Risky USBusiness

Say "what the fuzz."... If you can't say it, you can't do it.

Jordan BOUYAT
jbouyat@quarkslab.com
@la_F0uin3

Fernand LONE-SANG
flonesang@quarkslab.com

QUARKSLAB
INNOVATIVE SECURITY

Constat

Omniprésence de l'USB

- Postes de travail
- Bornes automatiques en tout genre
- Imprimantes
- Matériel embarqué
- Etc

Massivement utilisé, mais fonctionnement interne assez peu connu.

Intêrets

Attaques possibles

Les périphériques USB sont un vecteur d'attaque

- Accès physique à une machine pour peu de temps
- Périphérique qu'on laisse traîner volontairement
- Attaque sur un réseau coupé du Net

Sommaire de la présentation

Sommaire de la présentation

Hiérarchie

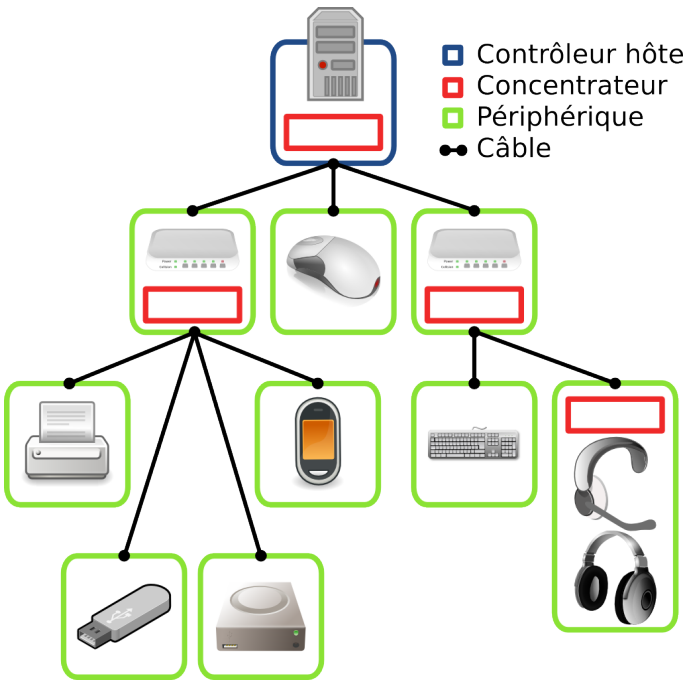
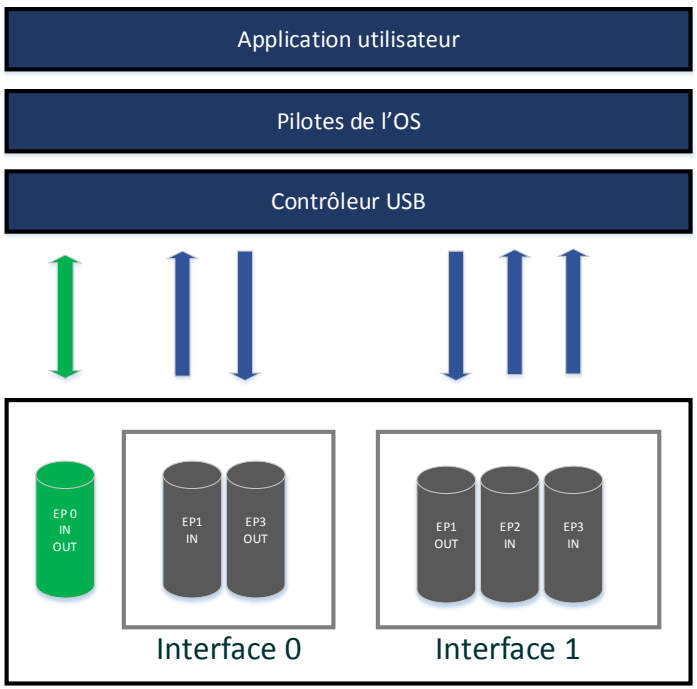


Figure: Topologie USB

- Une topologie hiérarchisée
- 1 contrôleur hôte : 127 périphériques
- Un hub peut être connecté à un autre hub
- Les connexions sont initiées par l'hôte (sauf OTG)

Vue d'un périphérique

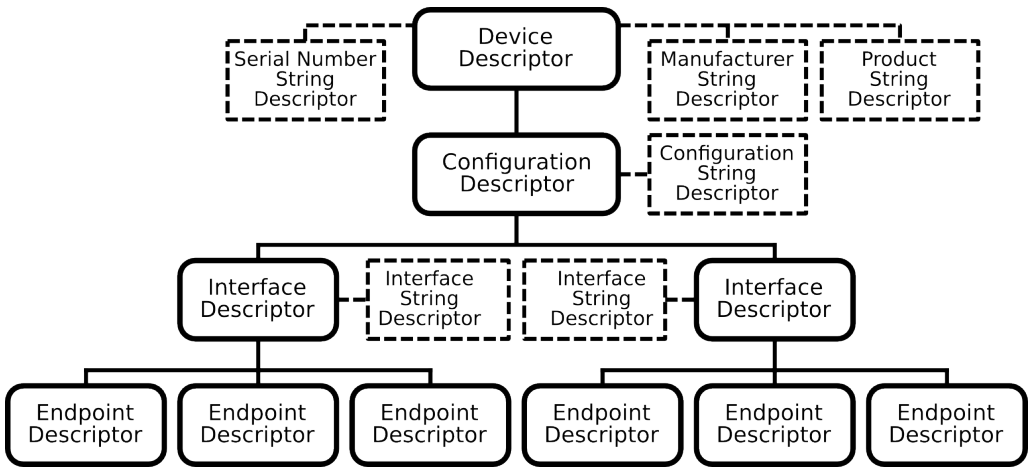


- Une interface fournit une fonctionnalité
- Elle est composée d'*endpoints*
- Les *endpoints* sont des liens logiques entre le périphérique et les pilotes
- Ils peuvent envoyer ou recevoir des données d'un type de transfert spécifique :
  - Control
  - Interrupt
  - Bulk
  - Isochronous

