Graphviz rendu facile avec GvGen

Sébastien Tricaud

PyCon FR 2008

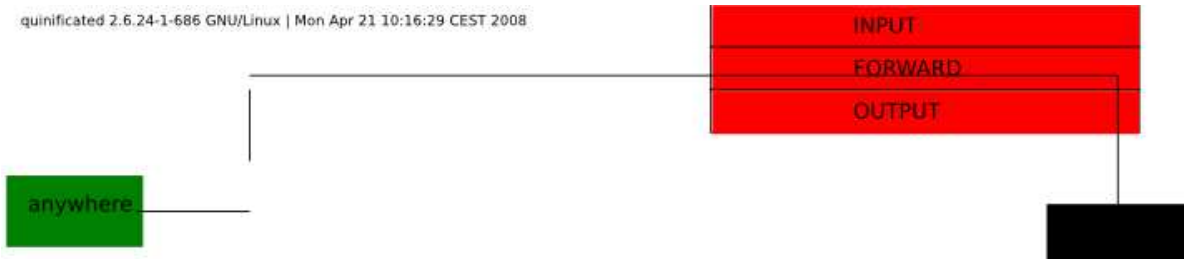
Besoins de visualisation

- Faire comprendre au chef le travail de sécurité
- Avoir une fa con (in)utile de représenter l'information
- Permettre l'exploitation de données rapidement
- Se concentrer sur l'essentiel (et se faire avoir sur le détail)

FW2FOO

Télécharger

http://ruined.sourceforge.net/fw2foo.tar.gz



Code: bash orienté objet (tm)

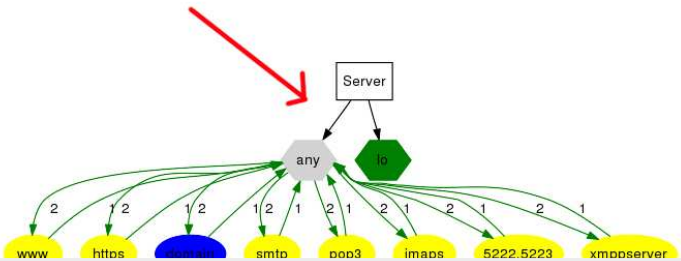
```
    objectlistlen=${#objchains_list[@]}
    while [ $i -lt $objectlistlen ]
    do
      object=${objchains_list[$i]}
      if [ "$object" == "$chain_name" ]
      then
  let "count += 1"
  if [ $count -eq $number ]
  then
      let "i += 1"
      echo "${objchains_list[$i]}"
      return
  fi
      fi
```

Ruined: RUby Iptables NEtwork Displayer

Télécharger

http://ruined.sourceforge.net/

```
# Generated by iptables-save v1.3.3 on Sat Jan 20 10:10:13 2007
*filter
:INPUT DROP [48:6961]
:FORWARD DROP [0:0]
:OUTPUT DROP [150:8452]
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m tcp --sport 80 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --sport 443 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m udp --sport 53 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --sport 25 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --sport 110 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --sport 993 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --sport 5222:5223 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --sport 5269 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 80 -m state --state NEW,RELATED,ESTABLISHED,UNTRACKED -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 443 -m state --state NEW,RELATED,ESTABLISHED,UNTRACKED -j ACCEPT
-A OUTPUT -p udp -m udp --dport 53 -m state --state NEW,RELATED,ESTABLISHED,UNTRACKED -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 25 -m state --state NEW,RELATED,ESTABLISHED,UNTRACKED -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 110 -m state --state NEW,RELATED,ESTABLISHED,UNTRACKED -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 993 -m state --state NEW,RELATED,ESTABLISHED,UNTRACKED -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 5222:5223 -m state --state NEW,RELATED,ESTABLISHED,UNTRACKED -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 5269 -m state --state NEW,RELATED,ESTABLISHED,UNTRACKED -j ACCEPT
COMMIT
# Completed on Sat Jan 20 10:10:13 2007
```



