Développement d'un cheval de Troie

Professeur responsable :	Gérald Litzistorf
En collaboration avec :	Cédric Renouard, Ilion Security
Candidat :	Jean-Marc Solleder

Plan de l'exposé

- Généralités :
  - Généralités sur les chevaux de Troie
  - Particularités de ce projet
- Démonstration
- Partie technique :
  - Transmission des événements
  - Compression des flux de données
  - Diminution de la taille d'un exécutable
- Conclusion et questions

Généralités

- Pas d’autoreproduction
- Nécessite une installation
  - Social Engineering
  - Failles

Généralités

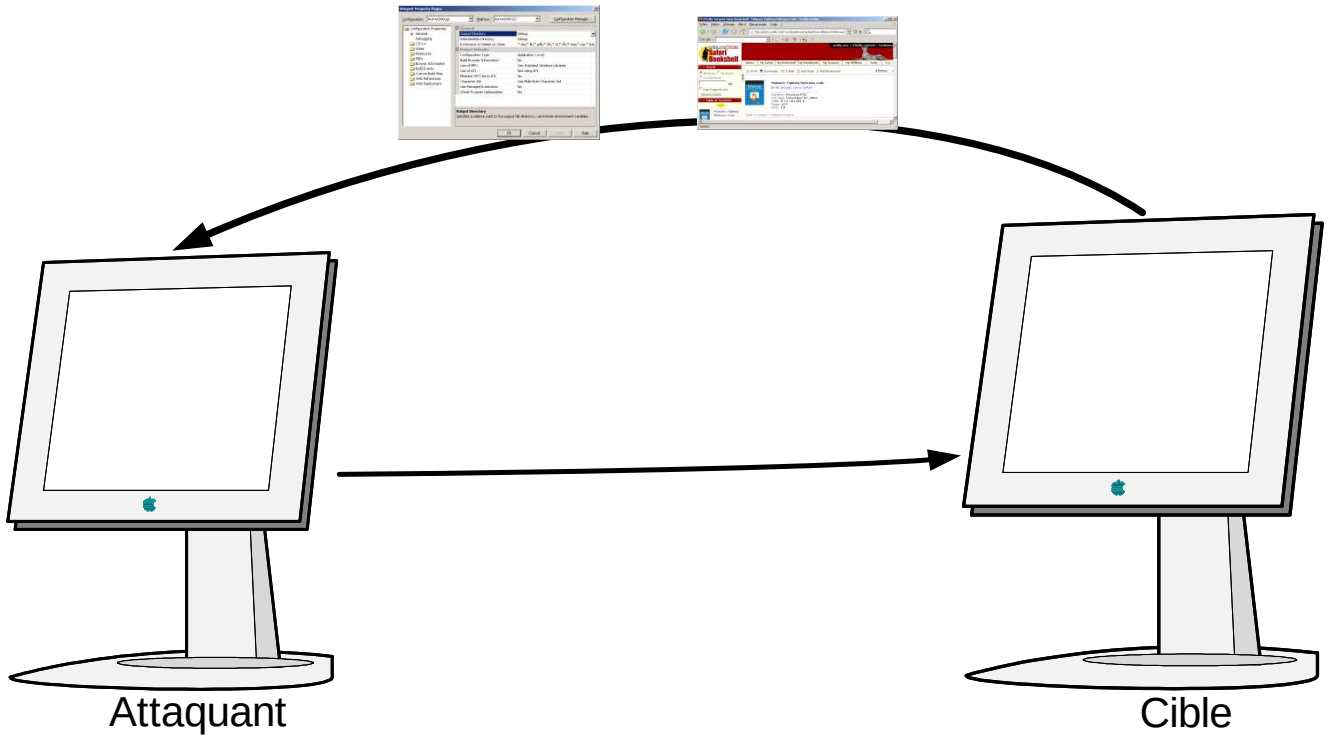
- Un cheval de Troie doit s'exécuter à l'insu de l'utilisateur de l'ordinateur

- Le programme ne doit pas être « bruyant »
  - Peu d'activités réseau
  - Utilisation modérée des ressources du système (mémoire, processeur, espace disque,…)

