 213, size 1)

“Chiffrement” (steg100)

- Transforme l’argument en bits (Poids fort)
- Transforme la clé en bits (Poids faible)
- Pour chaque bit de clé, encode 4 bits de l’argument
  - Si le bit de clé est 0, inverse les bits d’argument
- Écrit le bit <bit> de chaque octet avec un bit de l’argument depuis l’offset <pos>

“Chiffrement” (steg100)

-
-
-
-

le

Tests

- Utilise Hydrabus et le mode SDIO
  - Pas d’interférence de l’OS, scripting en Python
- Écrit un bloc de données de test
- Réimplémenté l’algorithme
  - Zappé le Mersenne Twister (inutile)

```python
# Login method
init_hydrabus()
init_target()
print("Before:")
hexdump.hexdump(read_block(0x2222)[:128])

enter_cmd2()
read_block(5)
P1 = read_block(5)
P2 = encode_response(P1, b'123456')
P2 = cksum(P2)
write_block(6, P2)
while s.send_short(13,raddr) != b'\x00\t\x00\x00':
    pass
leave_cmd2()
print("After:")
hexdump.hexdump(read_block(0x2222)[:128])
```

```
Before:
00000000: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000010: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000020: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000030: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000040: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000050: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000060: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000070: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
After:
00000000: 3D 53 45 43 52 45 54 3D  3D 53 45 43 52 45 54 3D  =SECRET==SECRET=
00000010: 3D 53 45 43 52 45 54 3D  3D 53 45 43 52 45 54 3D  =SECRET==SECRET=
00000020: 3D 53 45 43 52 45 54 3D  3D 53 45 43 52 45 54 3D  =SECRET==SECRET=
00000030: 3D 53 45 43 52 45 54 3D  3D 53 45 43 52 45 54 3D  =SECRET==SECRET=
00000040: 3D 53 45 43 52 45 54 3D  3D 53 45 43 52 45 54 3D  =SECRET==SECRET=
00000050: 3D 53 45 43 52 45 54 3D  3D 53 45 43 52 45 54 3D  =SECRET==SECRET=
00000060: 3D 53 45 43 52 45 54 3D  3D 53 45 43 52 45 54 3D  =SECRET==SECRET=
00000070: 3D 53 45 43 52 45 54 3D  3D 53 45 43 52 45 54 3D  =SECRET==SECRET=
```

Exploration

- Maintenant, possible d’envoyer des arguments arbitraires aux commandes
- Test des autres commandes
- La commande 9 est intéressante
  - Définit l’adresse à partir de laquelle le masque est appliqué
    - 0 par défaut
  - Probablement utilisé pour définir deux partitions, mais jamais utilisée par le logiciel

Contournement d’authentification

- Cette commande est acceptée sans authentification
- Passer 0xffffffff en argument place le début de la protection après le dernier bloc physique
- Permet de lire et écrire tous les blocs sans authentification

Et Linux alors ?

- Comme il n’y a pas de support Linux, vibe-codé un outil
- Flemme de coder la partie saisie de mot de passe, l’outil utilise la vulnérabilité pour débloquer la carte
- https://github.com/Baldanos/flexxon-tool

Démo !

Divulgation

- 10.12.2025: Premier contact en utilisant le formulaire sur le site
- 17.12.2025: Second contact par mail
- 07.02.2026: Troisième contact par mail

- Aucune réponse
  - Pas sûr qu’une remédiation soit possible

Conclusions

- Protection par mot de passe inutile
  - Le chiffrement ne change rien
- Si vous êtes une entreprise, créez un contact sécurité
  - Au pire, demandez au vendeurs de transmettre les demandes aux ingés
- Le Full Disclosure existe toujours !

Merci !

Bas les masques: Analyse d’une carte SD sécurisée

Nicolas Oberli
@baldanos
https://balda.ch

Le CFP pour BlackAlps est ouvert !
https://blackalps.ch/