Dom 27 mar 2016

SteelSeries Rival 100: ingeniería inversa de un dispositivo USB

La semana pasada me compré (un poco por capricho, lo admito) un nuevo mouse: el **SteelSeries Rival 100** . Este mouse ofrece muchas opciones de personalización, como el color de la luz de fondo o la configuración de la sensibilidad del sensor ... el único *problema* es que estas opciones solo son accesibles a través de un pequeño software, el **SteelSeries Engine 3** , que está disponible solo en Windows y Mac OS X.

Como mi PC se ejecuta exclusivamente en **Ubuntu** y no tengo el menor deseo de arrancar una máquina Windows cada vez que quiero configurar mi mouse, comencé la ingeniería inversa del mouse para poder desarrollar Una pequeña utilidad para configurarlo directamente bajo Linux.

Para la ingeniería inversa del ratón, necesitaré:

- una máquina virtual bajo Windows,
- de Wireshark,
- y un segundo mouse (u otro dispositivo señalador, como un touchpad o trackpoint).

Usaré la máquina virtual para ejecutar la utilidad de configuración, y mientras juego con la configuración, capturaré el tráfico del bus USB usando Wireshark, así que espero para poder

Recuperando información en el dispositivo USB

El primer paso es reunir alguna información sobre el ratón. El comando `lsusb` debe proporcionar una cantidad de información útil:

```
$ lsusb
...
Bus 001 Dispositivo 022: ID 1038: 1702 SteelSeries ApS
```

Aquí aprendemos:

- que el ratón está conectado al **bus USB número 1** ,
- que el sistema le dio el **número de serie 22** ,
- que el **identificador del fabricante** (*ID del proveedor*) es **1038** ,
- que el **identificador de este modelo de** mouse (*ID de producto*) es **1702** .

Ya es bastante información, pero deberíamos poder encontrar algo más en los registros del kernel de Linux:

```
$ dmesg
...
[23803.145407] usb 1-1: nuevo número de dispositivo USB de velocidad completa 22 con x
[23803.382869] usb 1-1: Nuevo dispositivo USB encontrado, idVendor = 1038, idProduct =
[23803.382871] usb 1-1: Nuevas cadenas de dispositivos USB: Mfr = 1, Product = 2, Seri
[23803.382873] usb 1-1: Producto: SteelSeries Rival 100 Gaming Mouse
[23803.382874] usb 1-1: Fabricante: SteelSeries
[23803.385604] hid-generic 0003: 1038: 1702.000D: hiddev0, hidraw6: USB HID Device v1.
[23803.387969] entrada: SteelSeries SteelSeries Rival 100 Gaming Mouse como /devices/p
[23803.388128] hid-generic 0003: 1038: 1702.000E: input, hidraw7: HID USB v1.11 Mouse
```

Aquí encontramos la mayoría de la información vista anteriormente, pero las líneas `"hid-genérico"` nos informan que el mouse proporciona **dos interfaces que** implementan la **clase HID** (*Dispositivo de Interfaz Humana*), que es una muy buena noticia.

Las interfaces que implementan esta clase son muy simples de controlar porque el kernel de Linux nos proporciona a cada uno de nosotros un archivo en `/ dev` para comunicarnos con estas interfaces sin tener que preocuparnos por los detalles del protocolo.

Estas dos líneas también nos informan que estas dos interfaces están vinculadas a los archivos `hidraw6` y `hidraw7` (en `/ dev`), que podemos verificar fácilmente:

```
/ dev / hidraw0 / dev / hidraw2 / dev / hidraw4 / dev / hidraw6
/ dev / hidraw1 / dev / hidraw3 / dev / hidraw5 / dev / hidraw7
```

Preparando la máquina virtual en Windows

Ahora que estamos en posesión de toda la información necesaria para identificar el dispositivo, es hora de continuar con la preparación de la máquina virtual. En mi caso, usaré VirtualBox con una máquina virtual Windows 8.1 que se cuelga en mi disco.

