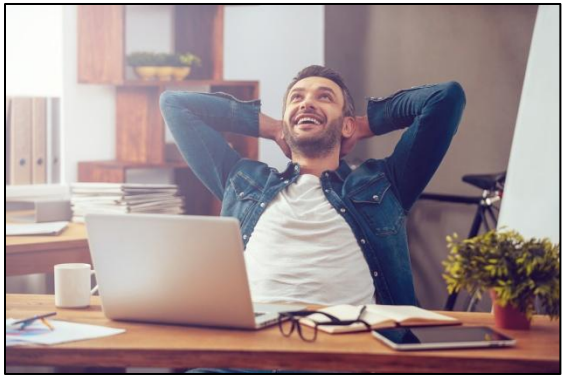
SourceFu

utilisation de l'interprétation partielle pour la désobfuscation de sources

Zilio Nicolas, @Big5_sec

Genèse



```
./obfuscatemax -i to_obfuscate/* -o obfuscated
```

```
rm -rf to_obfuscate
```

Constat

- Outils à base de Regex + formatage
  - Faciles à mettre en œuvre
  - Performances pas forcément au rendez-vous
- Outils de type Sandbox/Interprétation instrumentée
  - Potentiellement très efficace
  - on perd l'accès au source
  - Potentiellement auto-pown + anti-analyse
- Méthode manuelle
  - Long et fastidieux mais 100% efficace

Objectifs

- « Framework » de désobfuscation de sources
  - IDE de désobfuscation
  - Intégration avec d'autres outils d'analystes
  - Garder l'accès au source

- support des langages les plus communs
  - Powershell, Javascript, VB*

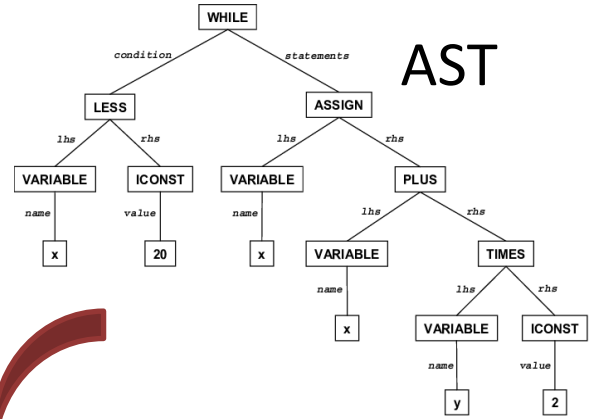
Une autre approche

- Approche intermédiaire entre la regex et la sandbox
  - Et si on interprétait que certaines parties de code?
  - Globalement, il suffirait d'optimiser le code pour le « désobfusquer »
    - Simplification des expressions
    - Evaluation des parties constantes