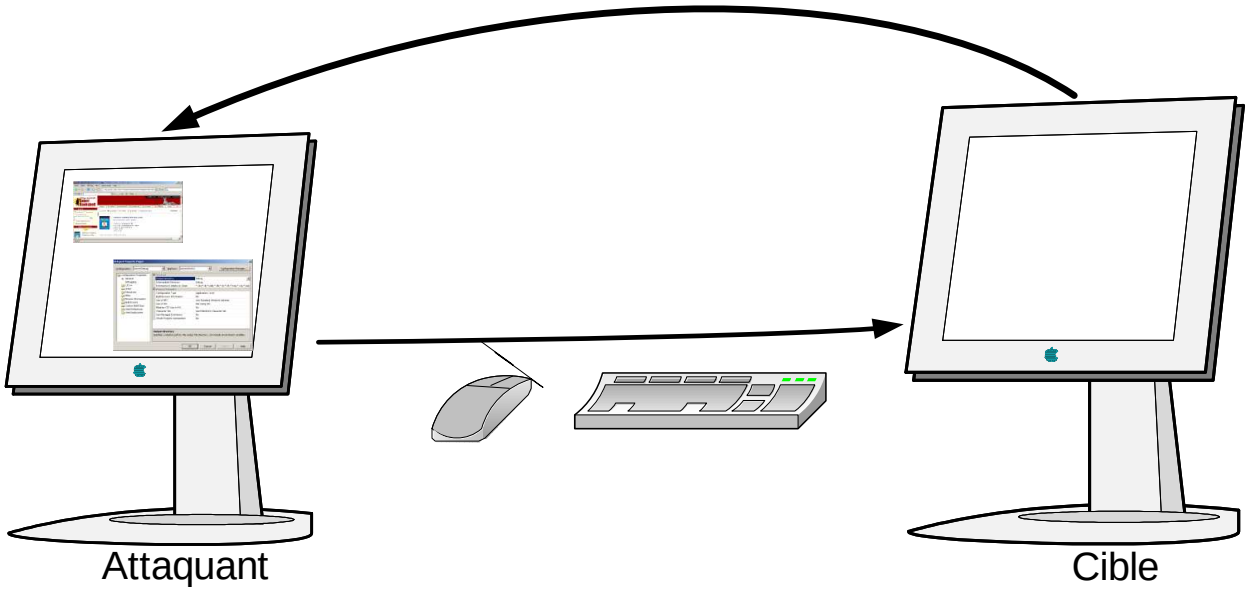
Particularités

- Cahier des charges :
  - Relais des interfaces graphiques et des événements
  - Diminution des flux de données entre les deux machines
  - Exécution en mode utilisateur