NOTA: Para aquellos que no tienen una máquina virtual en Windows, es posible obtener una muy fácilmente. Como parte del proyecto Modern IE, Microsoft proporciona máquinas virtuales de Windows gratuitas, listas para usar.

La preparación de la máquina virtual es muy simple, solo:

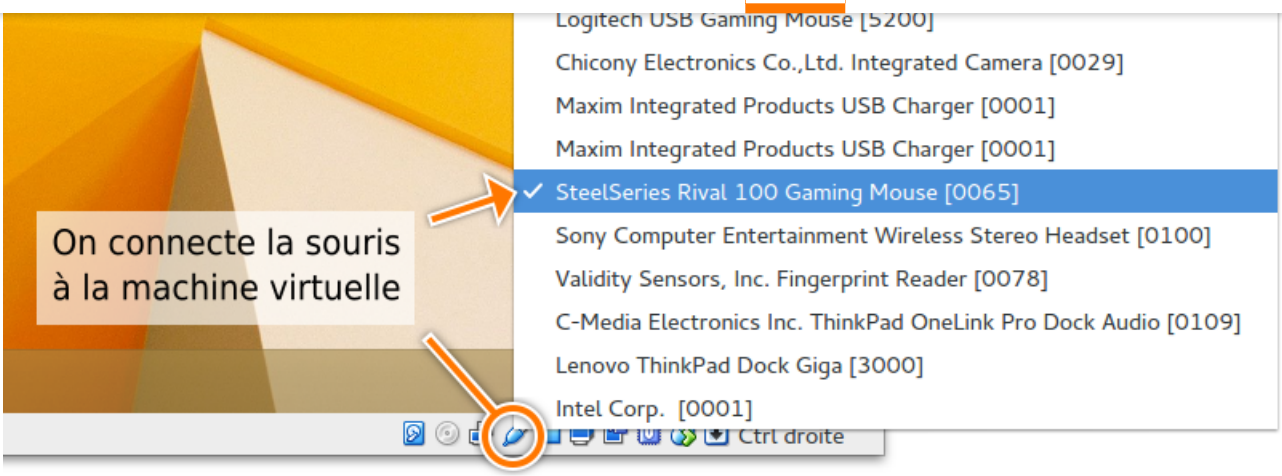
- descargar / instalar / iniciar la máquina virtual,
- descargue la versión del software SteelSeries Engine 3 adaptada a la versión de Windows instalada en la máquina virtual,
- e instalarlo en el Windows virtualizado.

Una vez hecho esto, todo lo que tienes que hacer es comprobar que todo funciona:

- conectamos el mouse a la máquina virtual usando el ícono en la parte inferior derecha de la ventana,
- lanzamos SteelSeries Engine 3
- y asegúrese de que reconozca bien el ratón.

Incluso se pueden probar algunos ajustes, normalmente todo debería ser funcional.

NOTA: Esto es cuando el segundo ratón se vuelve necesario. El Rival 100 ya no se puede usar una vez que está conectado a la máquina virtual.



Monitorización de bus USB utilizando Wireshark

En primer lugar, debe asegurarse de que el módulo usbmon del kernel esté completamente cargado. Esto permitirá a Wireshark recuperar los marcos que pasan en el bus USB. Podemos verificar que el módulo está cargado con el siguiente comando:

```
$ lsmod | grep usbmon
```

Si aparece una línea similar a esta, significa que el módulo está cargado:

```
usbmon 28672 0
```

de lo contrario, debe cargarlo usando el comando modprobe (en la raíz):

```
# modprobe usbmon
```