Code: Ruby

```
text.each do |line|

  $p = Policy.new

    case line
      when /^(\*).*/
        $t = Table.new("#{line.chomp.dump}")
      when /^(\:).*/
        $p.set($t.get_latest,"#{line.chomp.dump}")
      when /^(\-).*/
        string = line.chomp.dump
        string = string.gsub("\"","")
        $r = Rule.new("\-t #{$t.get_latest} #{string}")
      when /^(\#).*/
      when /^COMMIT/
    end
```

GvGlue

- Automatiser la création de graphes dirigés
- Besoins de tels graphes pour la sécurité informatique
- Besoin d'une API simple, cachant la complexité de Graphviz
- Après avoir essayé pas mal de languages, est-ce que python sera à la hauteur ?

Code: Python

```
try:
  for runner in self.data:
      if runner.find("%d [label=" % subgraphid) != -1:
          i = self.data.index(runner)
          mystr = runner[:len(runner)-1] + \
                  ",%s=\"%s\""%(propertyname, value) + \
                  "];"
          self.data.pop(i)
          if not getstr:
            self.data.insert(i, mystr)
```

Problème

- GvGlue était inmaintenable
- Besoin de fiabilité pour le projet Prelude IDS
- La visualisation est incontournable en sécurité : garder les objectifs de GvGlue

Ordonner des objets IDMEF

Code: Création du graphe

```
graph = gvgen.GvGen()
alert_g = graph.newItem("IDMEF Alert")
```

Ordonner des objets IDMEF

Code: Récupération de l'objet

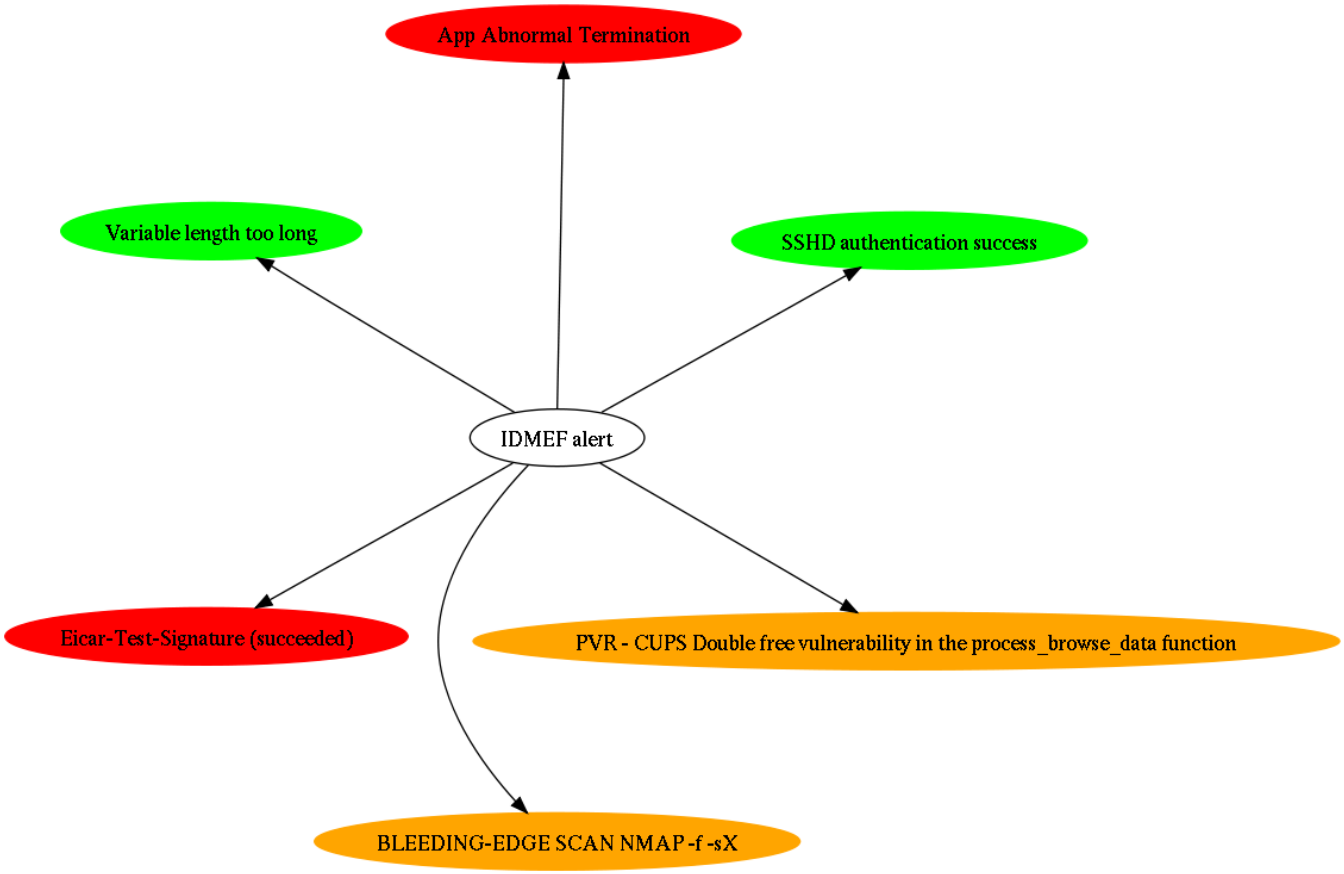
```
value = idmef.Get("alert.classification.text")
if value:
```