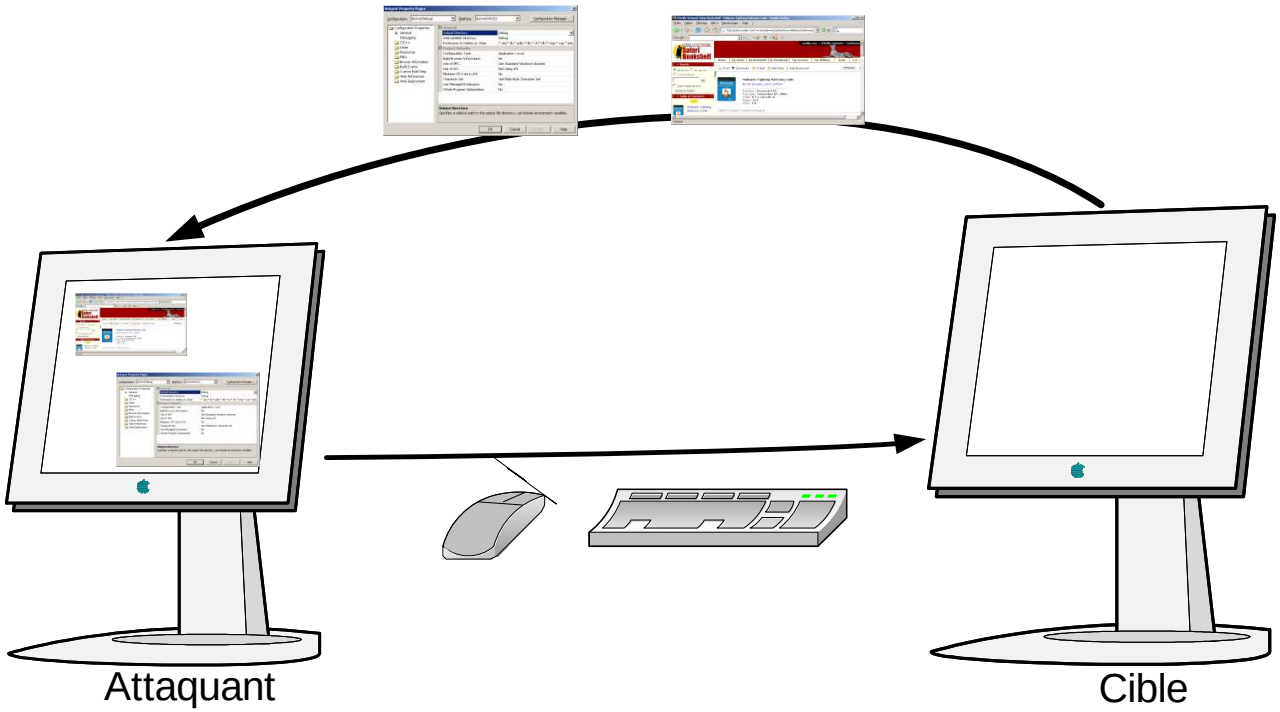
Particularités



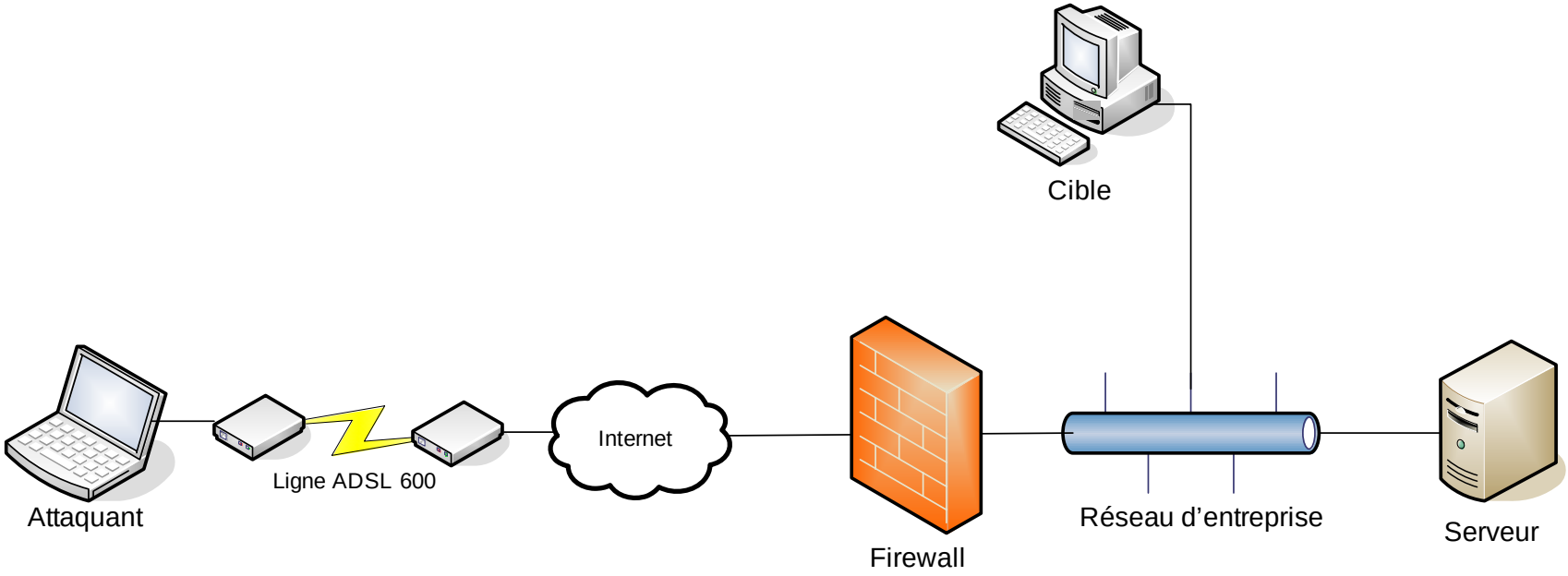
Particularités



Particularités



Démonstration



Partie technique

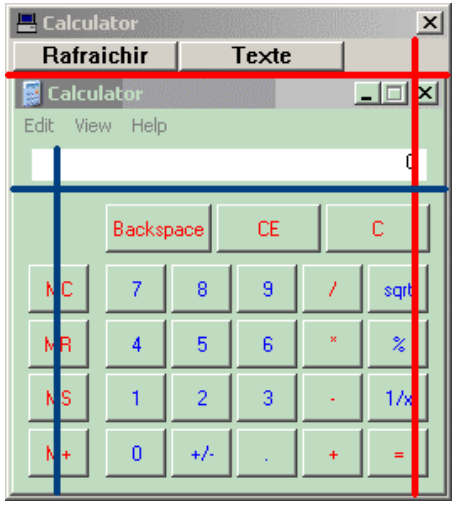
- Problématiques :
  - Capture des interfaces graphiques
  - Envoi des événements clavier / souris
  - Compression des flux de donnée
  - Diminution de la taille de l’exécutable

Événements

- Système d’événement
  - Grande quantité de messages
  - Commandes et notifications