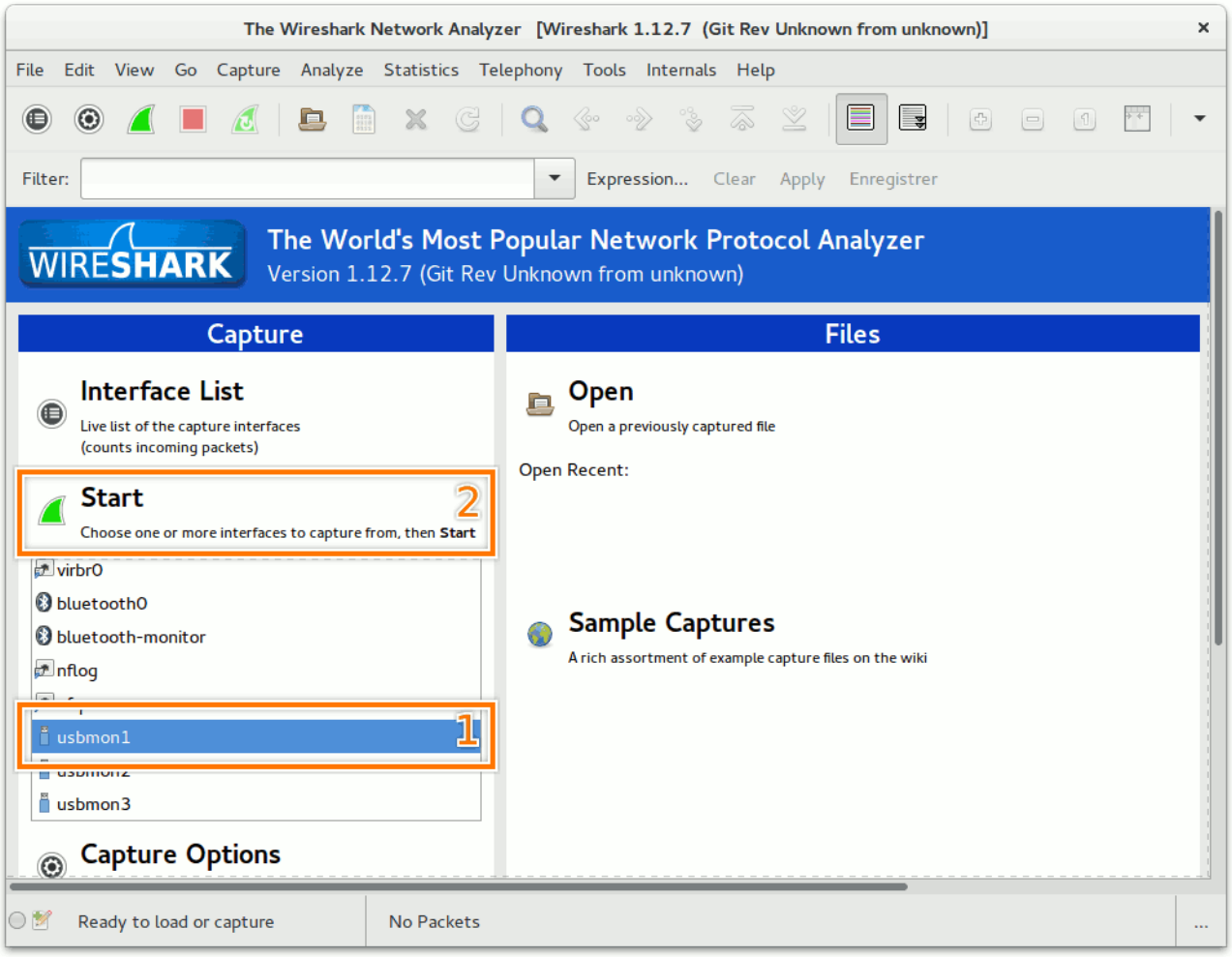
Una vez que se carga el módulo usbmon , solo queda iniciar Wireshark (en la raíz):

```
# wireshark
```

NOTA: No se recomienda ejecutar Wireshark como root. También harás una guirnalda al iniciarla si detecta que se lanzó en la raíz ... Dicho esto, no quiero ajustar sus permisos correctamente por ahora, así que voy a hazlo de todos modos

Para comenzar, debes seleccionar la interfaz para espiar. En este caso es `"usbmonX"` , donde `"X"` es el número del bus USB al que está conectado el mouse. En mi caso, está conectado al bus número 1, por lo que seleccionaré la interfaz `"usbmon1"` .