Descripteurs

Structures de données décrivant un périphérique :

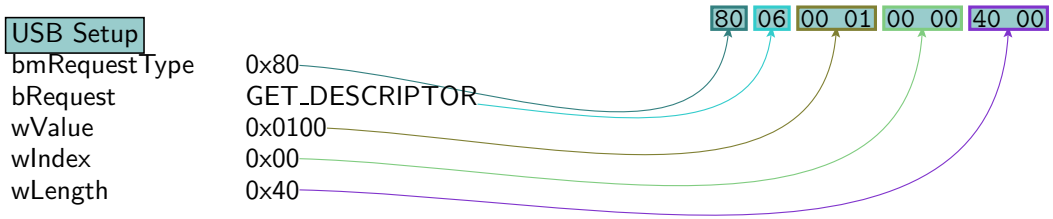
1. ses caractéristiques (version USB, VID, PID...) ;
2. ses interfaces (type, nombre d'*endpoints*...) ;
3. ses *endpoints* (direction, type de transfert...) ;

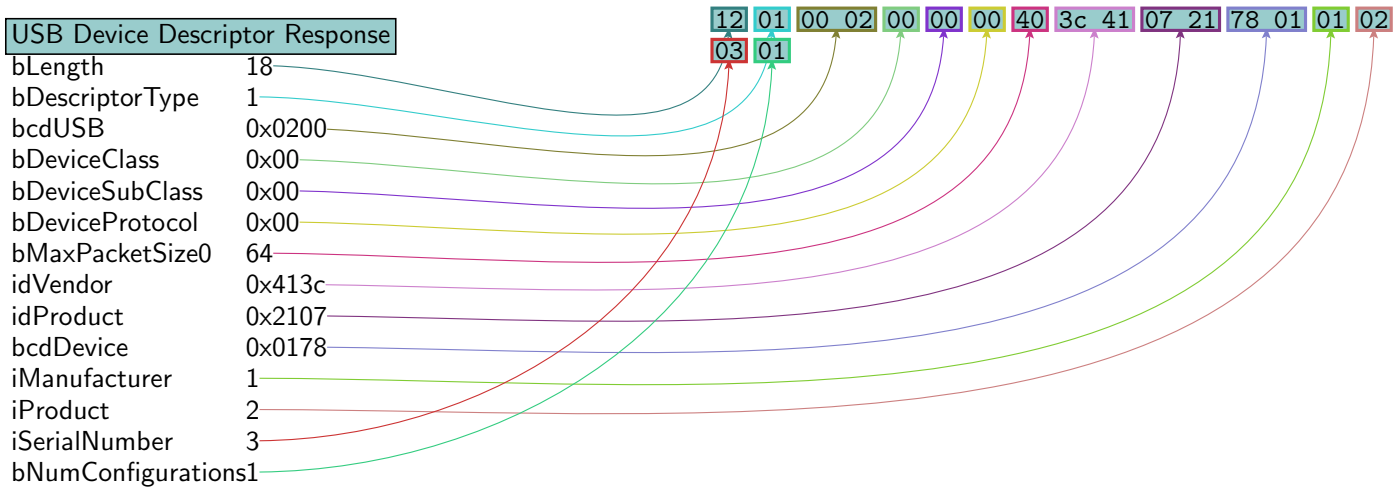


Un descripteur de configuration correspond à différentes associations d'interfaces.

Requêtes standards

Les descripteurs sont récupérés pendant la phase d'**énumération**.