Ordonner des objets IDMEF

Code: Distribution

```
act = graph.newItem("alert.classification.text",
                    None, distinct=1)
actc = graph.newItem(value, act, distinct=1)
graph.newLink(alert_g, actc)
```

Graphe obtenu



Creer des nodes

```
graph = pydot.Dot()
riri = pydot.Node("Riri")
fifi = pydot.Node("Fifi")
loulou = pydot.Node("Loulou")
```

Créer des parents

```
grand_parent = pydot.Cluster(graph_name = \
  "GrandParent", label = "Picsou")
graph.add_subgraph(grand_parent)

parent = pydot.Cluster(graph_name = \
  "Parent", label = "Donald")
grand_parent.add_subgraph(parent)

enfants = pydot.Cluster(graph_name = \
  "Enfants", label = "Enfants")
enfants.add_node(riri)
```

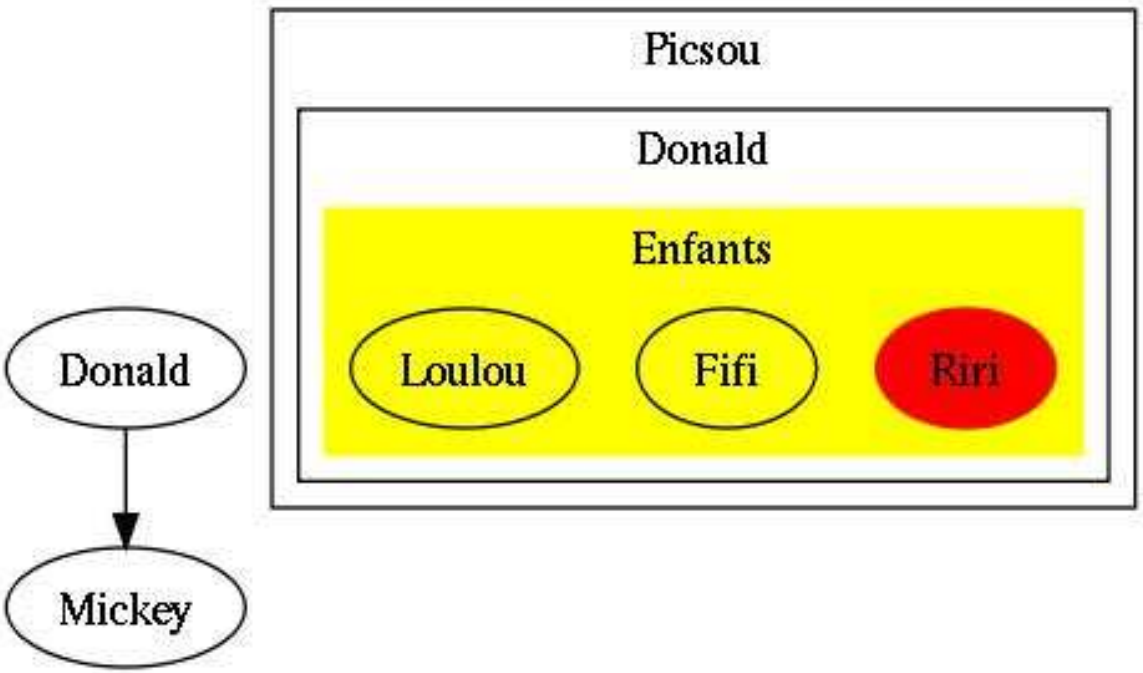
Créer des liens

```
graph.add_edge(pydot.Edge("Donald", "Mickey"))
```