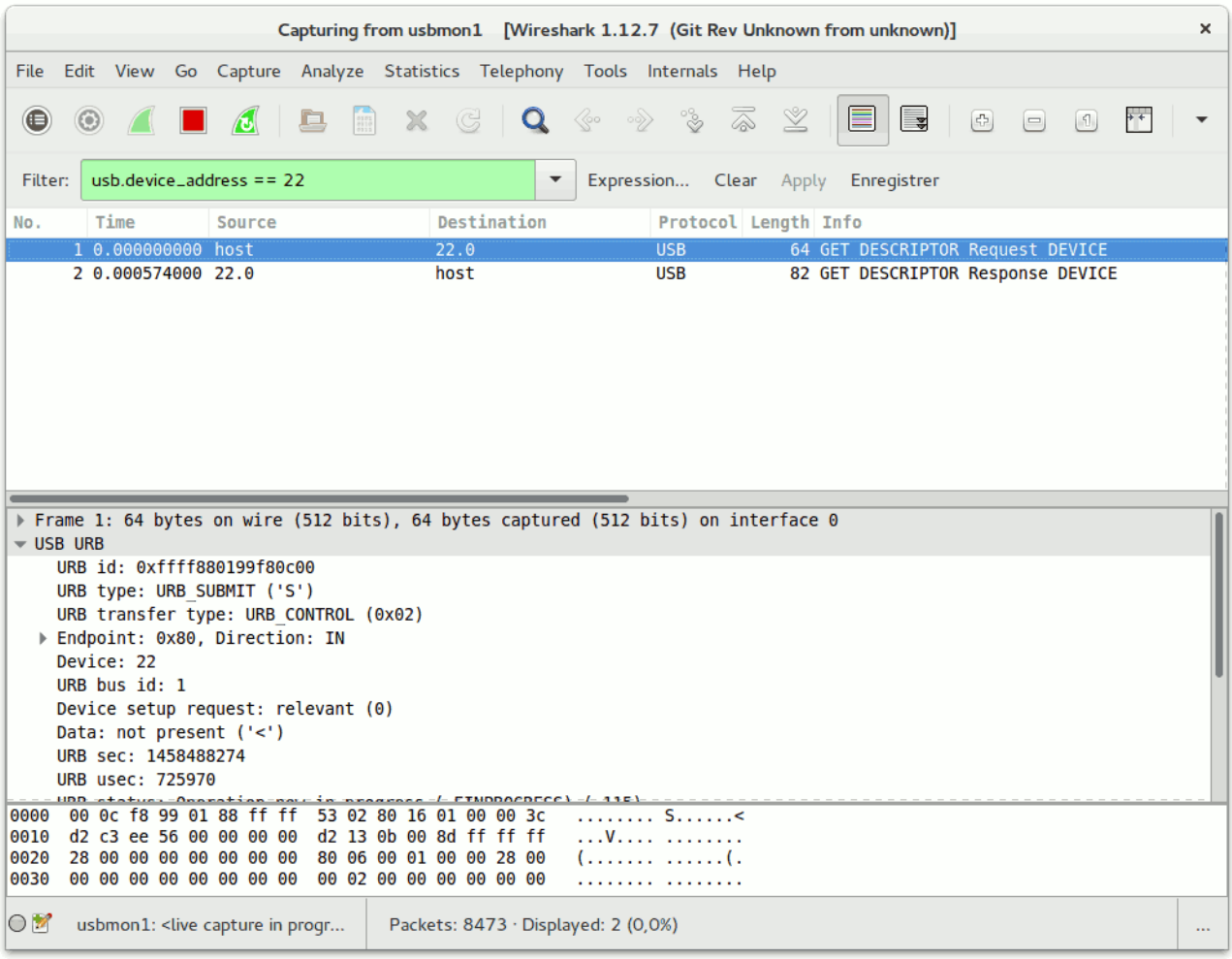
Il ne reste plus qu'à lancer la capture en cliquant sur le bouton `"Start"` (oui celui avec une icône verte en forme d'aileron de requin…).