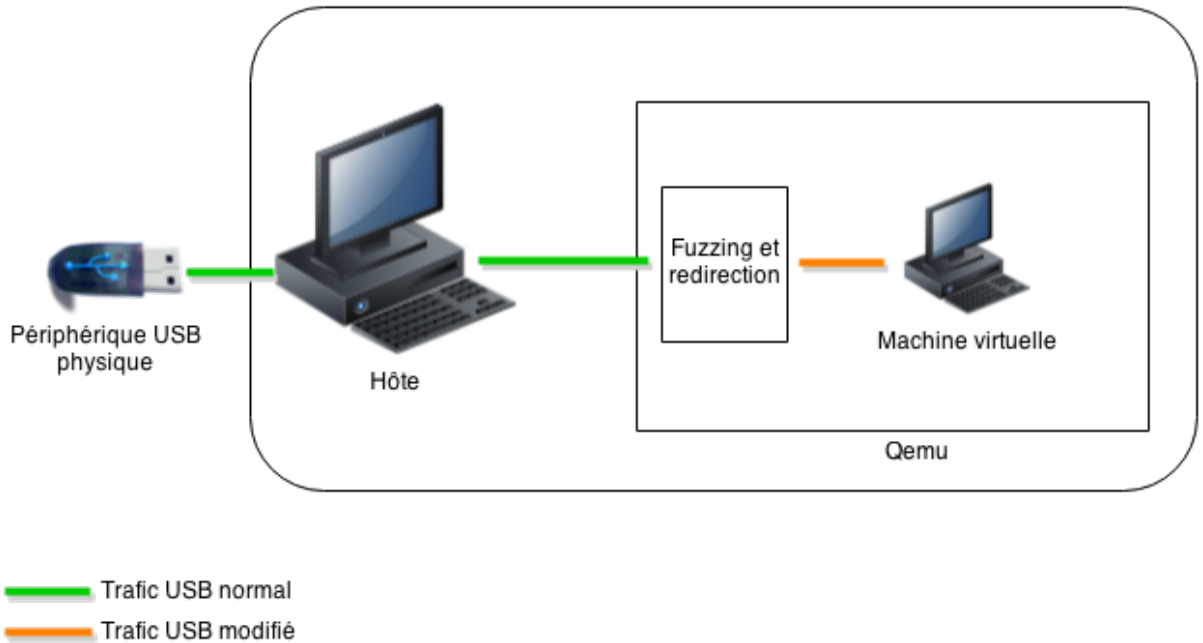
Énumération

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	host	0.0	USB	36	GET DESCRIPTOR Request DEVICE
2	0.000104	0.0	host	USB	46	GET DESCRIPTOR Response DEVICE
3	0.041951	host	0.0	USB	36	SET ADDRESS Request
4	0.064879	host	1.0	USB	36	GET DESCRIPTOR Request DEVICE
5	0.064948	1.0	host	USB	46	GET DESCRIPTOR Response DEVICE
6	0.080860	host	1.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
7	0.080987	1.0	host	USB	60	GET DESCRIPTOR Response CONFIGURATION
8	0.101878	host	1.0	USB	36	GET DESCRIPTOR Request STRING
9	0.102372	1.0	host	USB	62	GET DESCRIPTOR Response STRING
10	0.123878	host	1.0	USB	36	GET DESCRIPTOR Request STRING
11	0.123943	1.0	host	USB	32	GET DESCRIPTOR Response STRING
12	0.138879	host	1.0	USB	36	GET DESCRIPTOR Request STRING
13	0.138943	1.0	host	USB	50	GET DESCRIPTOR Response STRING
14	0.157873	host	1.0	USB	36	GET DESCRIPTOR Request DEVICE QUALIFIER
15	0.157938	1.0	host	USB	38	GET DESCRIPTOR Response DEVICE QUALIFIER
16	0.182785	host	1.0	USB	36	GET DESCRIPTOR Request DEVICE
17	0.182851	1.0	host	USB	46	GET DESCRIPTOR Response DEVICE
18	0.198830	host	1.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
19	0.198912	1.0	host	USB	37	GET DESCRIPTOR Response CONFIGURATION
20	0.212812	host	1.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
21	0.212884	1.0	host	USB	60	GET DESCRIPTOR Response CONFIGURATION
22	0.231808	host	1.0	USB	36	GET DESCRIPTOR Request STRING
23	0.231869	1.0	host	USB	30	GET DESCRIPTOR Response STRING[Malformed Packet]
24	0.244788	host	1.0	USB	36	GET DESCRIPTOR Request STRING
25	0.244866	1.0	host	USB	32	GET DESCRIPTOR Response STRING
26	0.257752	host	1.0	USB	36	GET DESCRIPTOR Request STRING
27	0.257816	1.0	host	USB	30	GET DESCRIPTOR Response STRING[Malformed Packet]
28	0.270781	host	1.0	USB	36	GET DESCRIPTOR Request STRING
29	0.270844	1.0	host	USB	62	GET DESCRIPTOR Response STRING
30	0.289728	host	1.0	USB	36	SET CONFIGURATION Request
31	0.312729	host	1.0	USBMS	36	GET MAX LUN Request
32	0.312779	1.0	host	USBMS	29	GET MAX LUN Response

Sommaire de la présentation

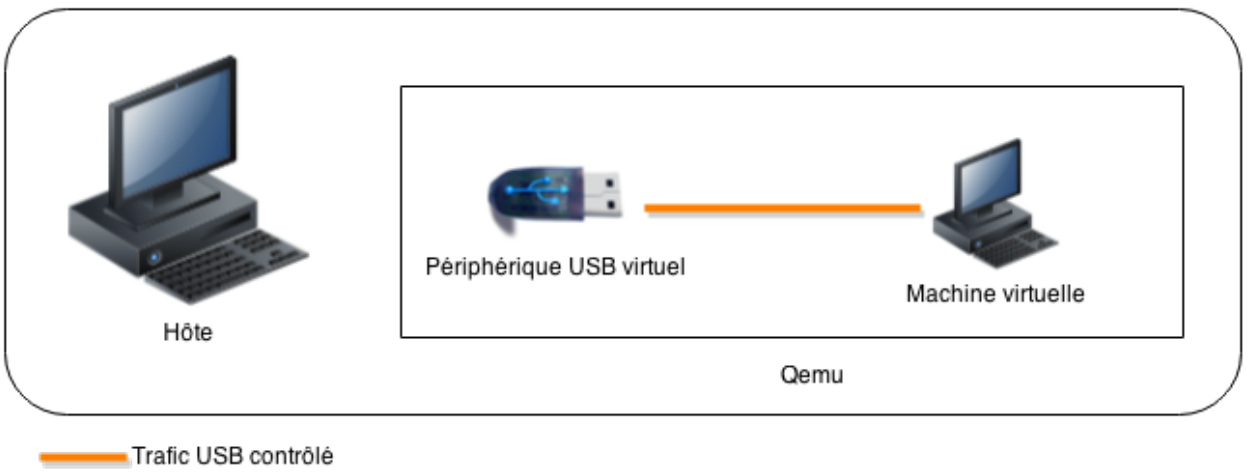
Qemu : configuration 1

Dumb fuzzer en interceptant le traffic circulant entre un périphérique et une machine virtuelle.



