How Legos(tm) can inspire Intrusion Detection Systems

Pierre Chifflier Sébastien Tricaud

INL
101/103 Bvd MacDonald
75019 Paris, France

RMLL 2008

What you never wanted to know about IDS

What are IDSs?

- Intrusion Detection Systems
- Marketing folks may call it
  - Intrusion Prevention System (IPS)[1]
  - Security Information and Event Management (SIEM)
- Since IPS and SIEM sound too 2005, we stick to IDS

[1]To prevent an attack, we should first detect it ;)

What are they?

- Host IDS (HIDS): Not (really) prone to false positives
- Network IDS (NIDS): Cannot decrypt unknown encrypted traffic, is **not** the target machine and sensitive to false positives
- Security Information Managers (SIM): Mixes HIDS and NIDS, correlates and give feedback to users.

Interesting sources of information out there

Why do we keep our interest in **Hybrid IDS** when we have **more than just NIDS and HIDS** ?

Interesting sources of information out there

Why do we keep our interest in **Hybrid IDS** when we have **more than just NIDS and HIDS** ?

Low Level Sources:

- **Routers**: Cisco, Linksys, Juniper, . . .
- **Firewalls**: Netfilter, NuFW, Checkpoint, pf, . . .
- **Operating systems**: System logs, users, running applications, . . .
- **Physical**: Alarm, . . .

Interesting sources of information out there

Why do we keep our interest in **Hybrid IDS** when we have **more than just NIDS and HIDS** ?

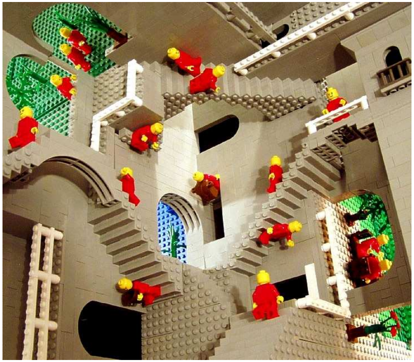
Low Level Sources:

- **Routers**: Cisco, Linksys, Juniper, . . .
- **Firewalls**: Netfilter, NuFW, Checkpoint, pf, . . .
- **Operating systems**: System logs, users, running applications, . . .
- **Physical**: Alarm, . . .

High Level Sources:

- **Honeypots**: Nepenthes, . . .
- **Network**: Snort, Sancp, NuFW, . . .
- **Host**: Auditd (SELinux), Linux PAM, Samhain, Ossec, Prelude LML, ClamAV . . .
- **Scanners**: Nessus, p0f, nmap . . .

Alerting

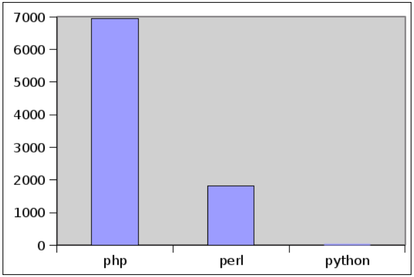


Examples of alerts :

- 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`

Comprendre notre environnement

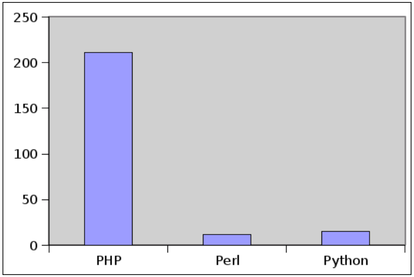
Contexte des vulnérabilités détectées
Janvier 1970 - Juin 2008



