The Metabrik Platform
Développement Rapide d'Outils de Sécurité Réutilisables

Patrice Auffret

metabrik.org

2 Juin 2016

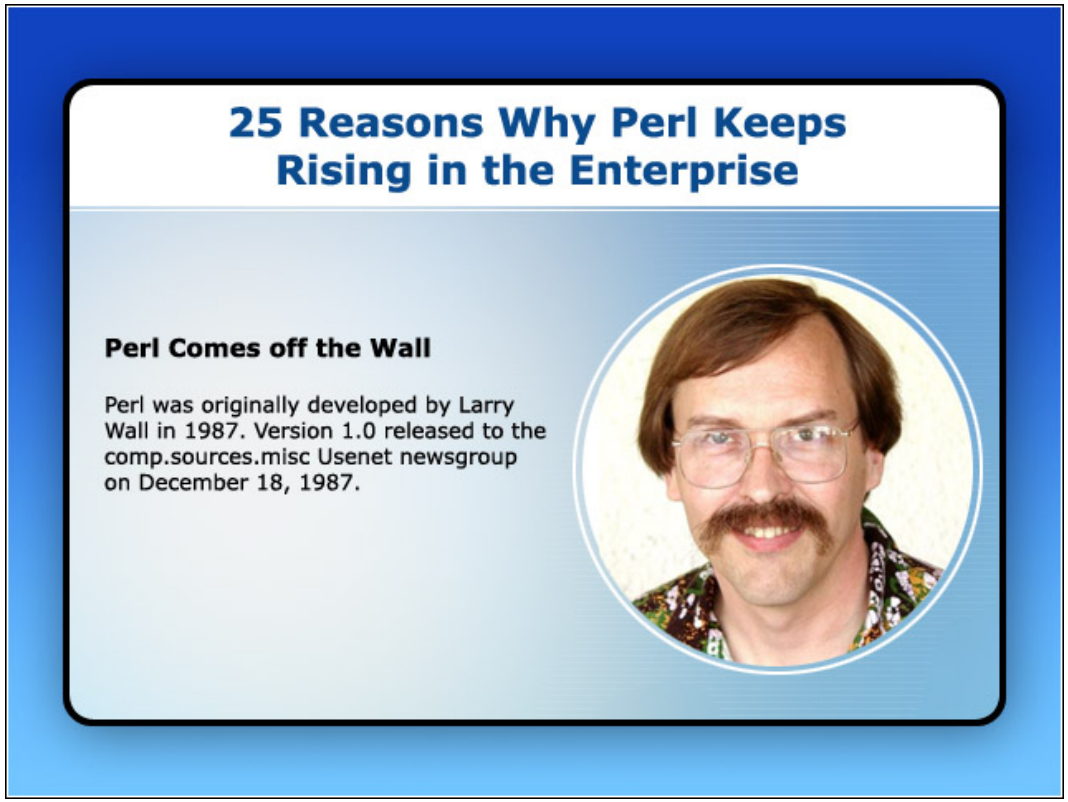
Qu'est-ce que Metabrik ?

Une plateforme

- Un shell type *UNIX*
- Un vrai langage avec *Read-Eval-Print-Loop*
- Des *Briks*
  - Une plateforme de développement
  - Le bon outil rapidement

Un vrai language ?

Oui.



- Source : eweek, 30 Avril 2010

Pourquoi ?

CLI vaincra

- Tout en ligne de commande
  - Tout automatiser
- *Do it once*
  - Scripts jetables rébarbatifs
  - Réutilisabilité du code
- Shell UNIX trop simples
  - *Pipe* trop limité
  - Intégration avec un vrai language
- Développement rapide en *CLI*
  - Permettant d'écrire un script
- Syntaxe normalisée *human readable*

Comparaison avec REbus

L'outil ne doit pas remplacer l'humain

- Metabrik : pas d'automatisation entrées/sorties
  - utilisation interactive contre automatique
  - les outils aident mais ne remplacent pas l'humain
- REbus : pas un shell
  - une syntaxe pas évidente
- Les deux utilisent des *wrapper*s d'outils existant
  - Metabrik a également beaucoup de *Brik*s "propres"

Démo - The Metabrik Shell

- 3 types de lignes de commandes
  - externes
  - commandes de *Briks*
  - code *Perl* (*REPL*)
- 5 commandes de *Briks*
  - *use, set, get, run, help*

Fonctionnalités

Les plus notables

- Shell : builtins
  - cd, alias, ...
- Shell personnalisable avec historique
  - Fichier *.metabrik_rc*
  - Fichier *.metabrik_history*
- Une syntaxe à la Metasploit
- Complétion des commandes, fichiers, variables
- Gestion simple des dépendances
  - *run brik : :tool install network : :sinfp3*

Les *Briks*

Vous avez la connaissance, le détail est dans la *Brik*

- Souvent des programmes externes (*wrapper*)
  - mais pas uniquement
- Orienté objet
  - une *Brik* peut hériter d'une ou plusieurs autres
- Ajout de fonctions aux outils existants
  - Il en manque toujours une
  - Exemple : *scalpel*
- Un module *Perl* réutilisable
  - une interface *CLI*
  - une interface *classique*
- La glue entre les *Briks* : les variables
  - gestion des entrées/sorties
  - le nouveau *pipe*

Démo - automatiser l'analyse de malware

Ou comment extraire les *Indicators of Compromise*

- Instrumentaliser une *VM* et prendre un snapshot
  - *system : :virtualbox*
- Exécuter des programmes à distance
  - *remote : :winexe*
  - *remote : :wmi*
- Faire un diff de l'état d'une machine *Windows*
  - *forensic : :volatility*

Démo - challenge inforensique

Ou comment résoudre rapidement un problème

- Analyse de fichiers
  - *file : :type*
  - *file : :compress*
  - *image : :exif*
- Extraire des données
  - *forensic : :scalpel*

Un Metatool

Industrialiser le prototype

- Prototype finalisé
  - run brik : :tool create_tool iplocation.pl
- Convertion des commandes du Shell en code

```
1  # Shell Metabrik
2  use lookup::iplocation
3  run lookup::iplocation from_ip 93.184.216.34
4
5  # Programme Perl
6  use Metabrik::Lookup::Iplocation;
7  my $li = Metabrik::Lookup::Iplocation->new_from_brik_init($con);
8  my $h = $li->from_ip($ip);
```

Conclusion

- *Production-ready*
- Plus de 200 *Brik*s ...
- Blog : https ://www.metabrik.org/
- Code : http ://trac.metabrik.org/browser
- Twitter : @Metabrik

Question(s) ?

Metabrik

There is a Brik for that.

- https ://www.metabrik.org/install/
- @Metabrik
- @PatriceAuffret