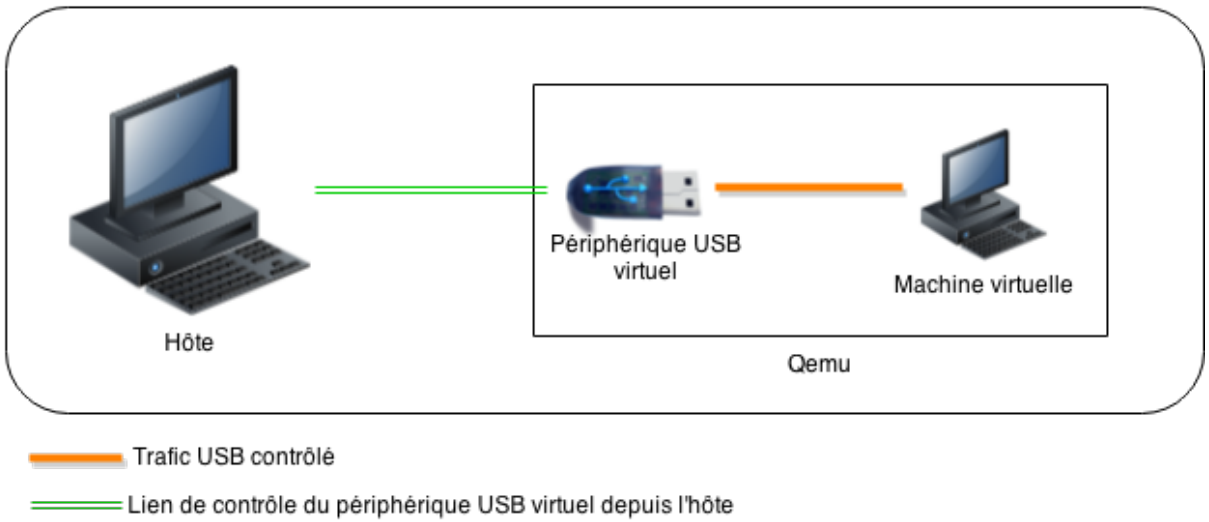
Qemu : configuration 2

Périphérique virtuel avec *fuzzer* intégré



Qemu : configuration 3

Les trames sont remontées en espace utilisateur, *fuzzées* et renvoyées dans la VM via le périphérique.



Retours

Avantages :

- Restauration rapide du système dans un état grâce aux *snapshots*
- Instrumentation et *monitoring* plus poussés de l'OS cible
- Parallélisation possible

Inconvénients :

- On ne peut pas virtualiser tous les systèmes
- Bugs possibles dans l'implémentation même d'USB au sein de l'hyperviseur

Possibilités

Matériel dédié classique

Avantage : Capture bas niveau/rejeu, langage de *scripting*
Inconvénient : Cher, peu flexible au niveau de l'API
Exemple : Beagle USB* de Totalphase

Micro-contrôleurs classiques et FPGA

Avantage : Peu cher
Inconvénient : Re-*flashage* à chaque modification du *fuzzer* : peu pratique
Exemple : PIC, AVR dont le Teensy avec LUFA, Daisho pour le FPGA

Solution intermédiaire : Facedancer ?

Facedancer

Présentation

- Développé par Travis Goodspeed
- Comprend un adaptateur série/USB, un micro-contrôleur et un contrôleur USB
- Permet d'émuler des périphériques USB en les contrôlant via des scripts Python sur une machine distante

Figure: http://int3.cc/

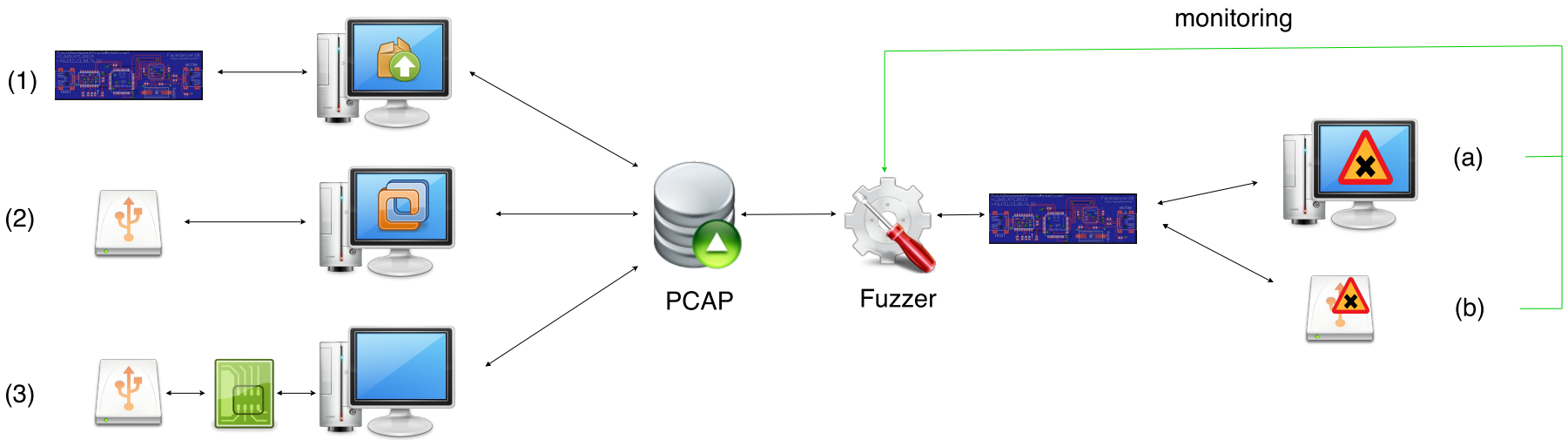
Limitations

- Uniquement 3 *endpoints*
- Pas de support des transferts isochrones
- Débit limité en raison de la liaison série *over* USB
- Pas de support d'USB3

Néanmoins, le Facedancer suffit pour commencer à *fuzzer*.

