L’apport de la visualisation dans la détection d’intrusion

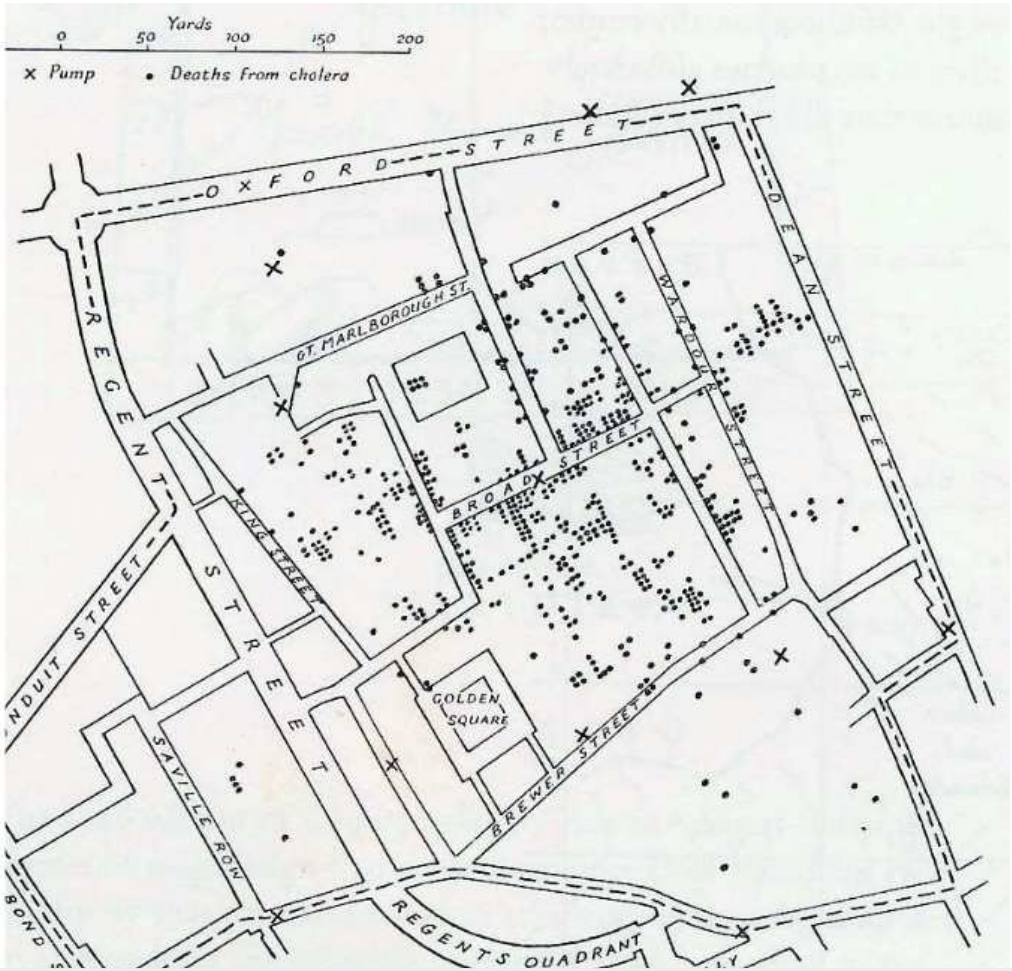
Sébastien Tricaud Pierre Chifflier

INL
15 rue Berlier
75013 Paris, France

OSSIR Rennes, Octobre 2008

Un peu d'aviation

Épidémie de choléra à Londres



Sébastien Tricaud, Pierre Chifflier

INL - 15 rue Berlier 75013 Paris, France

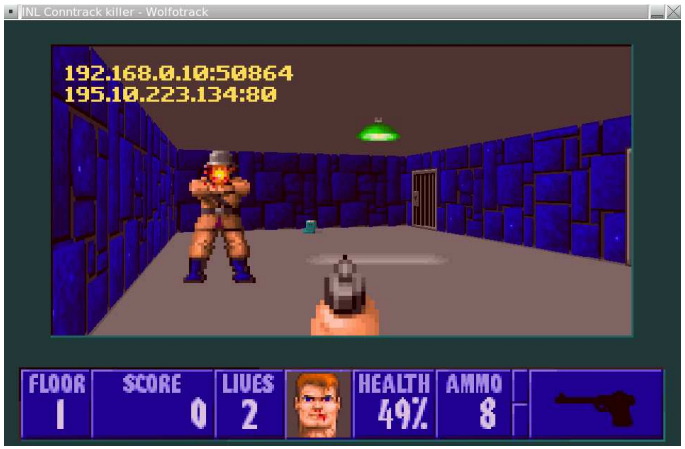
Propriétés d'un élément

- Remplissage (couleur, image, rayures..)
- Contraste
- Forme
- Orientation
- Hauteur
- Largeur
- Forme

Deux types d'associations de propriétés

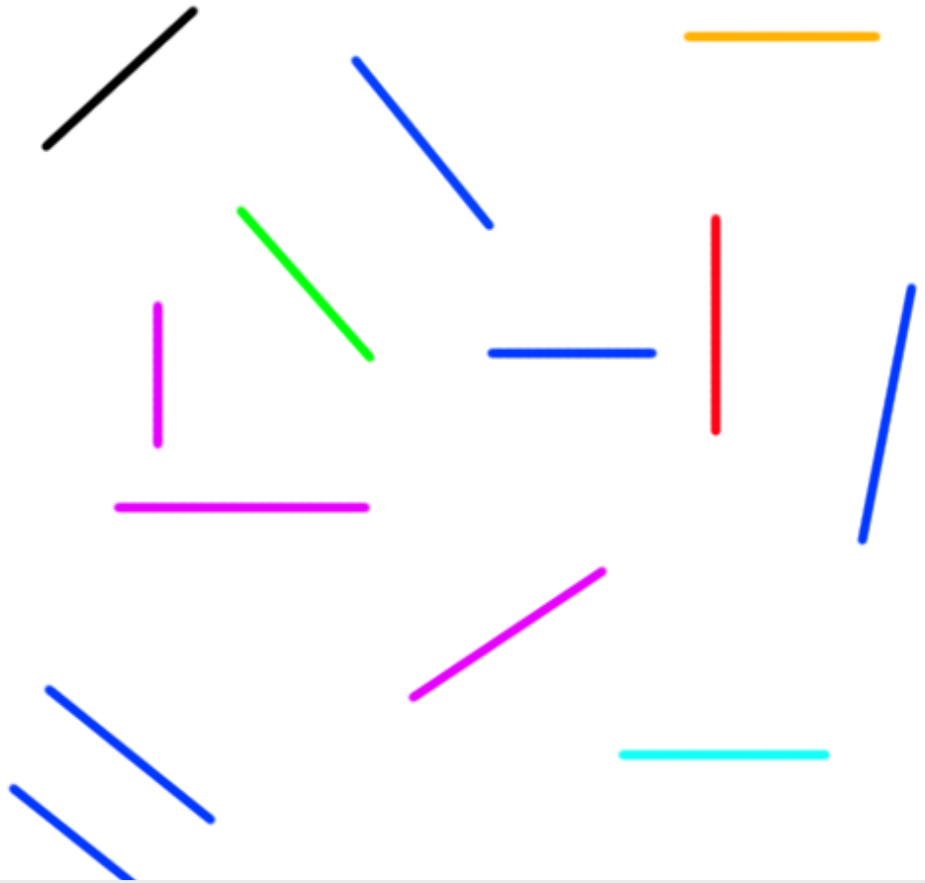
- Dimensions séparables : faciles à repérer
- Dimensions intégrales : difficiles à repérer