Une fois la capture lancée, la fenêtre de Wireshark va très vite se remplir de plein de lignes : il s'agit de l'ensemble du trafic circulant sur le bus USB de la machine. Seule une infime partie de tout ce trafic nous intéresse, il va donc falloir filtrer un peu tout ça. Pour ce faire on va entrer le filtre suivant dans le champ `"Filter"` qui se trouve en haut de la fenêtre :

```
usb.device_address == XXX
```

où `"XXX"` est le numéro qui a été attribué au périphérique (dans mon cas il s'agit du numéro 22).



Analyse du protocole de la souris

Maintenant que tout est prêt, l'analyse va pouvoir débuter. Pour commencer on va jouer avec un réglage qui devrait être facile à interpréter : la couleur du rétro-éclairage de la souris.

Dans l'interface du SteelSeries Engine 3, sélectionnons une couleur dont le code hexadécimal soit facile à repérer, par exemple `"#112233"` ou `"#AABBCC"`… Dans mon cas je choisis `"#FF4411"` par ce que c'est un orange que j'aime bien (*no comment*)…

Une fois le réglage validé (en cliquant sur le bouton `"Fermer"` de la fenêtre), la souris change instantanément de couleur, et deux nouvelles lignes apparaissent dans Wireshask :

- La première provient de notre machine (`"host"`) et se dirige vers la première interface de la souris (`22.0`),
- et la seconde provient de la souris et est destinée au PC.

USBHID. Si ce n'est pas le cas, il faut déconnecter puis reconnecter la souris à la Machine virtuelle. Cela permettra à Wireshark de capturer les trames qui lui indiqueront qu'il a à faire à un périphérique de classe HID.

À priori celle qui devrait nous intéresser est la première, et en fouillant un peu dans les données de la trame, on finit par tomber sur la ligne suivante :

```
Data Fragment: 0500FF4411000000000000000000000...
```

On y retrouve très clairement la couleur que nous avions sélectionnée précédemment. Pour s'assurer qu'il ne s'agit pas là d'une coïncidence, il est possible de réessayer avec une autre couleur, par exemple `"#00FFFF"`.

Cette fois encore, la souris change de couleur (elle vire au bleu) et deux nouvelles trames apparaissent dans Wireshark. Dans la première d'entre elles, on retrouve une fois de plus notre couleur :

```
Data Fragment: 050000FFFF000000000000000000000...
```

À ce stade, on peut conclure qu'il existe une commande `0x0500` qui prend en paramètre les trois canaux (rouge, vert et bleu) de la couleur souhaitée.

À présent essayons de changer la couleur à la main. Vous vous souvenez des fichiers que nous avions trouvés dans `/dev` au tout début de nos investigations ? Et bah c'est maintenant qu'ils vont nous être utiles.