Sommaire de la présentation

Architecture visée

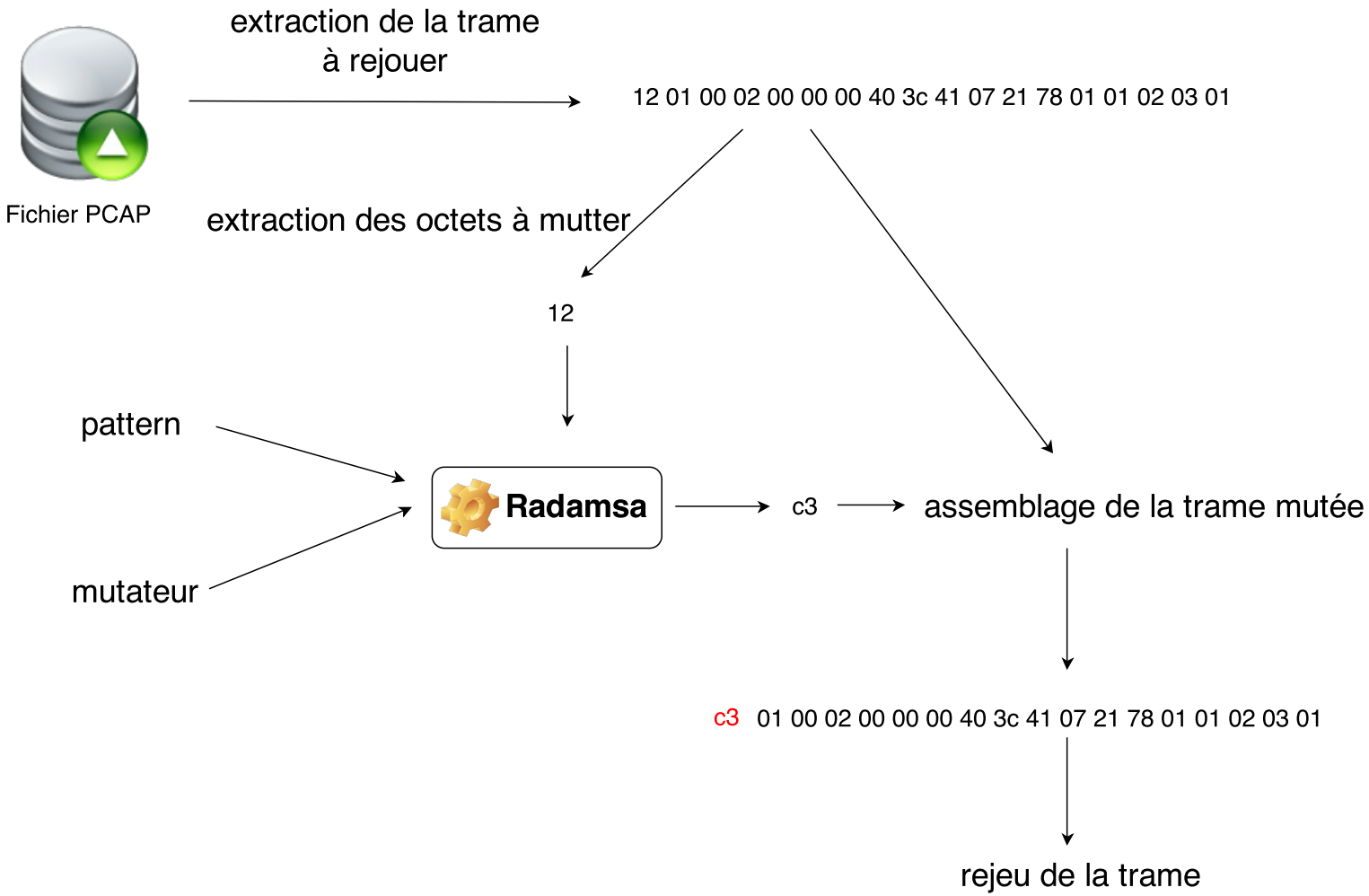


(1) : Capture du trafic du Facedancer
(2) : Capture du trafic dans VMware
(3) : Capture du trafic avec un sniffer physique

(a) : fuzzing d'hôte
(b) : fuzzing de périphérique

Figure: Architecture de *fuzzing* USB

Utilisation



Détails techniques

Base

- Basé sur l'outil *open source* Umap développé par Andy Davis.
- Umap repose sur le code du dépôt SVN de Travis Goodspeed.

Apports

Modifications

- Capture en PCAP et rejeu strict
- Mutations des trames rejouées avec Radamsa
- Choix des trames, octets et *patterns* de *fuzzing* à appliquer
- Moniteur avec rapport de crash
- Rejeu en mode "pas à pas" pour le debug

Sommaire de la présentation

Résultats obtenus sur Windows 8.1

Parsing HID

D'autres valeurs d'octets déclenchant le même crash que Davis :
Non exploitable.

Périphérique de stockage de masse

Mauvais contrôle des informations concernant les interfaces dans USBSTOR.sys :
Non exploitable.

Descripteur muté

```
⊟ CONFIGURATION DESCRIPTOR
    bLength: 9
    bDescriptorType: CONFIGURATION (2)
    wTotalLength: 32
    bNumInterfaces: 1
    bConfigurationValue: 1
    iConfiguration: 4
  ⊞ Configuration bmAttributes: 0xe0  SELF-POWERED  REMOTE-WAKEUP
    bMaxPower: 50  (100mA)
⊟ INTERFACE DESCRIPTOR (0.0): class Mass Storage
    bLength: 9
    bDescriptorType: INTERFACE (4)
    bInterfaceNumber: 0
    bAlternateSetting: 0
    bNumEndpoints: 0
    bInterfaceClass: Mass Storage (0x08)
    bInterfaceSubClass: 0x06
    bInterfaceProtocol: 0x50
    iInterface: 0
⊞ ENDPOINT DESCRIPTOR
⊞ ENDPOINT DESCRIPTOR
```

Création d'un descripteur de configuration renseignant une interface ne contenant aucun *endpoints*.
Résultat : crash.