Ce qui n'est pas trés recommendable

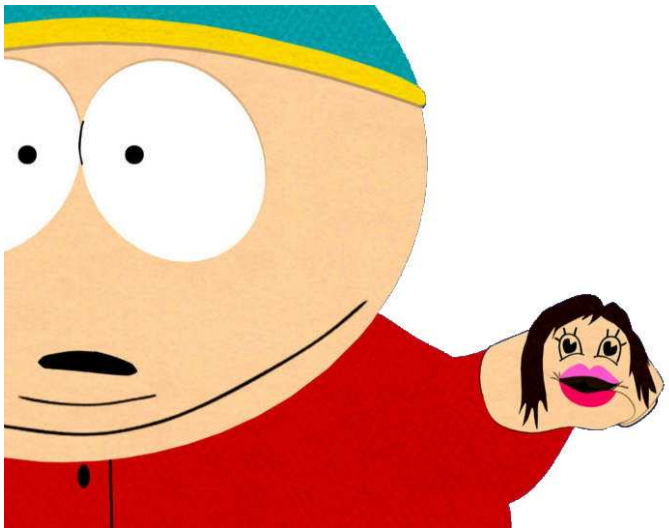


Hauteur vs Largeur

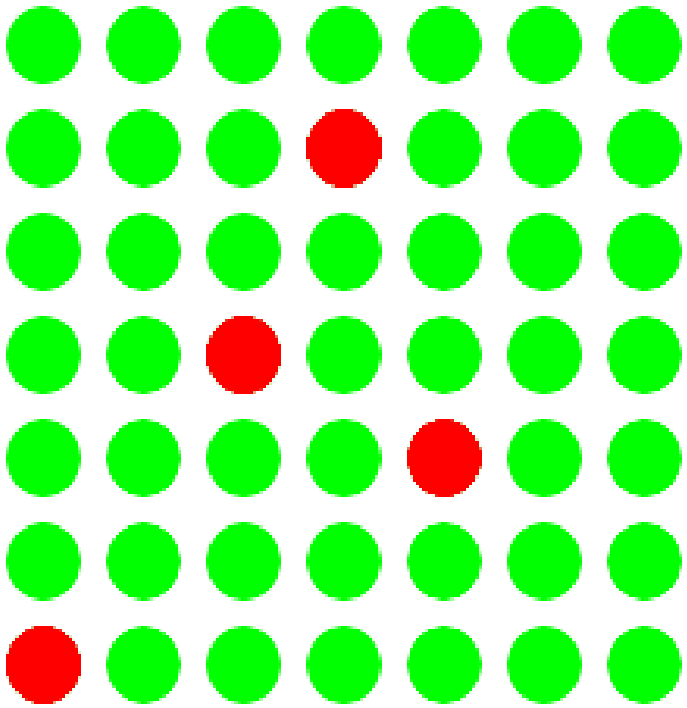
Orientation vs Couleur



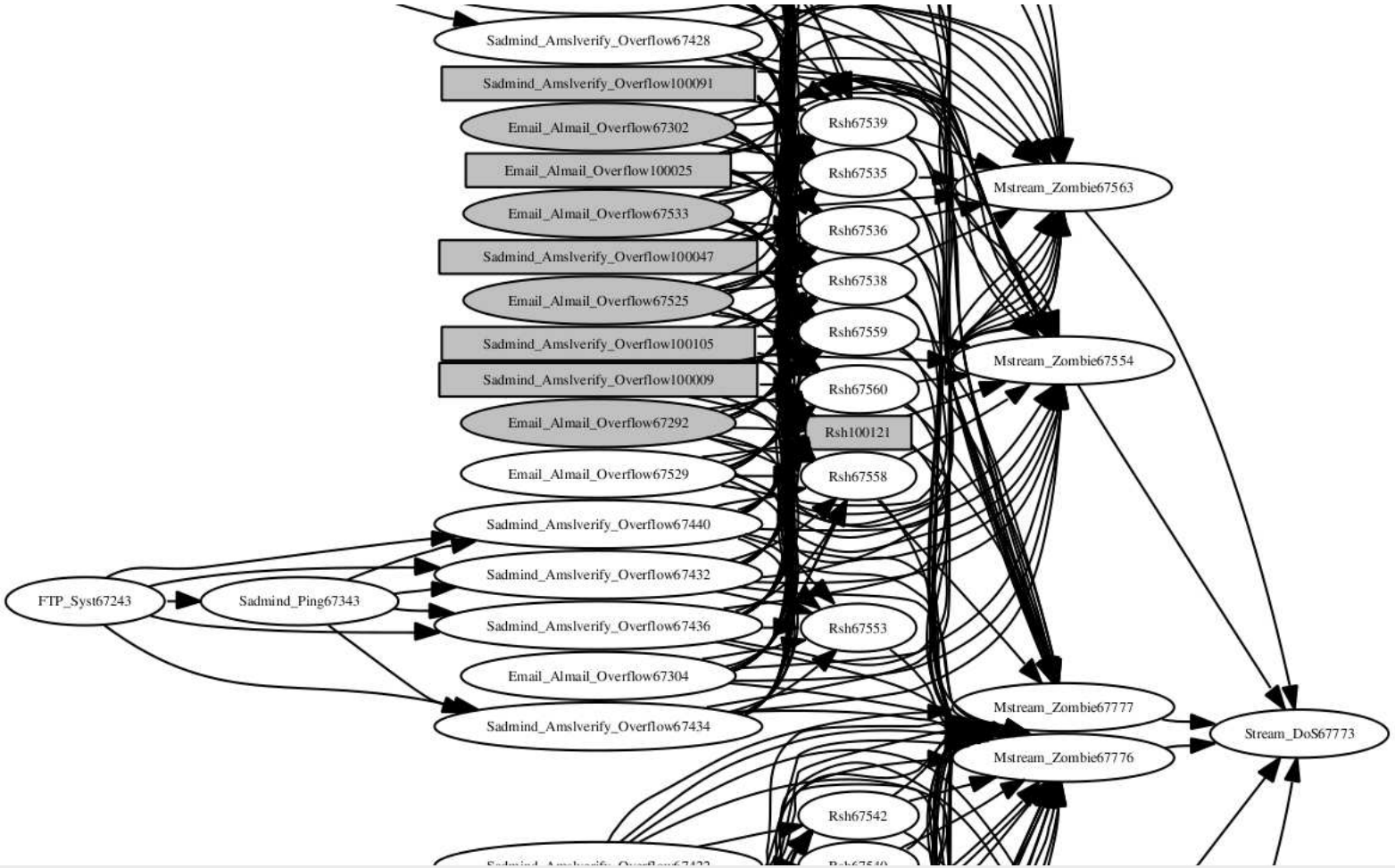
Ce qui est recommendable



Forme vs Couleur



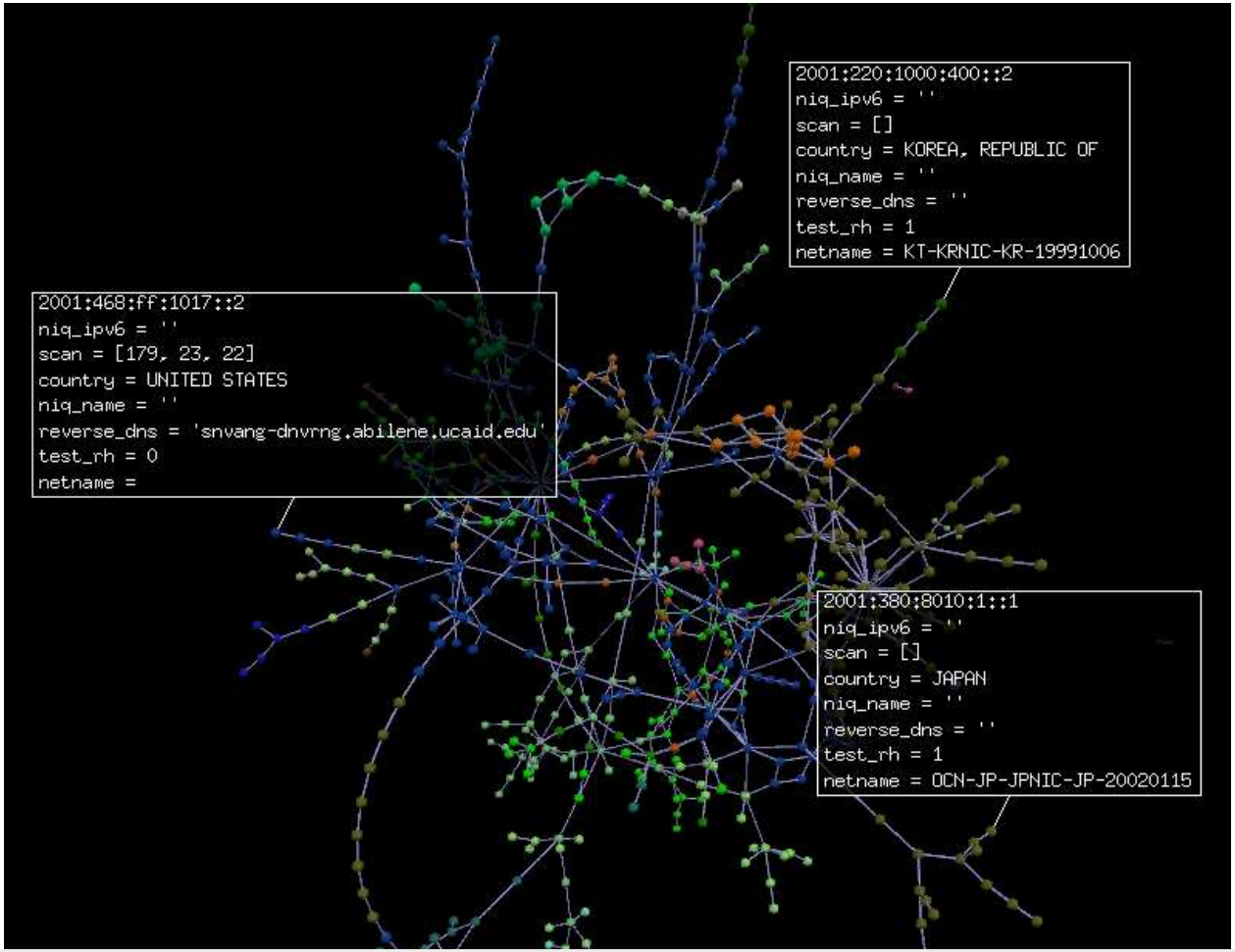
LLDOS 1.0 Alert correlation graph



Sébastien Tricaud, Pierre Chifflier

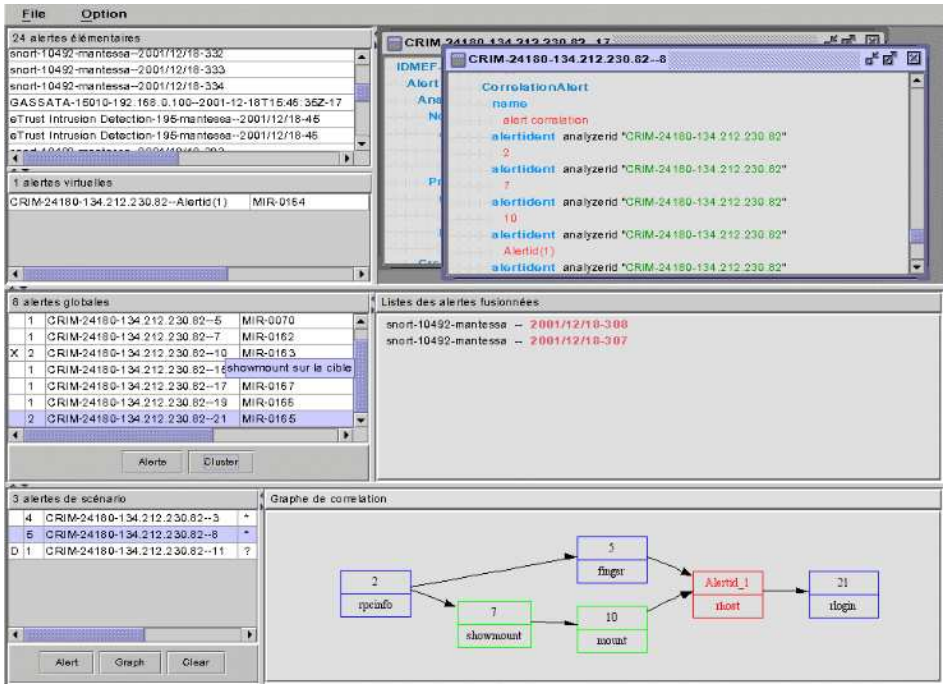
INL - 15 rue Berlier 75013 Paris, France

IPv6 world


2001:220:1000:400::2
niq_ipv6 = ''
scan = []
country = KOREA, REPUBLIC OF
niq_name = ''
reverse_dns = ''
test_rh = 1
netname = KT-KRNIC-KR-19991006
2001:468:ff:1017::2
niq_ipv6 = ''
scan = [179, 23, 22]
country = UNITED STATES
niq_name = ''
reverse_dns = 'snvang-dnvrng.abilene.ucaid.edu'
test_rh = 0
netname =
2001:380:8010:1::1
niq_ipv6 = ''
scan = []
country = JAPAN
niq_name = ''
reverse_dns = ''
test_rh = 1
netname = OCN-JP-JPNIC-JP-20020115

Correlation UI

Alert Correlation in a Cooperative Intrusion Detection Framework (Frédéric Cuppens / Alexandre Miège) :



Prelude notify

Notification pour l'administrateur

Que sont les IDS ?

- Systèmes de détection d'intrusions
- Les gens du marketing l'appellent
  - Systèmes de prévention d'intrusions (IPS)[1]
  - Security Information and Event Management (SIEM)
- Pendant cette présentation, on restera sur le terme IDS

[1] Pour prévenir une attaque, il faut d'abord la détecter ;)

Qu’est-ce qui existe ?

- Host IDS (HIDS)
- Network IDS (NIDS)
- Meta IDS (MIDS)

Beaucoup de sources d'informations

Un maximum de sources permet de trouver plus d'attaques

Beaucoup de sources d'informations

Un maximum de sources permet de trouver plus d'attaques

Sources bas niveau :

- **Routeurs** : Cisco, Linksys, Juniper, . . .
- **Pare-feux** : Netfilter, NuFW, Checkpoint, pf, . . .
- **Systémes d'exploitation** : journaux systèmes et applicatifs, utilisateurs, applications lancées, . . .
- **Physique** : Alarmes, . . .