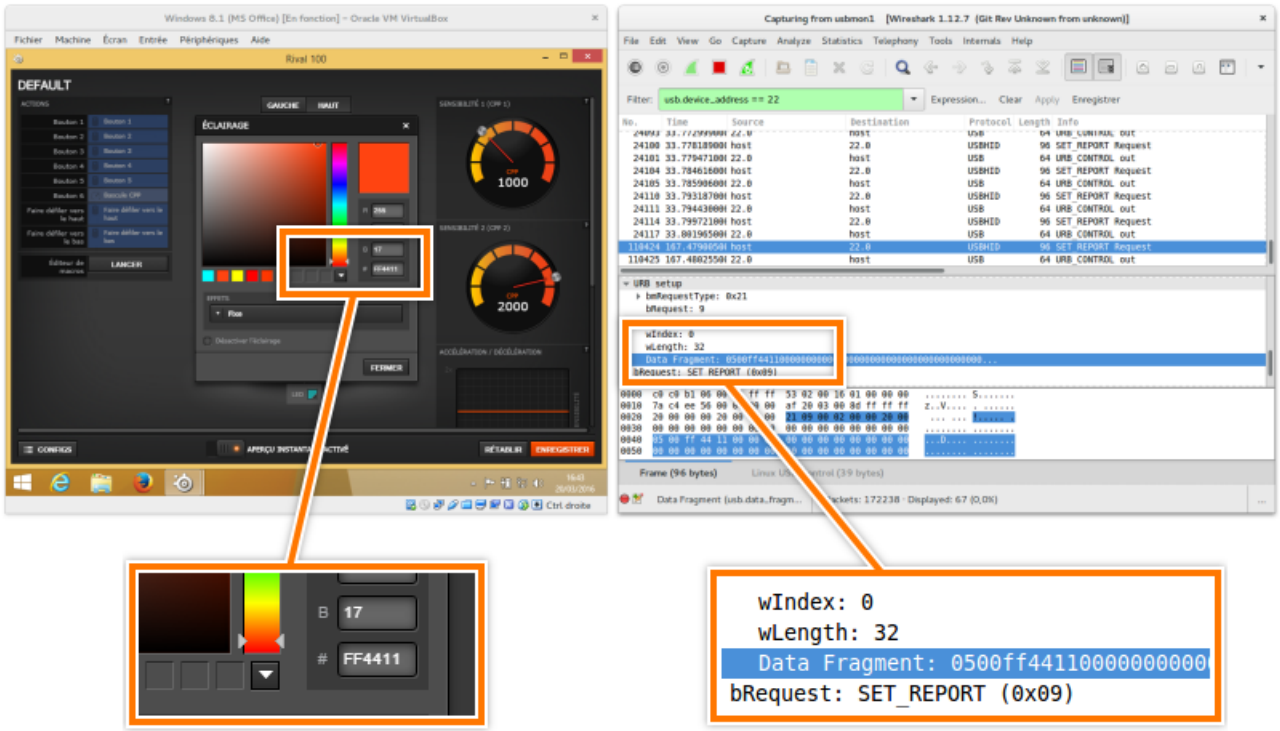
- Pour commencer, il faut ouvrir un terminal root (si vous êtes sous Ubuntu, vous pouvez utiliser la commande `"sudo su"`).
- Ensuite il suffit d'écrire les octets souhaités directement dans le premier des deux fichiers trouvés précédemment (c'est à dire `/dev/hidraw6` dans mon cas). Par exemple, essayons d'allumer la souris en rouge (`"#FF0000"`) :

```
echo -en "\x05\x00\xFF\x00\x00" > /dev/hidraw6"
```

NOTE : Il est possible que dans certains cas il faille répéter la commande précédente deux fois pour que cela fonctionne car pour une raison que j'ignore, les octets peuvent ne pas être directement écrits dans le fichier (ils restent dans le buffer d'écriture en attendant d'être effectivement écrits).

fichier (voire dans `/dev/hidraw0` dans certains cas…).

Il ne reste plus qu'à répéter les opérations pour chacun des réglages présents dans l'utilitaire de configuration de SteelSeries pour trouver toutes les commandes existantes.



Liste des commandes

Ci-dessous, un tableau récapitulatif des commandes que j'ai été en mesure de découvrir :

Nom	Commande	Paramètres	Descriptions
set_sensitiviry	`0x03`	`<preset> <value>`	Définit la sensibilité du capteur
set_polling_rate	`0x04`	`0x00 <rate>`	Définit la fréquence d'interrogation
set_color	`0x05`	`0x00 <red> <green> <blue>`	Change la couleur de la souris
set_light_effect	`0x07`	`0x00 <effect>`	Définit les effets lumineux (pulsation…)
save	`0x09`	-	Sauvegarde les paramètres dans la souris
set_btn6_action	`0x0B`	`<action>`	Définit l'action du bouton sous la molette

			souris, action indéterminée

set_sensitivity

La souris est capable de contenir deux pré-réglages de sensibilité que l'on peut alterner via le bouton situé sous la molette de la souris.

```
0x03 <preset> <value>
```