Enumération

Source	Destination	Protocol	Length	Info
host	0.0	USB	36	GET DESCRIPTOR Request DEVICE
0.0	host	USB	46	GET DESCRIPTOR Response DEVICE
host	0.0	USB	36	SET ADDRESS Request
host	1.0	USB	36	GET DESCRIPTOR Request DEVICE
1.0	host	USB	46	GET DESCRIPTOR Response DEVICE
host	1.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
1.0	host	USB	60	GET DESCRIPTOR Response CONFIGURATION
host	1.0	USB	36	GET DESCRIPTOR Request STRING
1.0	host	USB	62	GET DESCRIPTOR Response STRING
host	1.0	USB	36	GET DESCRIPTOR Request STRING
1.0	host	USB	32	GET DESCRIPTOR Response STRING
host	1.0	USB	36	GET DESCRIPTOR Request STRING
1.0	host	USB	50	GET DESCRIPTOR Response STRING
host	1.0	USB	36	GET DESCRIPTOR Request DEVICE QUALIFIER
1.0	host	USB	38	GET DESCRIPTOR Response DEVICE QUALIFIER
host	1.0	USB	36	GET DESCRIPTOR Request DEVICE
1.0	host	USB	46	GET DESCRIPTOR Response DEVICE
host	1.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
1.0	host	USB	37	GET DESCRIPTOR Response CONFIGURATION
host	1.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
1.0	host	USB	60	GET DESCRIPTOR Response CONFIGURATION
host	1.0	USB	36	GET DESCRIPTOR Request STRING
1.0	host	USB	30	GET DESCRIPTOR Response STRING[Malformed Packet]
host	1.0	USB	36	GET DESCRIPTOR Request STRING
1.0	host	USB	32	GET DESCRIPTOR Response STRING
host	1.0	USB	36	GET DESCRIPTOR Request STRING
1.0	host	USB	30	GET DESCRIPTOR Response STRING[Malformed Packet]
host	1.0	USB	36	GET DESCRIPTOR Request STRING
1.0	host	USB	62	GET DESCRIPTOR Response STRING
host	1.0	USB	36	SET CONFIGURATION Request