Beaucoup de sources d'informations

Un maximum de sources permet de trouver plus d'attaques

Sources bas niveau :

- **Routeurs** : Cisco, Linksys, Juniper, ...
- **Pare-feux** : Netfilter, NuFW, Checkpoint, pf, ...
- **Systémes d'exploitation** : journaux systèmes et applicatifs, utilisateurs, applications lancées, ...
- **Physique** : Alarmes, ...

Sources haut niveau :

- **Pots de miel** : Nepenthes, ...
- **Réseau** : Snort, Sancp, NuFW, ...
- **Machines** : Auditd (SELinux), Linux PAM, Samhain, Ossec, Prelude LML, ClamAV ...
- **Scanners** : Nessus, p0f, nmap ...

Exemples d'alertes :

- OSSEC : `SSHD authentication success.`
- Prelude LML : `Admin login successful`
- Snort : `BLEEDING-EDGE SCAN NMAP -f -sS`
- ClamAV : `Eicar-Test-Signature (succeeded)`
- Auditd (SE Linux) : `App Abnormal Termination`

Limitations inhérentes aux IDS

- Trop d'informations
- Vue limitée
- Faux positifs
- Faux négatifs
- Évasion (fragmentation, signatures, temps, . . .)

les IDS et la corrélation

- Pour limiter les lacunes des IDS, il nous faut la corrélation :
  - Meta-IDS
  - Architecture distribuée et adaptative pour **centraliser** l'information
  - Définir avec précision **chaque** alerte et **chaque** agent

IDMEF: Intrusion Detection Message Exchange Format

- Normaliser les alertes des agents sans regarder leur type
  - Une alerte peut-être hétérogène
  - Les environements de détection d'intrusions sont différents
  - Les environements d'opération sont différents
  - Les capacités des agents sont différentes
- Fournir un vocabulaire exhaustif pour les développeurs et utilisateurs d'IDS

$\Rightarrow$ IDMEF (RFC 4765)
`http://www.rfc-editor.org/rfc/rfc4765.txt`

Exemple d'alerte : NuFW 1/3

- Exemple d'alerte IDMEF, avec les champs remarquables.
- Alerte émise pour une nouvelle connection HTTP utilisant Firefox.

```
messageid: 5478076470
analyzer(1):
        analyzerid: 2334565015741136
        name: nufw
        manufacturer: http://www.nufw.org/
        model: NuFW
        version: 2.3.0 ($Revision: 3475 $)
        class: Firewall
        ostype: Linux
        osversion: 2.6.20-15-386
        process:
                name:
                pid: 15197
```

Exemple d'alerte : NuFW 2/3

```
create_time: 29/06/2007 11:26:24.0 +02:00
classification:
        text: Connection opened
detect_time: 29/06/2007 11:32:56.0 +02:00
analyzer_time: 29/06/2007 11:32:56.642005 +02:00
source(0):
        spoofed: unknown (0)
        node:
                category: unknown (0)
                address(0):
                        category: ipv4-addr (7)
                        address: 192.168.0.2
        user:
                category: application (1)
                user_id(0):
                        type: current-user (1)
                        name: pollux
                        number: 1000
        process:
                name: firefox
                path: /usr/bin/firefox
        service:
                iana_protocol_number: 6
                iana_protocol_name: tcp
                port: 3489
```

Exemple d'alerte : NuFW 3/3

```
target(0):
        decoy: unknown (0)
        node:
                category: unknown (0)
                address(0):
                        category: ipv4-addr (7)
                        address: 82.165.85.221
        service:
                iana_protocol_number: 6
                iana_protocol_name: tcp
                port: 80
assessment:
        impact:
                severity: low (2)
                type: user (5)
                description: Connection state changed
```

Prelude IDS

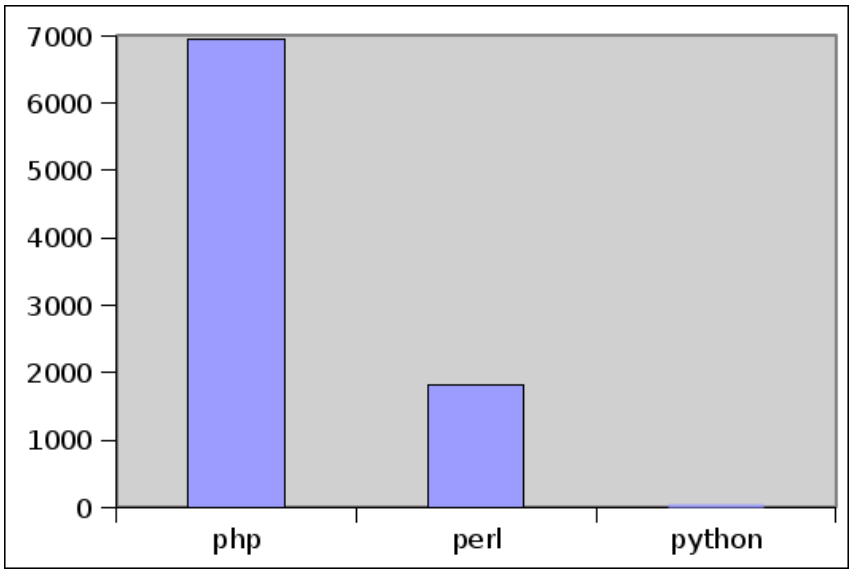
- Meta-IDS implémentant IDMEF
  - libprelude, libpreludedb
  - Prelude LML: Analyse les logs
  - Prelude Correlator: Correle les alertes des agents
  - Prelude Manager: Centralise, stocke/relaye les alertes
  - Prewikka: Interface graphique

Prelude IDS

- Meta-IDS implémentant IDMEF
  - libprelude, libpreludedb
  - Prelude LML: Analyse les logs
  - Prelude Correlator: Correle les alertes des agents
  - Prelude Manager: Centralise, stocke/relaye les alertes
  - Prewikka: Interface graphique
- Fonctionnalités requises pour la corrélation:
  - **Cryptage** (GnuTLS) entre agents et managers, de manager à manager
  - **Failover**, quand votre agent est coupé et ne peut pas envoyer ses alertes
  - **Relayer** pour centraliser, sauvegarder et filtrer les alertes (permettant aussi la haute disponibilité)
  - **Relayage inverse** pour sécuriser les DMZ
  - **Normaliser** vos alertes : compléter le message IDMEF

Comprendre notre environnement

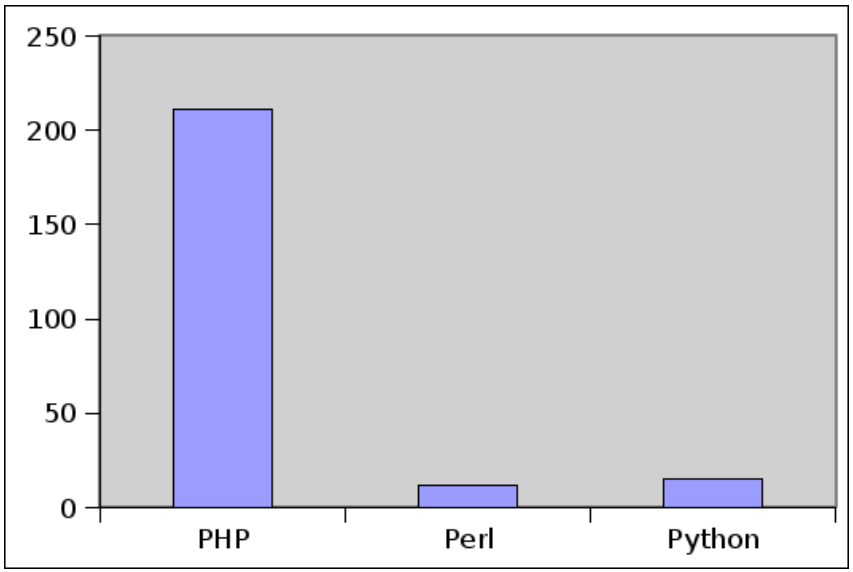
Contexte des vulnérabilités détectées
Janvier 1970 - Juin 2008