Paramètres :

- `preset` : Numéro du pré-réglage à modifier
  - `0x01` : pré-réglage numéro 1
  - `0x02` : pré-réglage numéro 2
- `value` : Sensibilité du capteur parmi les valeurs suivantes :
  - `0x08` : 250 DPI
  - `0x07` : 500 DPI
  - `0x06` : 1000 DPI (valeur par défaut du pré-réglage numéro 1)
  - `0x05` : 1250 DPI
  - `0x04` : 1500 DPI
  - `0x03` : 1750 DPI
  - `0x02` : 2000 DPI (valeur par défaut du pré-réglage numéro 2)
  - `0x01` : 4000 DPI

set_polling_rate

Définit la fréquence à laquelle la position de la souris sera lue par l'ordinateur. Plus cette valeur est élevée, plus la précision sera grande. Cependant, une valeur élevée signifie également plus de trafic sur le bus USB et une plus grande utilisation du CPU.

```
0x04 0x00 <rate>
```

Paramètres :

- `0x00` : le premier paramètre n'est pas utilisé sur ce modèle et doit être défini à `0x00`
- `rate` : le taux de rafraichissement parmi les valeurs suivantes :
  - `0x04` : 125 Hz (8 ms)

- `0x01` : 1000 Hz (1 ms, valeur par défaut)

set_color

Définit la couleur du rétro-éclairage de la souris.

```
0x05 0x00 <red> <green> <blue>
```

Paramètres :

- `0x00` : le premier paramètre n'est pas utilisé sur ce modèle et doit être défini à `0x00`
- `red`: valeur du canal rouge de la couleur RGB (compris entre `0x00` et `0xFF`)
- `green`: valeur du canal vert de la couleur RGB (compris entre `0x00` et `0xFF`)
- `blue`: valeur du canal bleu de la couleur RGB (compris entre `0x00` et `0xFF`)

set_light_effect

Configure un effet de lumière.

```
0x07 0x00 <effect>
```

Paramètres :

- `0x00` : le premier paramètre n'est pas utilisé sur ce modèle et doit être défini à `0x00`
- `effect` : l'effet à appliquer :
  - `0x01` : statique
  - `0x02` : pulsation lente
  - `0x03` : pulsation moyenne (appelée « respiration » dans le SteelSeries Engine 3)
  - `0x04` : pulsation rapide

save

Sauvegarde les réglages actuels dans la mémoire interne de la souris. Si cette commande n'est pas appelée après avoir modifié les paramètres de la souris, celle-ci retournera à sa précédente configuration la prochaine fois qu'elle sera reconnectée à la machine.

```
0x09
```

set_btn6_action

Définit l'action que le bouton numéro 6 (numéroté comme tel dans SteelSeries Engine 3) doit effectuer.

```
0x0B <action>
```

Paramètres :

- `action` : action du bouton :
  - `0x00` : permet le basculement entre les pré-réglages de sensibilité du capteur (valeur par défaut)
  - `0x01` : rend le bouton accessible par le système d'exploitation (reconnu en tant que `"bouton 10"` par `xev`)

Conclusion

Au final, le reverse engineering de cette souris s'est avéré beaucoup plus simple que je ne l'aurais pensé. Et maintenant que je suis en possession de toutes les informations nécessaires, je devrais être en mesure de développer assez rapidement un petit utilitaire en ligne de commande pour configurer ma souris (je le placerais sur Github lorsque je l'aurai terminé).

EDIT 16/04/2016: J'ai écrit le petit utilitaire en python, et comme promis il est disponible sur Github : github.com/flozz/rivalcfg. Si vous souhaitez étudier le code du logiciel, je vous recommande de jeter un coup d'œil à la version 1.0.1 qui est la plus simple à comprendre.