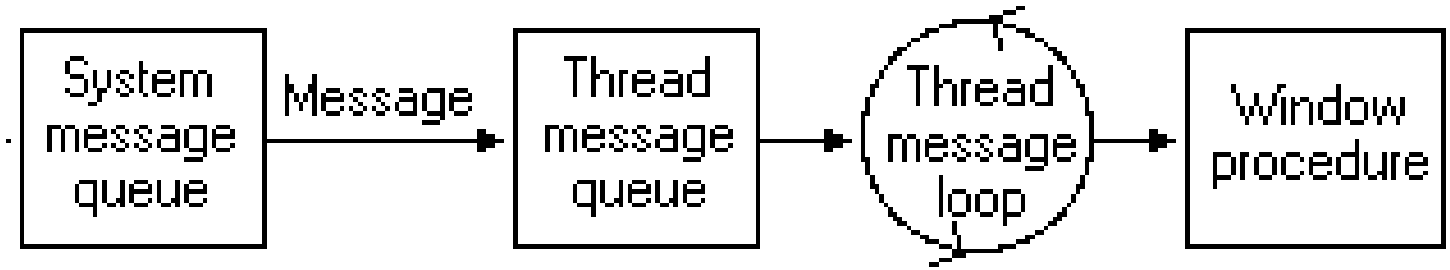
Événements

- Souris :
  - Cliques et double cliques
  - Problème de coordonnées :



Événements

- Clavier :
  - Plusieurs niveaux :



Événements

- Coordonnées :

Événements

- Il reste des problèmes :
  - Certaines applications ne répondent pas correctement (Mozilla)

Compression

- Format bitmap :
  - Simple tableau de pixels
  - Taille d’un pixel dépend du nombre de bits par pixels (bpp)

Compression

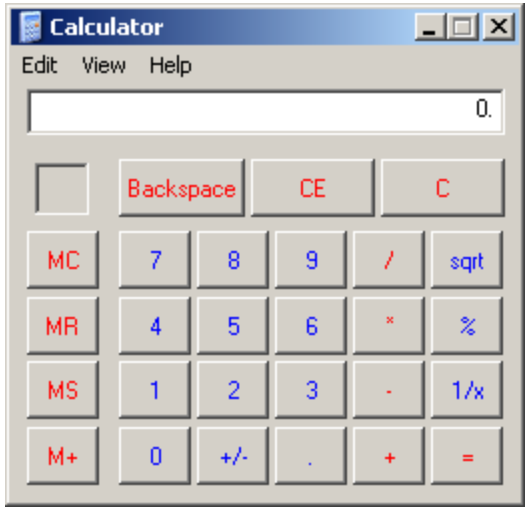
- Problématique :