Recherche de tous les CVE ayant pour mot le language sur http://nvd.nist.gov/

Comprendre notre environnement

Contexte des vulnérabilités détéctées
Janvier 1970 - Juin 2008

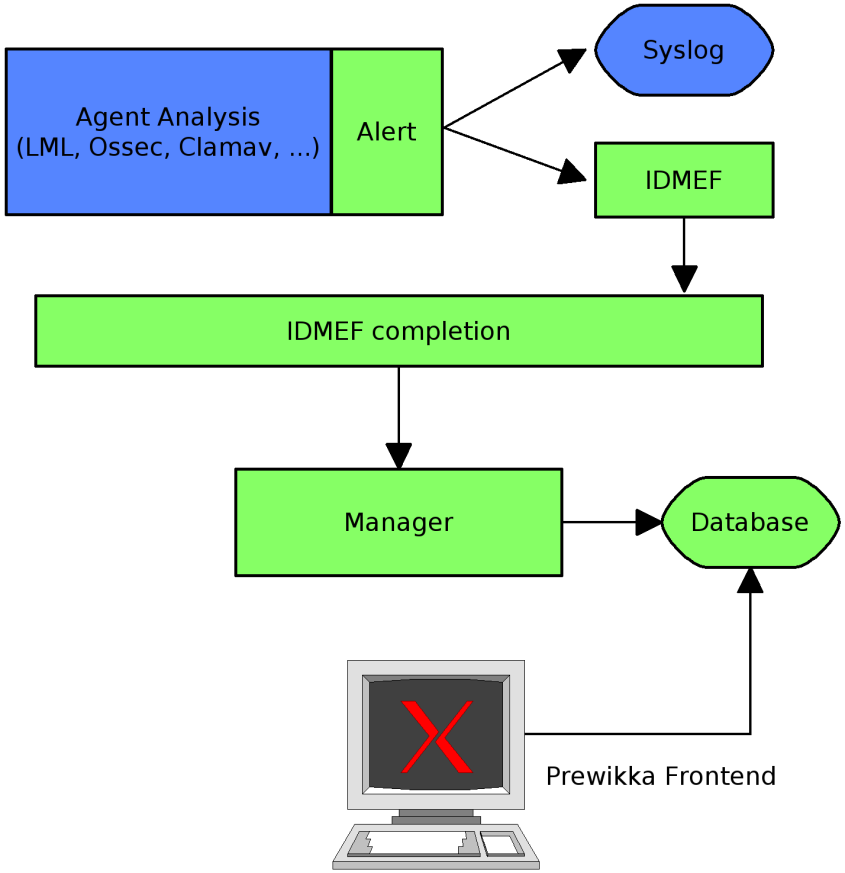


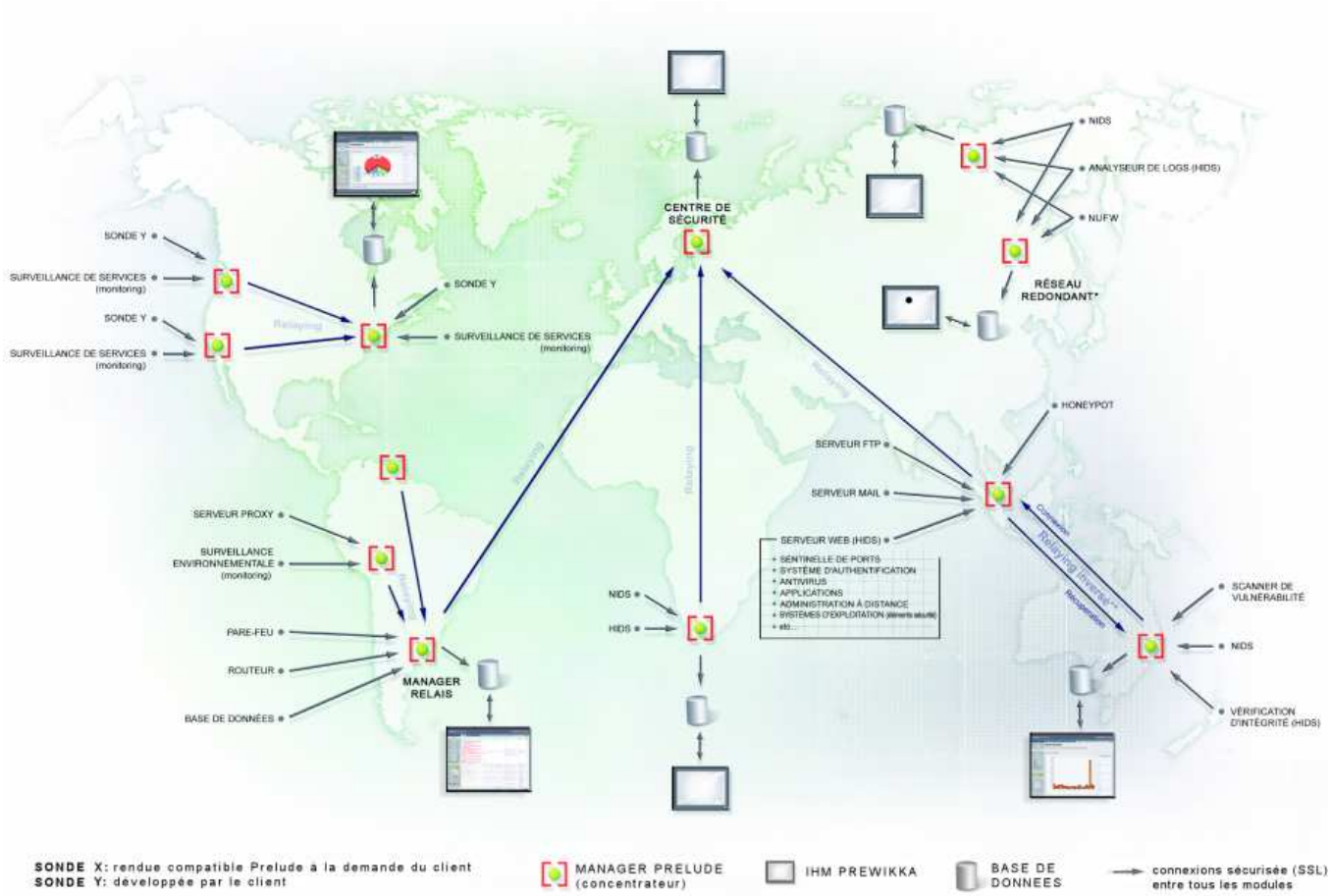
Recherche de tous les CVE touchant directement le language sur http://nvd.nist.gov/

Objectifs de Prelude IDS

- Détecter les intrusions
  - Combiner un maximum de sources
  - Faciliter l'integration d'outils existants
- Gérer les alertes
  - Normalisation
  - Relais, Sauvegardes
  - Assurance d'arrivée
- Permettre le traitement aisé des alertes
  - IHM Prewikka