Pilotes du contrôleur et pilotes du système

USBSTOR.sys

Analyse du crash

On passe dans `USBSTOR_SelectConfiguration`

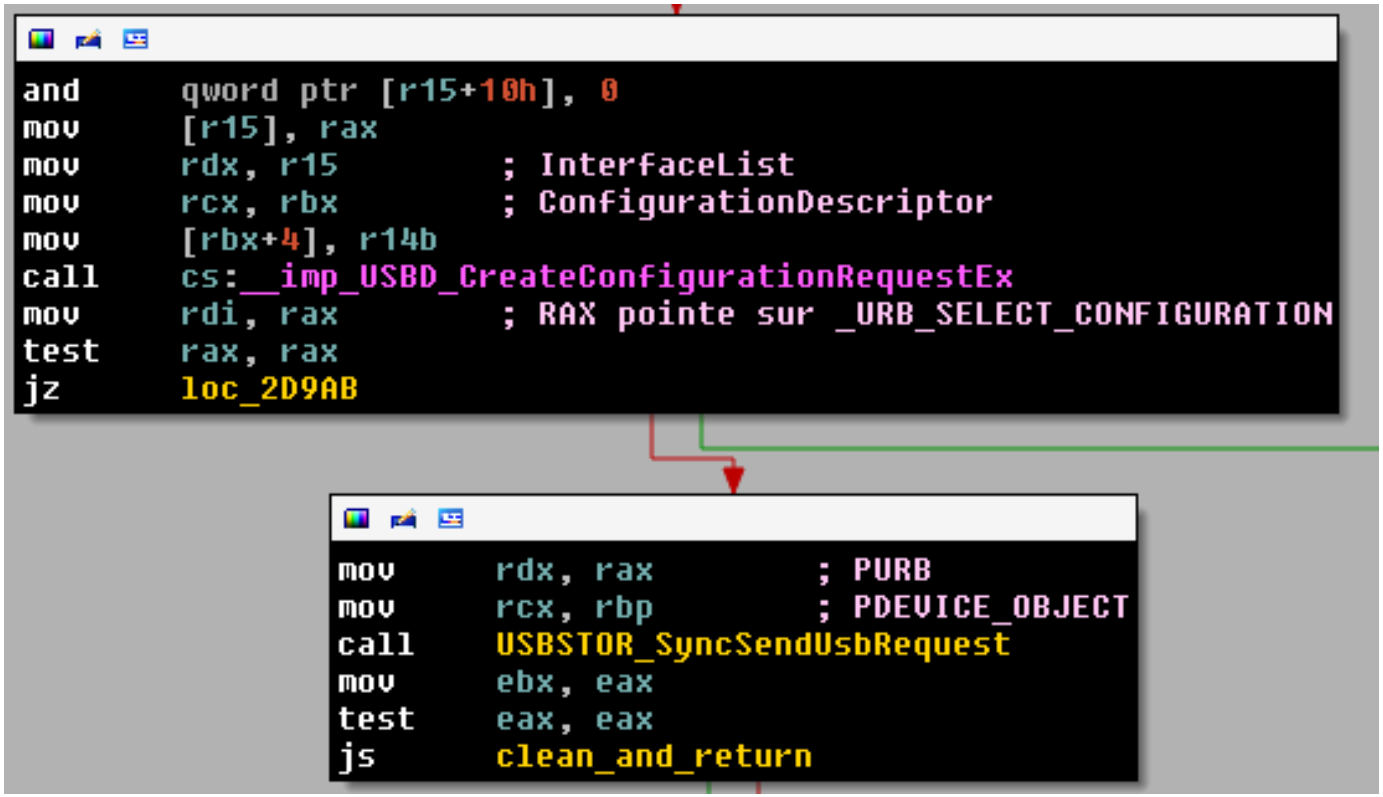


Figure: USBSTOR.sys : `USBSTOR_SelectConfiguration+EE`

Analyse du crash

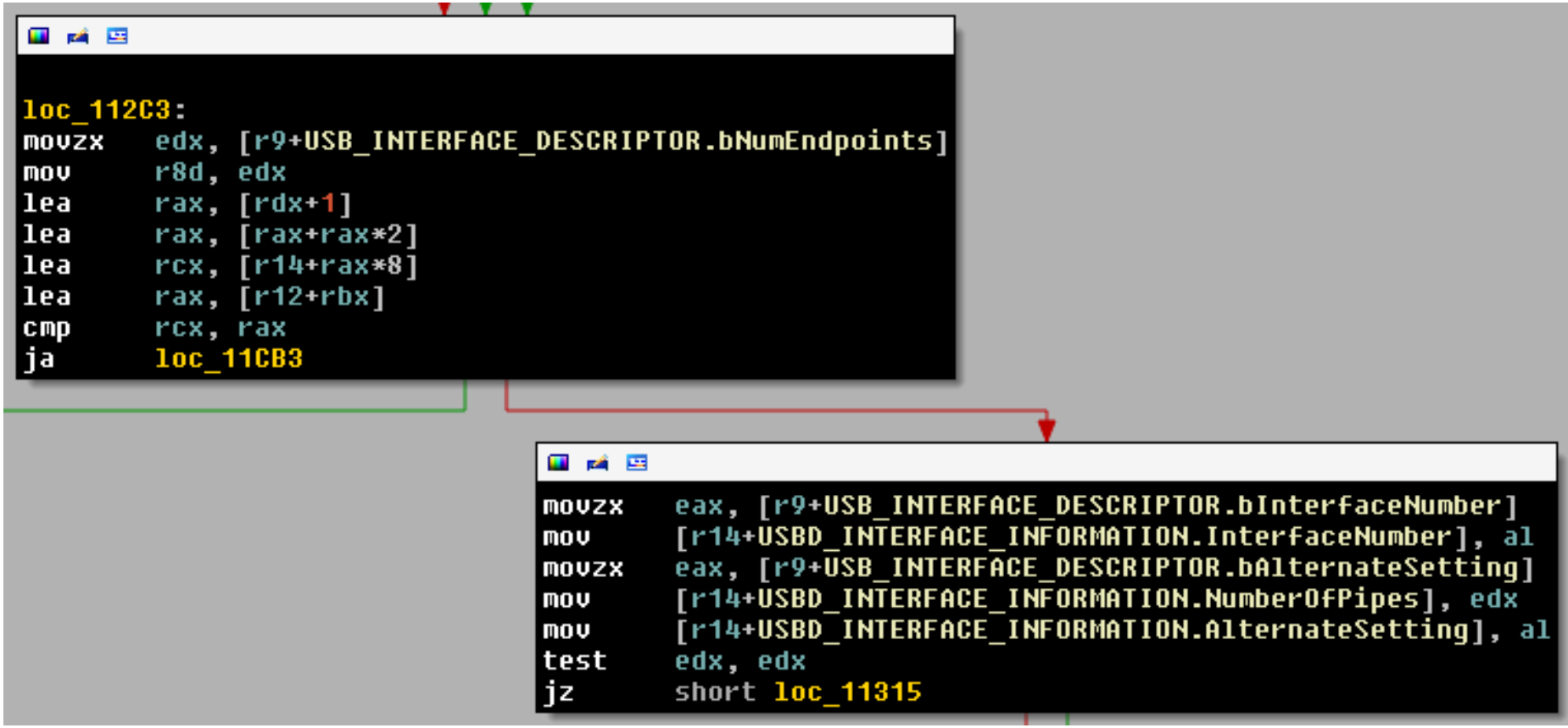


Figure: usbd.sys : `USBD_CreateConfigurationRequestEx+113`

Recopie du champ `USB_INTERFACE_DESCRIPTOR.bNumEndpoints`

Analyse du crash

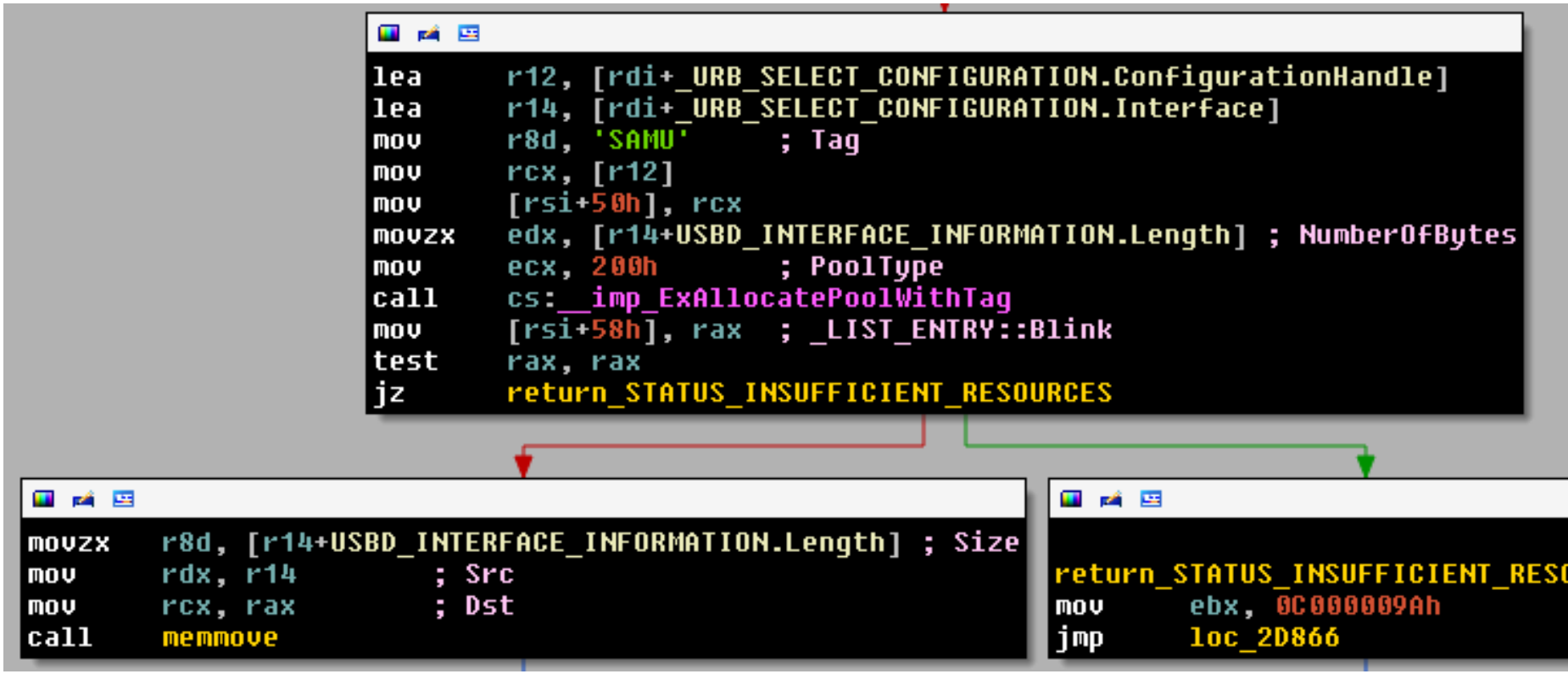
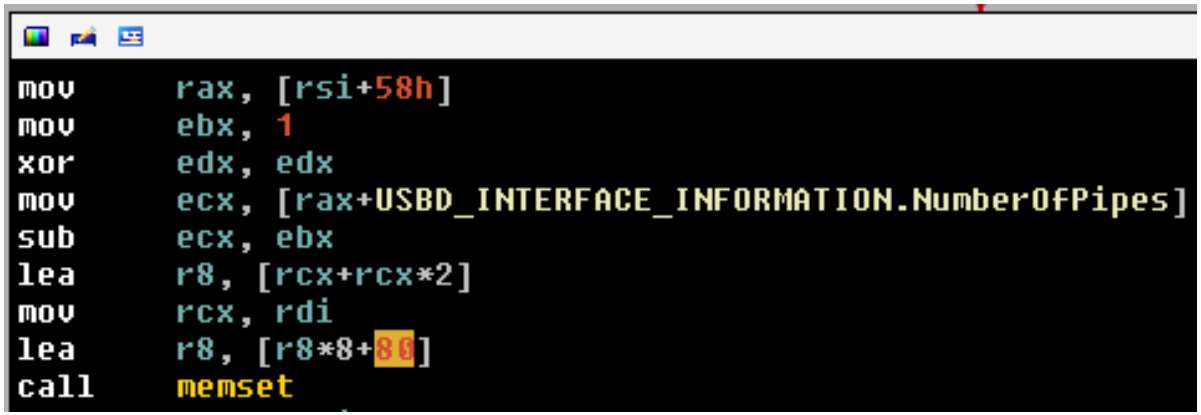


Figure: USBSTOR.sys : `USBSTOR_SelectConfiguration+11`

Copie de la structure `USBD_INTERFACE_INFORMATION`

Origine du crash en x64



```
mov     rax, [rsi+58h]
mov     ebx, 1
xor     edx, edx
mov     ecx, [rax+USBD_INTERFACE_INFORMATION.NumberOfPipes]
sub     ecx, ebx
lea     r8, [rcx+rcx*2]
mov     rcx, rdi
lea     r8, [r8*8+80]
call    memset
```

$ECX \longleftarrow$ nombre d'*endpoints*
$ECX \longleftarrow ECX - 1$
$R8 \longleftarrow 3 * RCX$
$R8 \longleftarrow R8 * 8 + 80$
`memset(@dest, 0x0, R8)`

Si nombre d'*endpoints* $= 0$:
$ECX \longleftarrow 0 - 1 = 0xffffffff$
$R8 \longleftarrow 0xffffffff * 3 = 0x0002fffffffd$
$R8 \longleftarrow 0x0002fffffffd * 8 + 80 = 0x1800000038$
`memset(@dest, 0x0, 0x1800000038)`

Problème en x86

```
mov     eax, [ebx+2Ch]