Ajouter des propriétés

```
enfants.set("color", "yellow")
enfants.set("style", "filled")
riri.set("color", "red")
riri.set("style", "filled")
```

Obtenir l'image



Code dot généré

```
digraph G {
        node [label="\N"];
        graph [bb="0,0,348,214"];
        subgraph cluster_GrandParent {
                graph [label=Picsou,
                        lp="210,195",
                        bb="80,56,340,206"];
                subgraph cluster_Parent {
                        graph [label=Donald,
                                lp="210,165",
                                bb="88,64,332,176"];
                        subgraph cluster_Enfants {
                                graph [label=Enfants,
                                        style=filled,
                                        color=yellow,
                                        lp="210,135",
                                        bb="96,72,324,146"];
                                Riri [style=filled, color=red, pos="289,98", width="0.75", height="0.50"];
                                Fifi [pos="217,98", width="0.75", height="0.50"];
                                Loulou [pos="138,98", width="0.94", height="0.50"];
                        }
                }
        }
        Mickey [pos="36,26", width="1.00", height="0.50"];
        Donald [pos="36,98", width="1.00", height="0.50"];
        Donald -> Mickey [pos="e,36,44 36,80 36,72 36,63 36,54"];
}
```

Pourquoi GvGen est mieux(tm)

Créer des nodes + parents

```
graph = gvgen.GvGen()

picsou = graph.newItem("Picsou")
donald = graph.newItem("Donald", picsou)
riri = graph.newItem("Riri", donald)
fifi = graph.newItem("Fifi", donald)
loulou = graph.newItem("Loulou", donald)
```

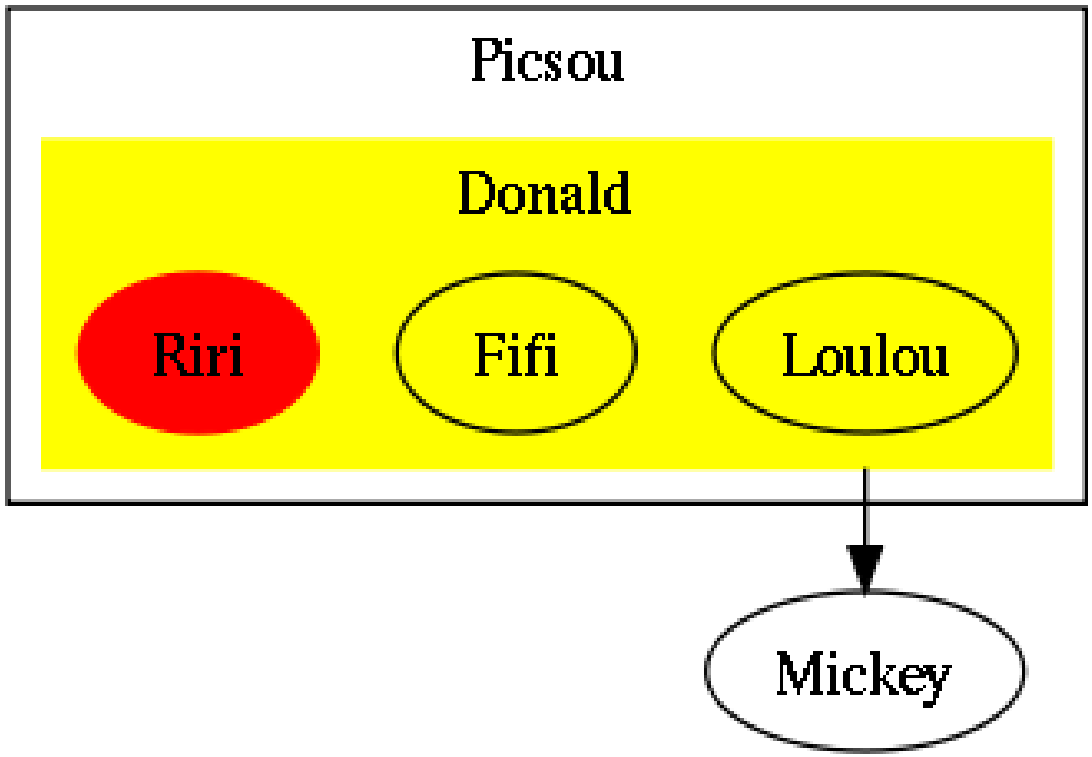
Créer des liens

```
graph.newLink(donald, mickey)
```