Exemple d'application fonctionnant avec Prelude





Sébastien Tricaud, Pierre Chifflier
INL - 15 rue Berlier 75013 Paris, France

5 diapositives pour coder un IDS ainsi que le code Prelude

- Vérifier les modifications sur un fichier
- Utilisation de MD5 additionné pour chaque octet
- Construction d'une base de données de référence
- Envoi du message IDMEF à Prelude

Code Python 1/4

Faire une somme md5 d'un fichier

```
def sumfile(file):
        fp = open(file, 'rb')
        sum = md5.new()

        for char in fp.read():
                sum.update(char)

        fp.close()

        return sum.hexdigest()
```

Code Python 2/4

Construction d'une base de données

```
dirfiles = os.listdir(".")
for fname in dirfiles:
        if createdb:
                fpmd5db = open(md5db, "a")
                data = "%s:%s\n" % (fname,sumfile(fname))
                fpmd5db.write(data)
                fpmd5db.close()
```

Code Python 3/4

Récupérer les MD5 de la base

```
def getmd5fromdb(file):
        fp = open(md5db, "r")
        for line in fp.readlines():
                (f,m) = line.split(':')
                if file == f:
                        fp.close()
                        return m[:len(m)-1]

        return -1
```

Code Python 4/4

La somme de correspond pas ? on alerte

```
m = getmd5fromdb(fname)
if m != sumfile(fname):
        print "ALERT! File %s was altered" % fname
```

Code Prelude 1/1

On remplace le print par le code Prelude

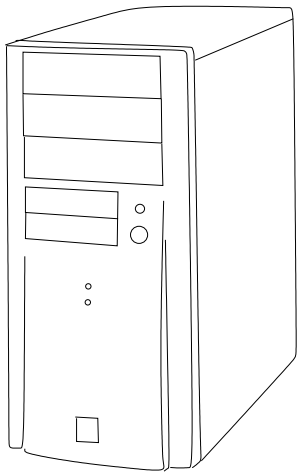
```
import PreludeEasy

idmef = PreludeEasy.IDMEF()
alertext = "File %s was altered" % fname
idmef.Set("alert.classification.text", alertext)

client = PreludeEasy.ClientEasy("JRES sensor")
client << idmef
```

La correlation

Transformation d’une ou plusieurs alertes en attaque

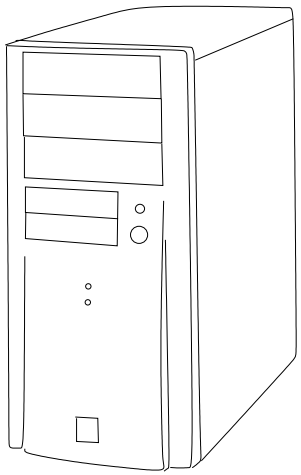
Sébastien Tricaud, Pierre Chifflier
INL, 15 rue Berlier, 75013 Paris, France