push    38h             ; sizeof(_URB_SELECT_CONFIGURATION)
pop     esi
mov     eax, [eax+USBD_INTERFACE_INFORMATION.NumberOfPipes]
dec     eax
imul    eax, 14h        ; sizeof(USBD_PIPE_INFORMATION)
add     eax, esi
push    eax
push    0
push    edi
call    _memset
```

$EAX \longleftarrow$ nombre d'*endpoints*
$EAX \longleftarrow ECX - 1$
$EAX \longleftarrow EAX * 0x14 + 0x38$
`memset(@dest, 0x0, EAX)`

Si nombre d'*endpoints* $= 0$:
$EAX \longleftarrow 0 - 1 = 0xffffffff$
$EAX \longleftarrow 0xffffffff * 0x14 + 0x38 = 0x24$
`memset(@dest, 0x0, 0x24)`

Les 20 derniers octets de la structure `_URB_SELECT_CONFIGURATION` ne sont pas initialisés.

Sommaire de la présentation

Conclusion et perspectives

Avancement

- Sources de capture opérationnelles : Facedancer et VMware
- Fuzzing d'hôte fonctionnel

Reste à faire

- Augmenter les performances :
  - FPGA
  - Carte ARM avec port OTG pour capture/rejeu et émulation avec USBGadget
- Implémenter le *fuzzing* de périphérique
- Ajouter d'autres sources de capture : sniffer matériel
- Support USB3

Questions?

Merci à toute l'équipe de QuarksLab et en particulier à Fernand Lone-Sang, Kevin Szkudlapski et Damien Aumaître.

QUARKSLAB
INNOVATIVE SECURITY

www.quarkslab.com

contact@quarkslab.com | @quarkslab.com