Ajouter des propriétés

```
graph.propertyAppend(donald, "color", "yellow")
graph.propertyAppend(donald, "style", "filled")
```

Obtenir l'image



Code dot généré

```
digraph G {
compound=true;
   subgraph cluster1 {
   label="Picsou";
      subgraph cluster2 {
      color="yellow";
      style="filled";
      label="Donald";
         node5 [label="Loulou"];
         node4 [label="Fifi"];
         node3 [color="red",style="filled",label="Riri"];
      }
   }
   node6 [label="Mickey"];
node5->node6 [ltail=cluster2];
}
```

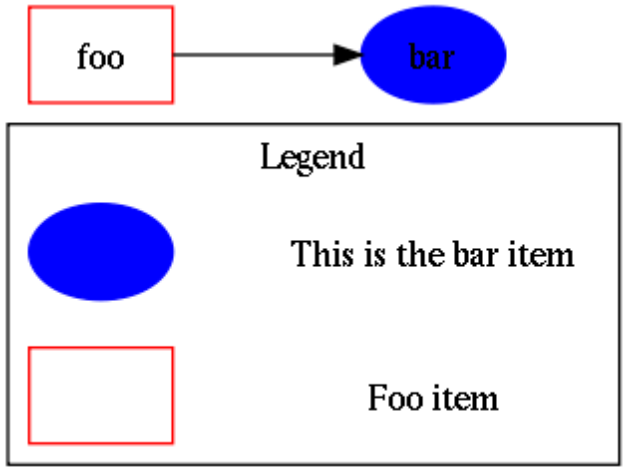
Les effets Kiss-cool

Applicables sur parent comme enfant

```
graph.styleAppend("Post", "color", "blue")
graph.styleAppend("Post", "style", "filled")
graph.styleAppend("Post", "shape", "rectangle")
graph.styleApply("Post", postman)
```

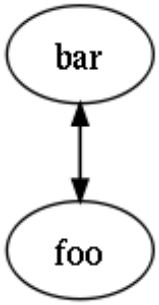
Essayez de le faire en graphviz ;)

```
graph = gvgen.GvGen("Legend")
....
graph.legendAppend("foostyle", "Foo item")
graph.legendAppend("barstyle", "This is the bar item")
```



Double sens

```
graph.smart_mode = 1
graph.newLink(a,b)
graph.newLink(b,a)
```



Téléchargez

svn co http://software.inl.fr/svn/mirror/tools/gvgen

Page du projet

http://software.inl.fr/trac/wiki/GvGen

Projets utilisant GvGen

- **Graphdep**: http://haypo.hachoir.org/trac/wiki/graphdep
- **Iptables-graph**: http://software.inl.fr/trac/trac.cgi/browser/mirror/tools/iptables-graph

Questions ?

Merci de votre attention

Contactez moi:

- Sébastien Tricaud <s.tricaud@inl.fr> http://www.gscore.org/blog