1. Scan

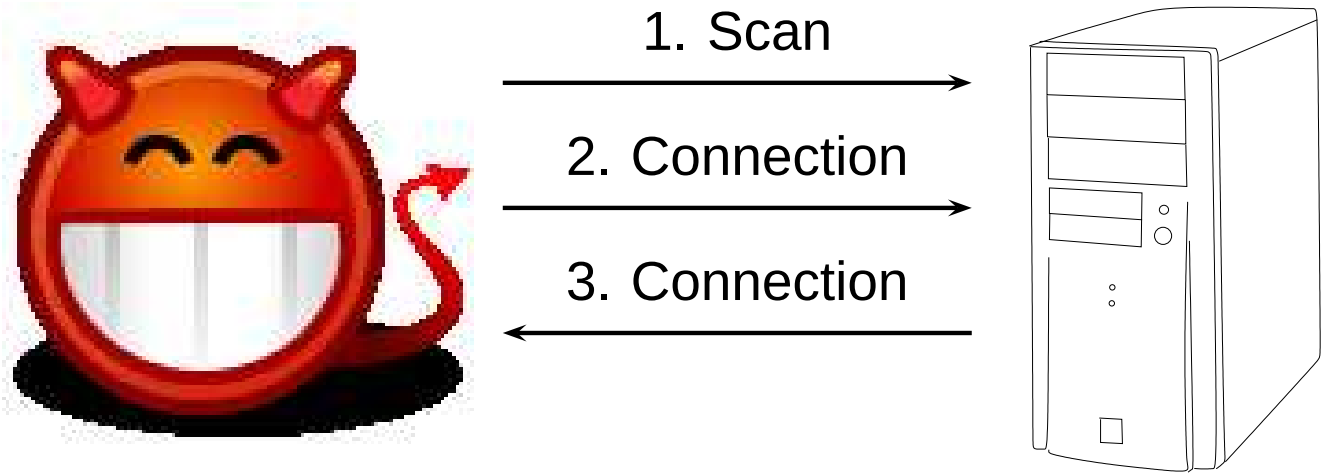


1. Scan

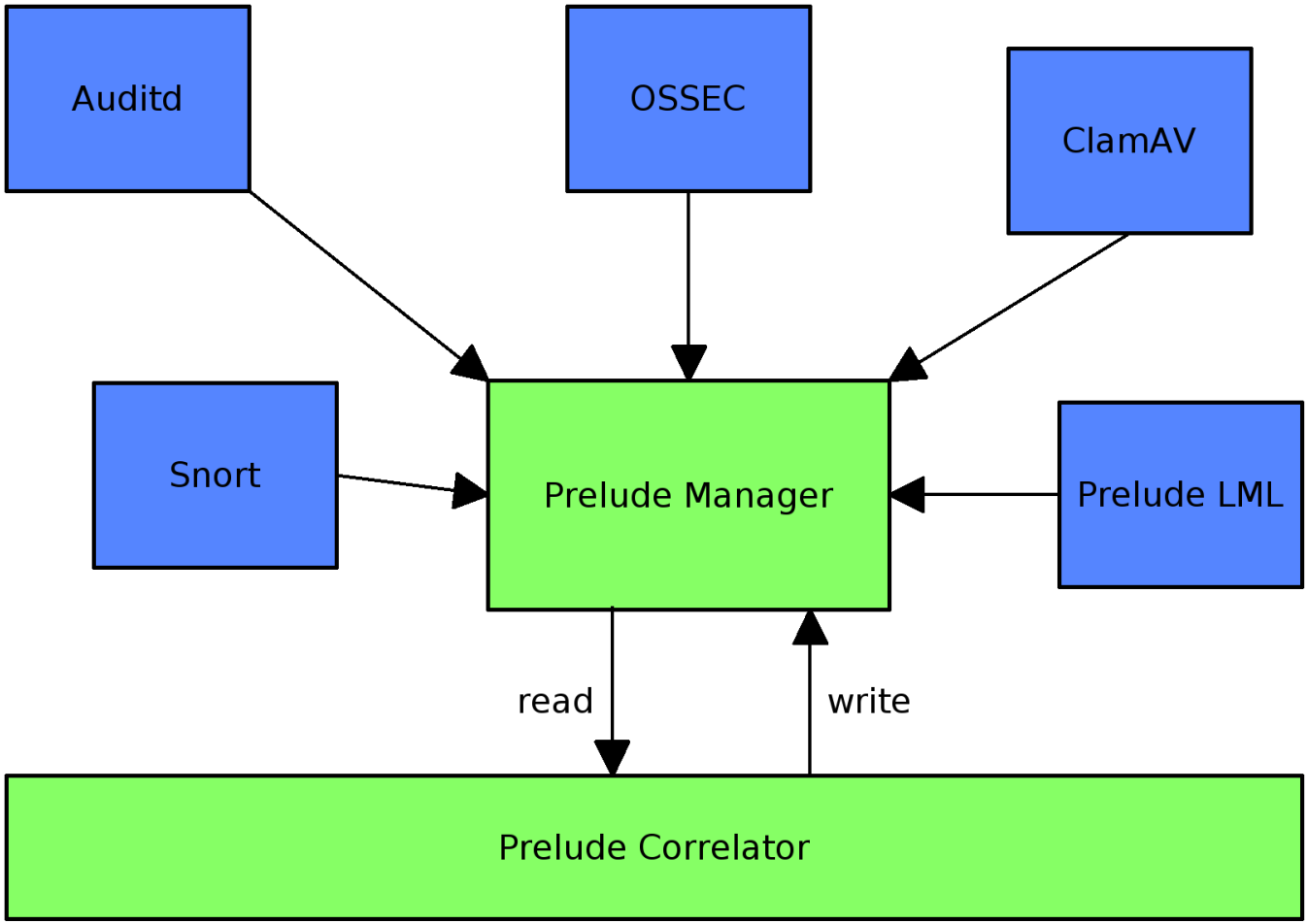
2. Connection



Sébastien Tricaud, Pierre Chifflier INL - 15 rue Berlier 75013 Paris, France



Sébastien Tricaud, Pierre Chifflier
INL, 15 rue Berlier, 75013 Paris, France


Auditd
OSSEC
ClamAV
Snort
Prelude Manager
Prelude LML
read
write
Prelude Correlator

Règle de correlation

Code 1/3 : Données qui nous intéressent

```
function brute_force(INPUT)

local is_failed_auth = INPUT:match("alert.classification.text",
                                   "[Ll]ogin|[Aa]uthentication",
                                   "alert.assessment.impact.completion",
                                   "failed")


local result = INPUT:match("alert.source(*).node.address(*).address", "(.+)",
                           "alert.target(*).node.address(*).address", "(.+)");
```

Règle de correlation

Code 2/3 : Mise à jour du contexte

```
if is_failed_auth and result then
    for i, source in ipairs(result[1]) do
        for i, target in ipairs(result[2]) do

            local ctx = Context.update("BRUTE_ST_" .. source .. target,
                                       { expire = 2, threshold = 5 })
            ctx:set("alert.source(>>)", INPUT:getraw("alert.source"))
            ctx:set("alert.target(>>)", INPUT:getraw("alert.target"))
            ctx:set("alert.correlation_alert.alertident(>>).alertident",
                    INPUT:getraw("alert.messageid"))
            ctx:set("alert.correlation_alert.alertident(-1).analyzerid",
                    INPUT:getAnalyzerid())
```

Règle de correlation

Code 3/3 : Envoi de l'alerte

```
if ctx:CheckAndDecThreshold() then
    ctx:set("alert.classification.text", "Brute force attack")
    ctx:set("alert.correlation_alert.name", "Multiple failed login")
    ctx:set("alert.assessment.impact.severity", "high")
    ctx:set("alert.assessment.impact.description",
     "Multiple failed attempts have been made to login to a user account")
    ctx:alert()
    ctx:del()
end
```

Réactions

- Raporter le problème (email)
- Archiver
- Prépare la visualisation
- Contre-mesure
  - (essayer de) bloquer une attaque (*dangereux !*)
  - Collecter plus d'informations
  - Envoyer des commandes aux agents
- Notifier

Récupérez les sources

svn co http://svn.prelude-ids.org/prelude-correlator/trunk

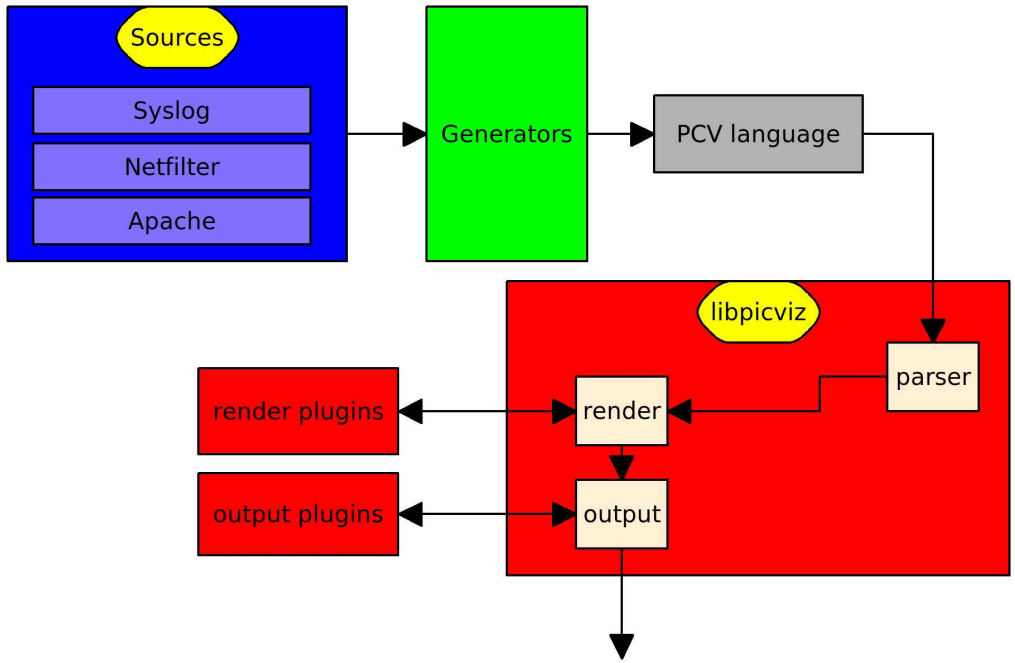
Picviz

Picviz
http://www.wallinfire.net/picviz

But

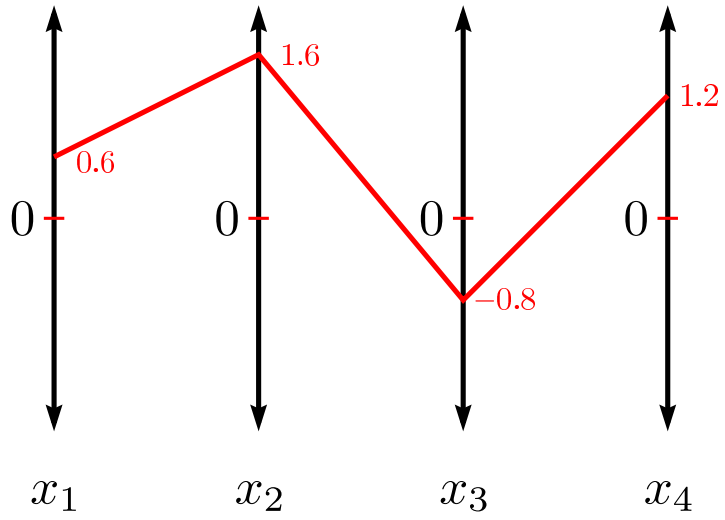
- Permettre la création et l'exploitation des coordonnées parallèles
  - Facile à scripter
  - Facile à comprendre (après initiation ;-])
  - Facile à filtrer
  - Redoutable lorsque l'on veut comprendre plusieurs millions d'évènements