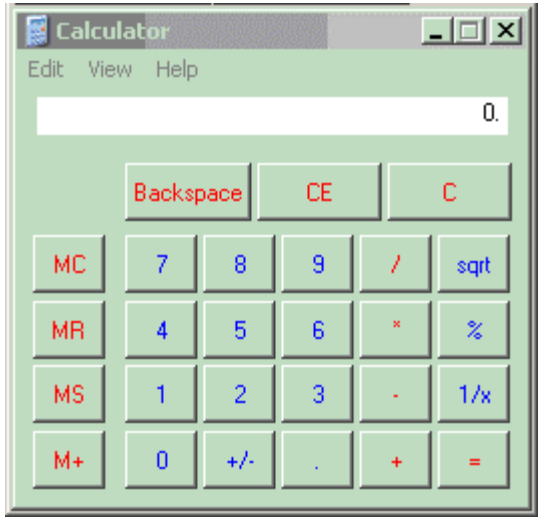
$$\frac{1024 \cdot 768 \cdot 32}{8} \cong 3{,}1[Mo]$$

Ce qui est évidemment inacceptable !!

Compression

- Diminution du nombre de bits par pixel à 8 bpp :





Compression

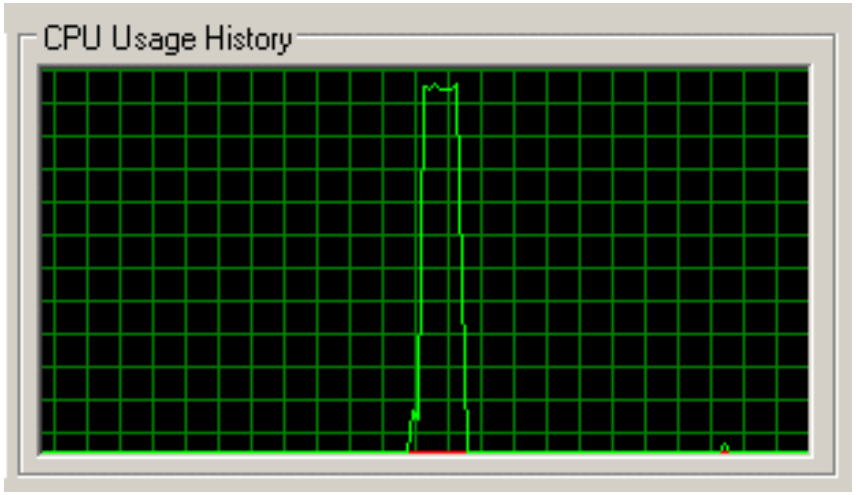
- Librairies LibJpeg et Zlib
  - Utilisées par VNC
  - Pas assez compacte
  - Utilisation de *dll*
- Impossible à utiliser pour ce projet !

Compression

- LZ77
  - Développé en 1977 par Abraham Lempel et Jackob Ziv
  - Forts taux de compression (jusqu'à moins de 5%)
  - Compression par substitution

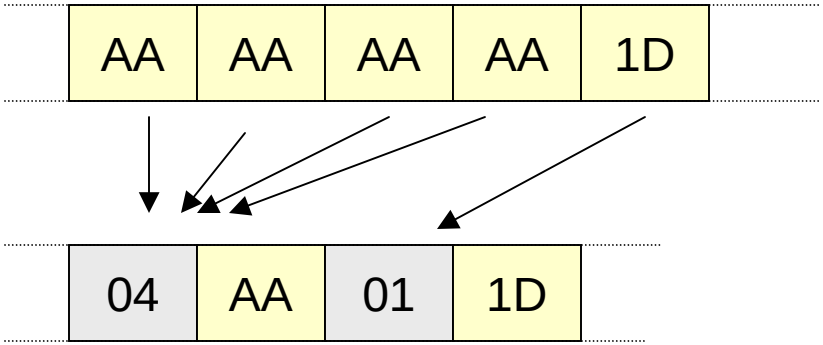
Compression

- Malheureusement LZ77 est optimisé pour la décompression
- Compression lente et gourmande en CPU :