Idée de base

Source obfusquée

parsing
-> ANTLR



AST

Optimisations
(réduction AST)
-> ANTLR

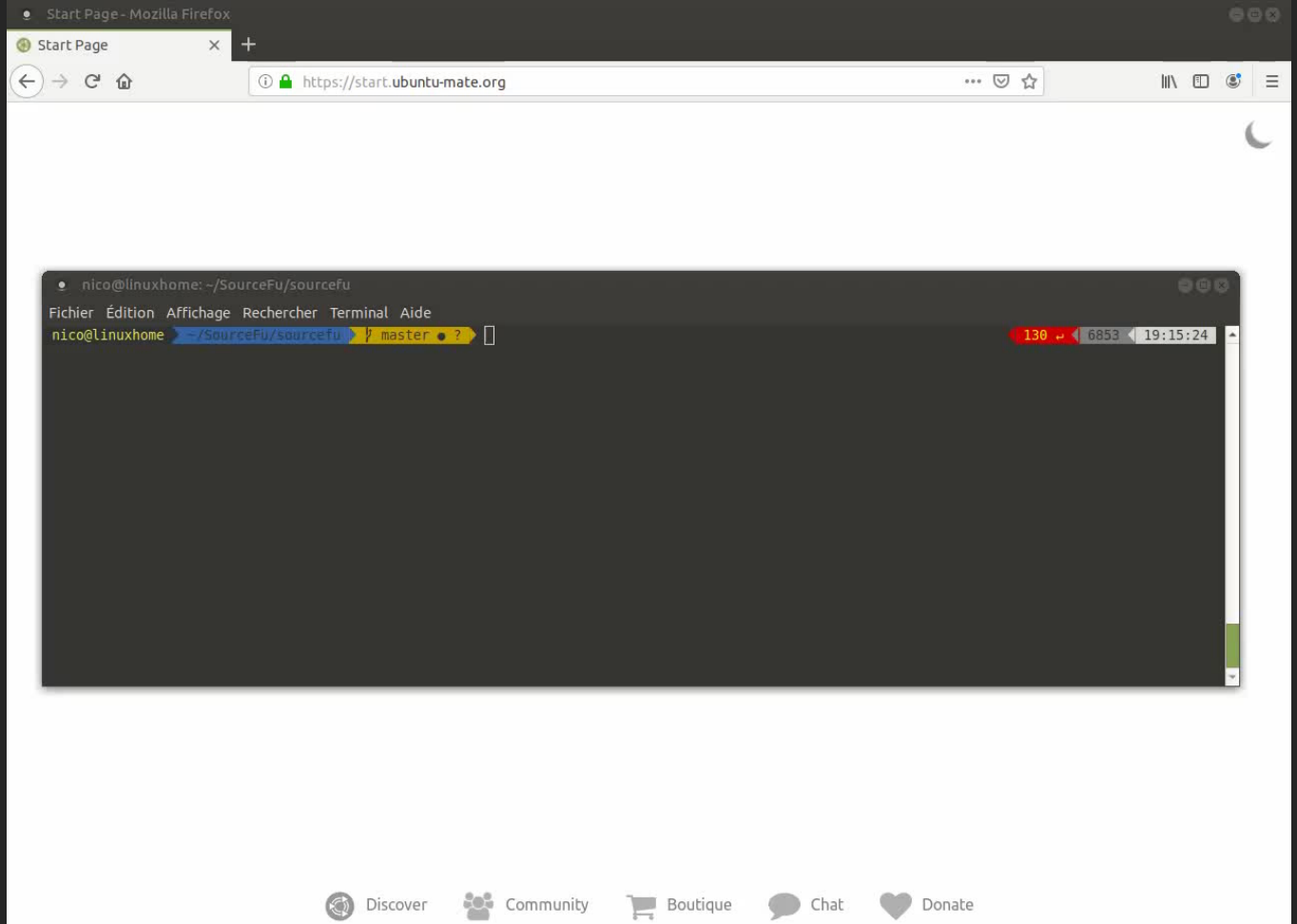
réécriture

Source « désobfusquée »

Mais ça existe déjà!

- L'approche en elle-même n'est pas nouvelle
  - CCC 2010 : *Code Deobfuscation by Optimization*, B. Spasojevic
  - *Automatic Binary Deobfuscation,* Journal in Computer Virology, Y. Guillot and A. Gazet

- Concernant le source, des outils utilisant ce principe ont même été publiés
  - ~~*JsDetox* by @svent_t~~
  - ~~*Jstillery* by OpenMinded~~


Start Page - Mozilla Firefox
Start Page
https://start.ubuntu-mate.org
nico@linuxhome: ~/SourceFu/sourcefu
Fichier Édition Affichage Rechercher Terminal Aide
nico@linuxhome ~/SourceFu/sourcefu master ● ?
130 6853 19:15:24
Discover
Community
Boutique
Chat
Donate

Sous le capot

Suppression commentaires
Remplacement constantes du langage

Optimisations 1-Passe

Tant que (une des optimisations de la boucle retourne un changement dans le code :

| optimisations multi-passes

Suppression code mort
Evaluation des expressions
Propagation des constantes
Simplification CFG

Nettoyage

Renommage symboles
Beautify

Sous le capot - 1

- interprétation des expressions
  - toute grammaire contient un truc du genre :

```
singleExpression
    : Function Identifier? '(' formalParameterList? ')' '{' functionBody '}' # FunctionExpression
    | Class Identifier? classTail                                             # ClassExpression
    | singleExpression '[' expressionSequence ']'                             # MemberIndexExpression
    | singleExpression '.' identifierName                                     # MemberDotExpression
    | singleExpression arguments                                              # ArgumentsExpression
    | New singleExpression arguments?                                         # NewExpression
    | singleExpression {this.notLineTerminator()}? '++'                       # PostIncrementExpression
    | singleExpression {this.notLineTerminator()}? '--'                       # PostDecreaseExpression
    | Delete singleExpression                                                 # DeleteExpression
    | Void singleExpression                                                   # VoidExpression
    | Typeof singleExpression                                                 # TypeofExpression
    | '++' singleExpression                                                   # PreIncrementExpression
    | '--' singleExpression                                                   # PreDecreaseExpression
    | '+' singleExpression                                                    # UnaryPlusExpression
    | '-' singleExpression                                                    # UnaryMinusExpression
    | '~' singleExpression                                                    # BitNotExpression
    | '!' singleExpression                                                    # NotExpression
    | singleExpression ('*' | '/' | '%') singleExpression                     # MultiplicativeExpression
    | singleExpression ('+' | '-') singleExpression                           # AdditiveExpression
```