Architecture



Kesako les coordonnés parallèles ? (1/2)

$$\vec{u} = (0.6, 1.6, -0.8, 1.2) \in \mathbb{R}^4$$

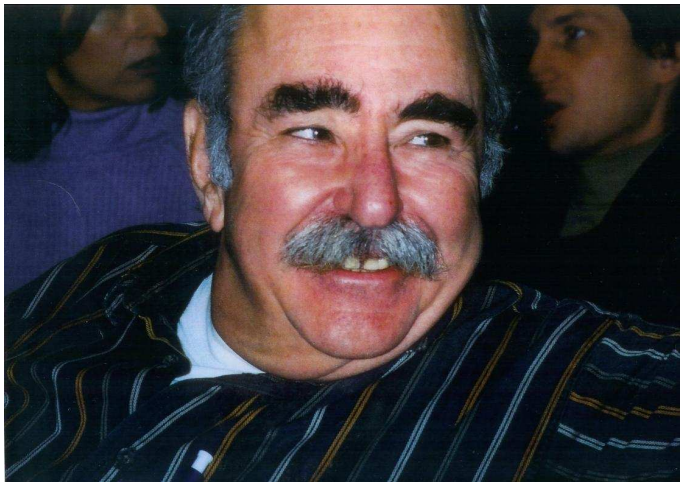


Kesako (2/2)

- N dimensions, ∞ évènements, tout type d'évènement
- Chaque axe
  - contient une variable différente
  - doit être équidistant
  - re coit la valeur minimale de chaque variable en bas, maximale en haut
- L'ordre est important
  - Temps = Premier axe
  - Source à gauche de la destination
  - Données 'garbage' sur le dernier axe

L'inventeur

Inventé et surtout appliquée en 1959 par Alfred Inselberg.
Senior Fellow San Diego Supercomputing Center and Computer Science and Applied Mathematics Departments Tel Aviv University, Israel



- Conflict Resolution, One-Shot Problem and Air Traffic Control, 1st Canadian Conf. on Comp. Geom., 1989, 26-9

Les outils

Picviz fournit:

- **Scripts perl**: des outils pour traduire des logs en PCV
- **pcv**: un binaire pour transformer le PCV en image
- **picviz-gui**: une interface graphique pour interoger les lignes

Utilisation

Source PCV

```
header { title = "Ossir"; }
axes {
    timeline t;
    integer  in;
}
data {
   t="14:42", in="12" [color="red"];
   t="14:45", in="432";
}
```

Générer l'image

```
pcv -Tplplot fichier.pcv 'filtre'
```

Filtrer

- Filtrer les points: show only plot > 250 on axis 2
- Filtrer les points: show except plot > 50% on axis 2
- Filtrer les chaînes: hide only value = ".*[fF]oo.*" on axis 1

Types d'axes

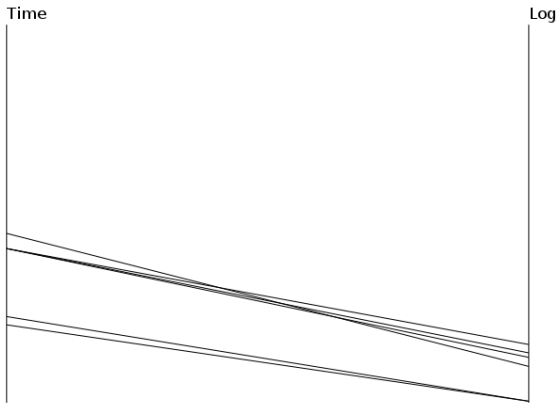
- Temps: timeline, years
- Nombres: integer, short, gold, char
- Adresses: ipv4, ipv6
- Chaînes: string

Time matters

- Mettre la variable á l'échelle
- Une représentation sur 24h :
  - Permet de voir l'heure à laquelle tous nos évenements se sont produits
  - Empêche de différencier les jours
- En montrant de images de mes logs, les gens voient l'heure à laquelle je me couche et me lève :-)

Positionner des chaînes de caractères

Algorithme basique :

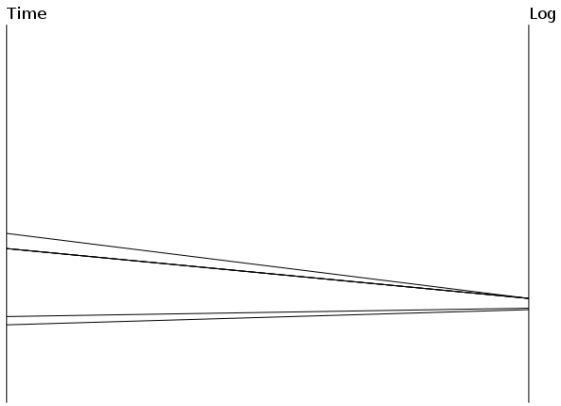


Logs

```
ab
ba
invalid user carlabru
invalid user blingbling
invalid user admin
invalid user root
```

Positionner des chaînes de caractères

Algorithme sur préfixe :

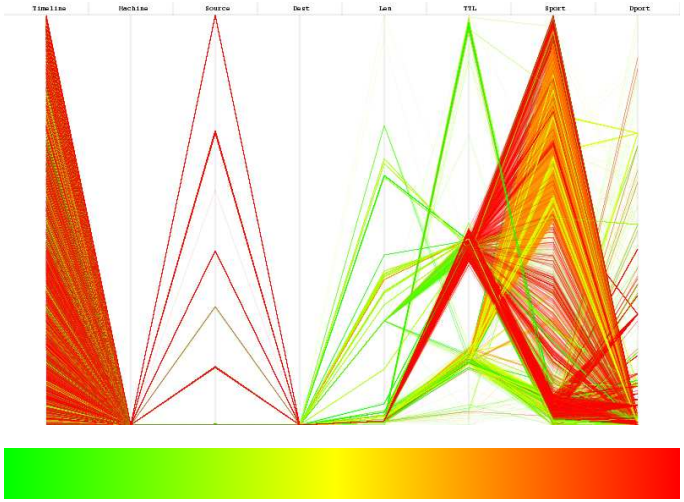


Logs

```
ab
ba
invalid user carlabru
invalid user blingbling
invalid user admin
invalid user root
```

Heatlines

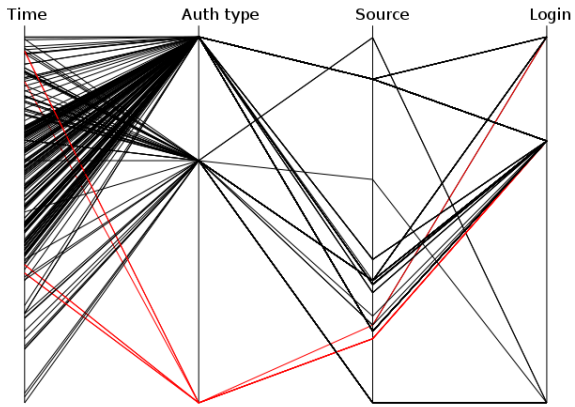
Plus la ligne revient, plus elle tend vers le rouge



Picviz::Dshield

```
use Picviz::Dshield;
$dshield = Picviz::Dshield->new();

if ($dshield->ip_check("192.168.1.42")) {
    print "IP found";
} else {
    print "IP not found";
}
```

SSH authentication



Artcor.pl

- Script simple fait à partir de constats sur les PC
- Vérifications IP, port avec Dshield
- Authentifications depuis multiples adresses IP
- Plusieurs types d'authentifications

Questions ?

- S. Tricaud - stricaud@inl.fr - http://www.gscore.org/blog
- P. Chifflier - chifflier@inl.fr - http://www.wzdftpd.net/blog