Compression

- La compression RLE (*Run Length Encoding*)
  - Codage des répétitions :



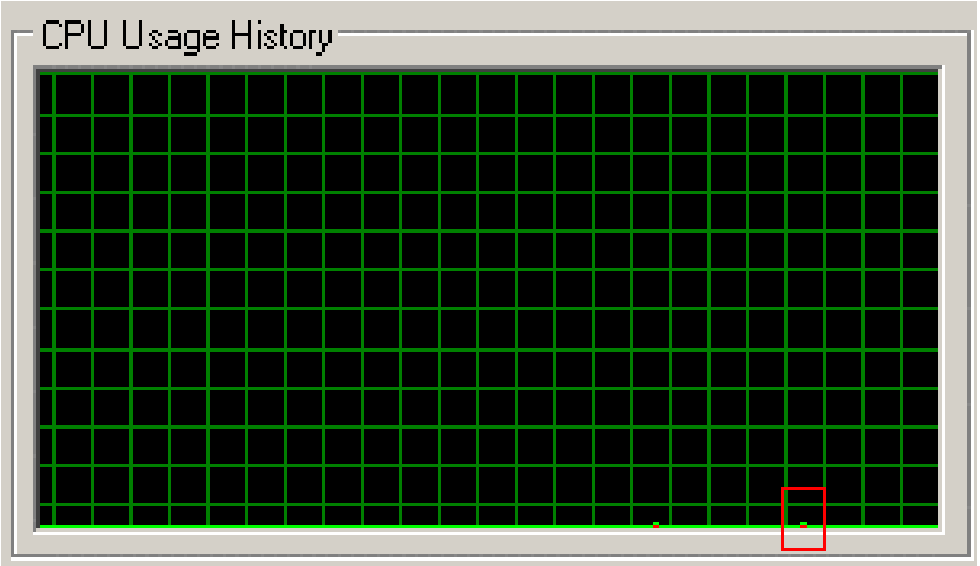
Compression

- Le résultat de ce codage dépend des valeurs (nombre d'octet) que nous prenons pour les données et les *run length*

- Le logiciel cible permet de tester différentes longueurs de données et de *run length* et de choisir celle qui compresse le plus

Compression

- Le RLE consomme très peu de ressources processeur :



Compression

- Le taux de compression du RLE dépend très fortement du nombre de répétition dans l'image
- Les images d'une interface graphique varient généralement peu dans le temps

Compression

- Codage différentiel :