  - ➢ Remplacer un mix de sous expressions par l'expression finale simplifiée

Sous le capot - 2

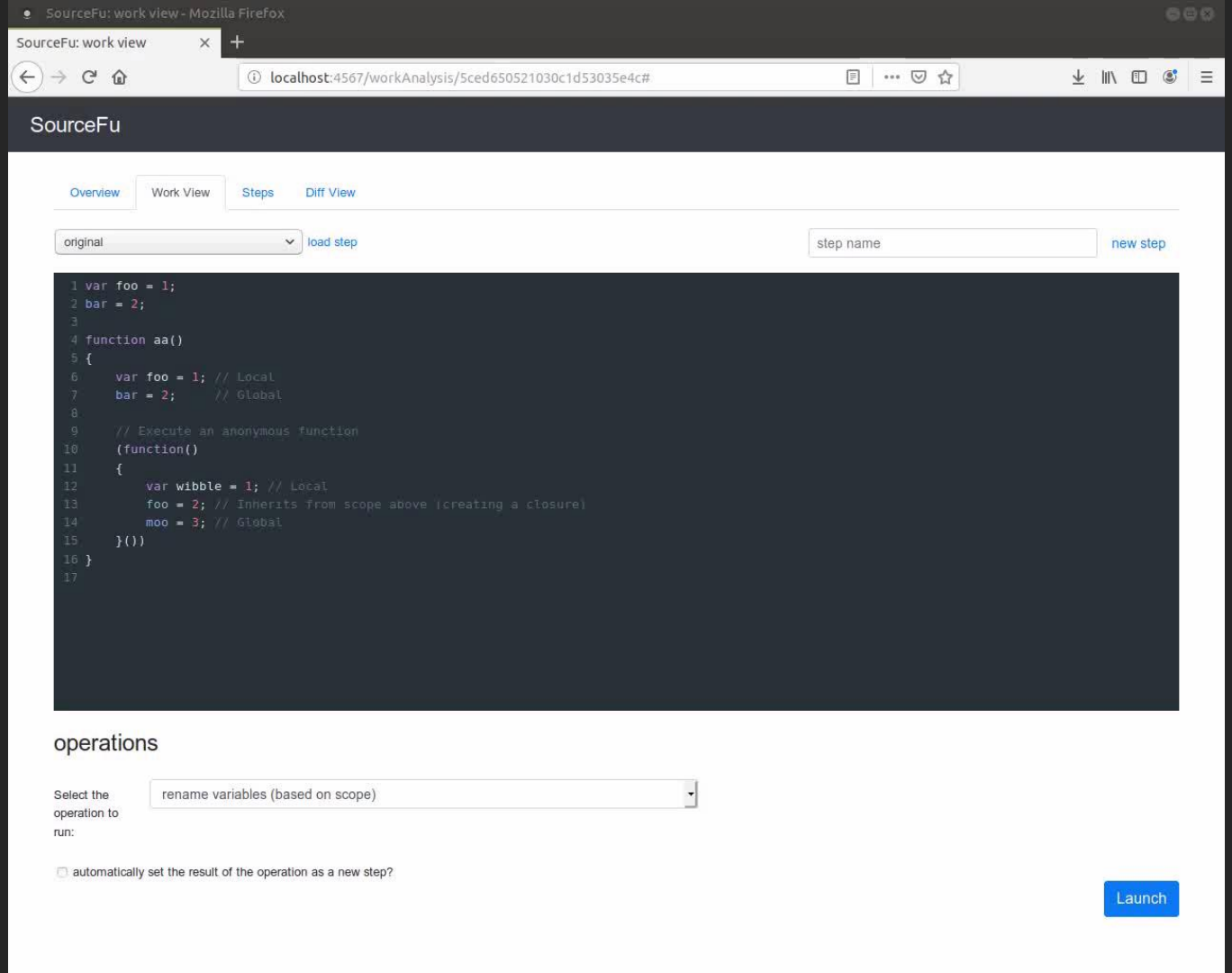
- Exemple de simplification de code
  - Blocs « IF-ELSE IF -ELSE »
    - Supprimer les conditions vérifiant « Faux »
    - Ne garder que la condition « Vrai » si toutes les conditions précédentes vérifient « Faux »