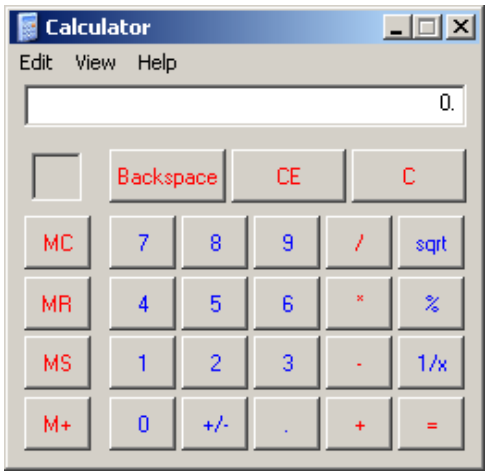


Image 1

⊗

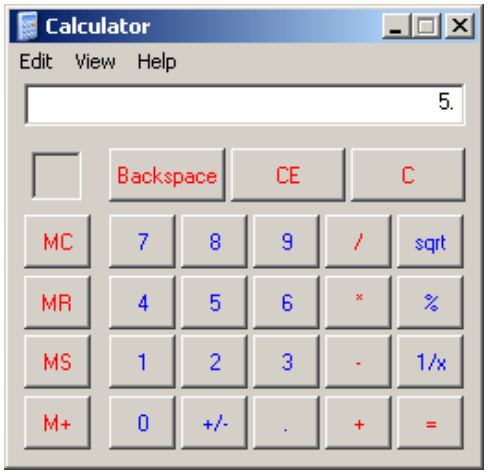


Image 2

=

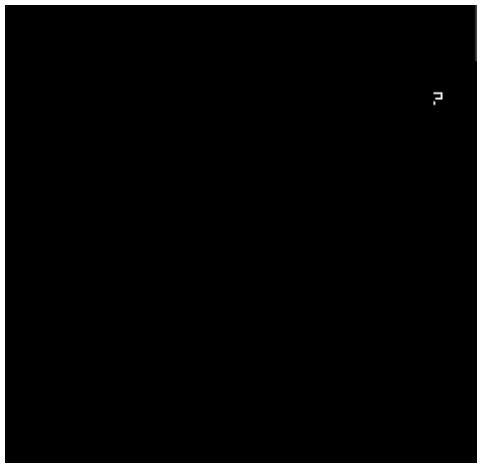


Image différentielle

Compression

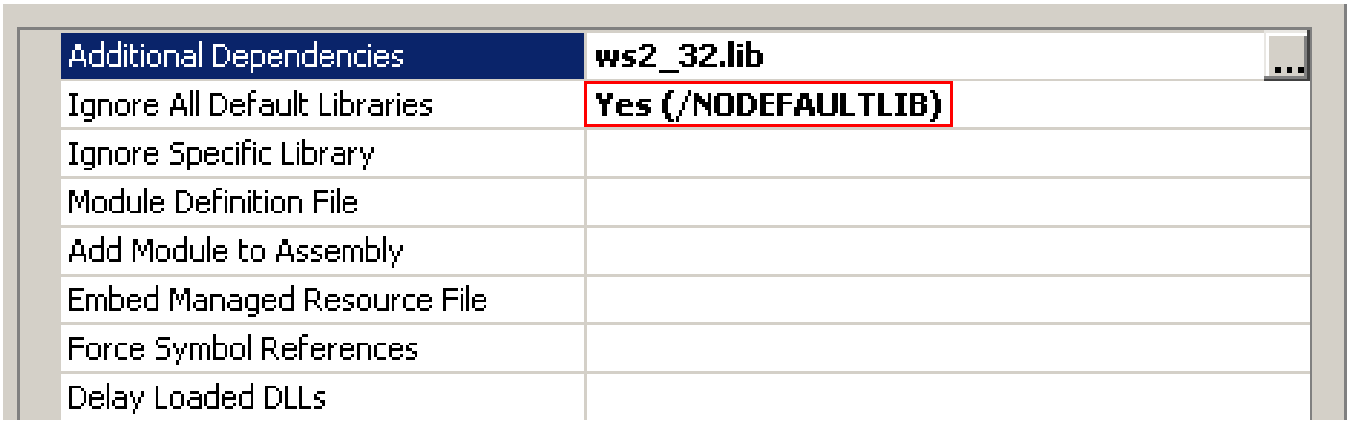
Application	Taille première image	Taille images différentielles
cmd.exe	9,928%	Aucune modification : 0,007%
		Modifications faibles : 0,1%
		Fortes modifications : 9,621%
calc.exe	20,230%	<1%

Exécutable

- Facteur taille très important pour le programme cible
- Paramètres du compilateurs standards (retrait des informations de *debug*, optimisation de la taille …)
- Taille environ 40 Ko

Exécutable

- Suppression de librairies standard :



Exécutable

- Point d'entrée d'une application Windows :

```
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,
                   int nCmdShow)

{
        //… corps du programme …
}
```

Exécutable

- WinMain() n’est pas le vrai point d’entrée
- WinMainCRTStartup() initialise les paramètres de WinMain()
- WinMainCRTStartup() est initialisée par les librairies par défaut
- Il a donc fallu la réécrire

Exécutable

- Autres problèmes :
  - Fonctions de gestions de la mémoire et de gestion de chaînes ne sont plus disponibles
  - Memcpy,memmove et memset ne sont plus disponibles
  - Les « security cookies » doivent être désactivés.