Sous le capot - 3

- Renommage des symboles
  - Eviter les variables avec un nom trop ressemblant

```
function lllIllllllI(){lllIllllllI='lllIllllllI';llllIllll
I='lllIlIlllI';lllIlllllll=llIllllllI+lllIllllllI+llllIlll
llI;return lllIlllllll}
```

  - Les variables sont listées par "scope", puis renommées en fonction de celui-ci avec un index


SourceFu: work view - Mozilla Firefox
SourceFu: work view
localhost:4567/workAnalysis/5ced650521030c1d53035e4c#
SourceFu
Overview
Work View
Steps
Diff View
original
load step
step name
new step
1 var foo = 1;
2 bar = 2;
3
4 function aa()
5 {
6 var foo = 1; // Local
7 bar = 2; // Global
8
9 // Execute an anonymous function
10 (function()
11 {
12 var wibble = 1; // Local
13 foo = 2; // Inherits from scope above (creating a closure)
14 moo = 3; // Global
15 }())
16 }
17
operations
Select the operation to run:
rename variables (based on scope)
automatically set the result of the operation as a new step?
Launch

Pourquoi le projet peut être intéressant

- SOC/CERT
- Amélioration capacités IDE et outils de traitement de sources/données
- Fuzzing à base de grammaires
- Recherche académique?

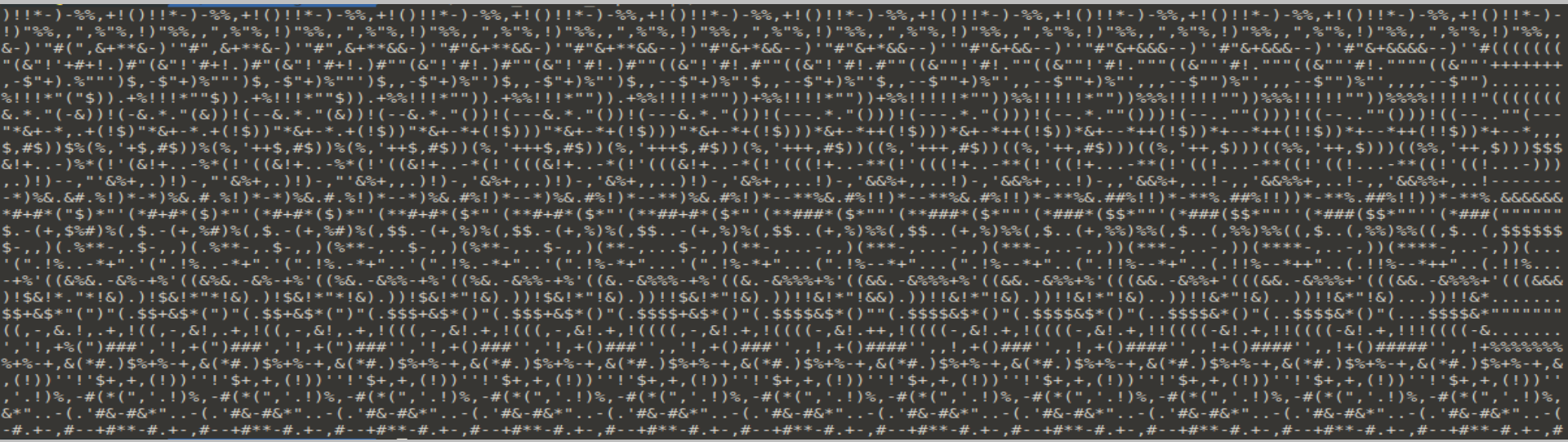
Etat actuel : SourceFu

- Version pre-alpha-0.01
- Supporte un peu le VBA, à peine le Javascript, et pas encore le Powershell….
- Plein d'améliorations possibles…
  - Intéraction directe AST, Évaluation plus complète, amélioration grammaires, intégration travail analyste…

Devenez contributeurs !

- Un simple intérêt pour l'outil suffit

https://github.com/Big5-sec/SourceFu



[Teaser] On se revoit plus tard dans l’année…