Exécutable

- Gain important :
  - Taille de l’exécutable cible passée de plus de 120ko avec les informations de debug à environ 12ko à la fin de la réduction

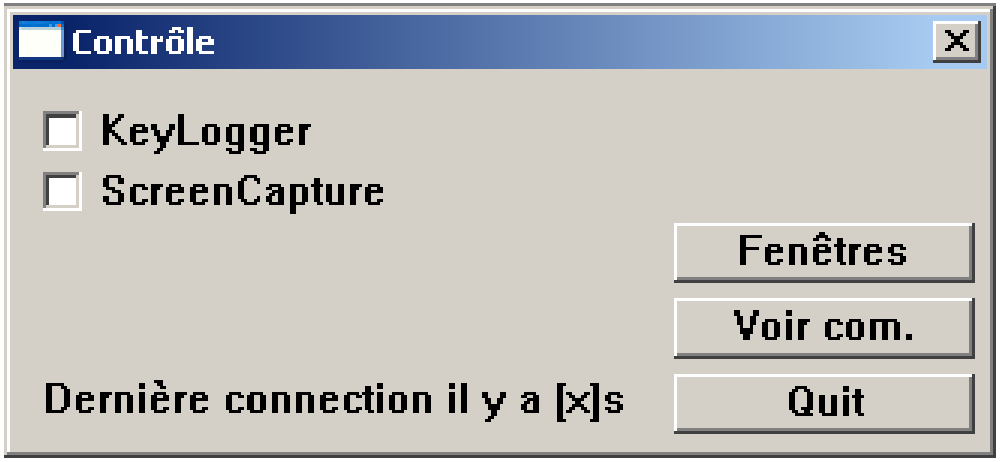
Conclusion

été
01
02
03
04
05
06
07
08
09
10
11
12

- Cahier des charges
- Prise en main de Visual Studio et du platform SDK
- Phase d’analyse

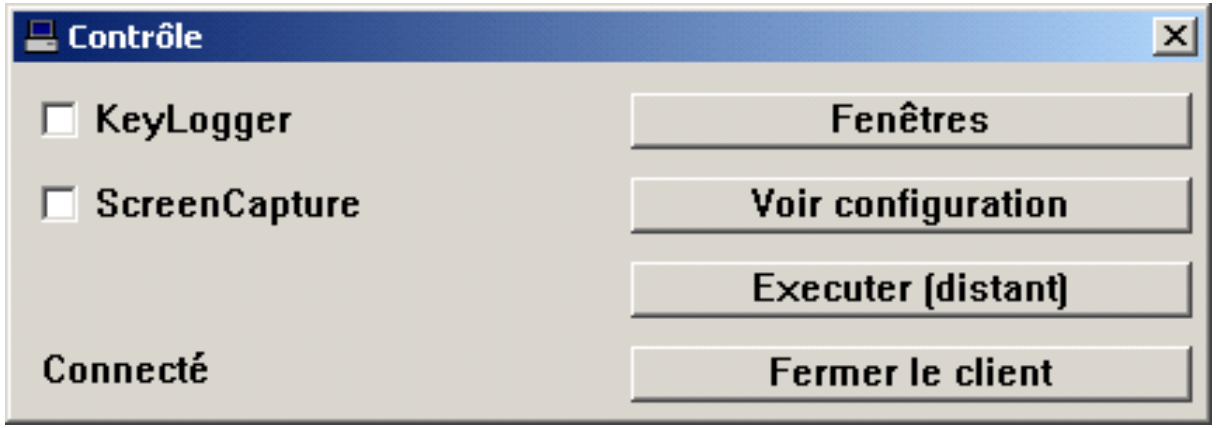
Conclusion

- Développement de la première version



Conclusion

- Développement de la version 2



Conclusion

été
01
02
03
04
05
06
07
08
09
10
11
12

- Debug
- Rapport
- Présentation



Questions