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

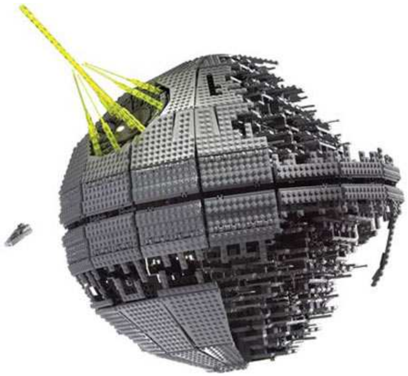
Comprendre notre environnement

Contexte des vulnérabilités détéctées
Janvier 1970 - Juin 2008

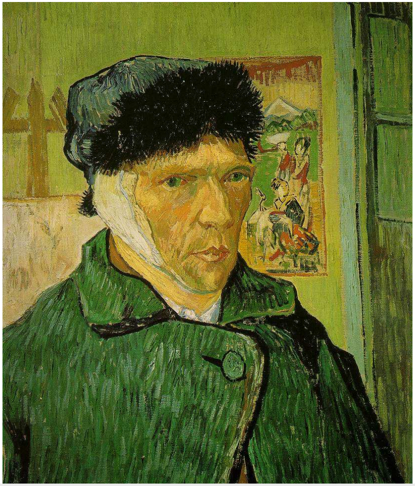


Recherche de tous les CVE touchant directement le language sur http://nvd.nist.gov/

Les menaces actives



Les menaces passives



NIDS requirements

- Capture
- Defragmentation
- Protocol decoder (RPC, UTF-8, HTTP, ...)
- Pattern Matching
- Analysis
- Alerting

Defragmentation:

- Major issue: explained but **not detailed** in the RFC
- NIDS != Target machine
- Mandatory at the Data-Link layer (pcap)
- Should not be done at the Network layer: your OS is nice

Focus on Linux:

- IPV4: linux-src/net/ipv4/ip_fragment.c
- IPV6: linux-src/net/ipv6/reassembly.c

Focus on IPv4:

- Defragmentation is handled by ip_defrag()
- Called only by:
  - ip_local_deliver()
  - ip_call_ra_chain: only if the socket is bound to an interface

Defragmentation

- Linux **does not** defragment on FORWARD
- Netfilter does upon request
- modprobe nf_conntrack_ipv4

Boyer-Moore

- search pattern with the input starting from the rightmost character of the search pattern
- Best case: complexity O(n/m)
- single signature (multi with Setwise-BM)

E2xB

- A Domain-Specific String Matching Algorithm for Intrusion Detection
- designed for providing quick negatives:
  - most of the time a machine **is not under attack**
  - if attacked, the attack pattern is usually in **one packet**

Reference

http://www-igm.univ-mlv.fr/ lecroq/

Finding the evil

Snort offers several ways to match a pattern:

- **Binary**:
  `content:"|0A 00 00 01 85 04 00 00 80|root|00|"` (sid:1775)
- **Simple pattern**:
  `content:"fuck fuck fuck"` (sid:1316)
- **PCRE**:
  `pcre:"/ˆ x3c(REQIMG|RVWCFG) x3e/ism"` (sid:2460)

Problem: How it handles pattern matching algorithms along with PCRE ?

Finding the evil

- Long patterns are easier to find
- PCRE and pattern matching in Snort engine:
  - Load the longest pattern matching option of each signature in the Multi-Pattern Search Engine
    - fpAddLongestContent() function from fpcreate.c
  - Traffic goes through MPSE for pre-qualification
  - Rules sequentially tested
  - PCRE option ignored until full rule test **after** pre-qualification
- PCRE uses its own DFA/NFA

-> Because of this, the less PCRE we have, the best we are.

HIDS features

Analyzing:

- Processes
- Files
- Syscalls
- Logs
- . . .

Logs Analysis

- Match logs in various formats
  - Apache
  - Syslog
  - ...

OSSEC regex to detect SQL injections

```
<rule id="31103" level="6">
  <if_sid>31100</if_sid>
  <url>='|select%20|select+|insert%20|%20from%20|%20where%20|union%20|</url>
  <url>union+|where+|null,null|xp_cmdshell</url>
  <description>SQL injection attempt.</description>
  <group>attack,sql_injection,</group>
</rule>
```

Bypass your HIDS

Log injector trivial code

```
#include <syslog.h>

int main(void)
{
  openlog("sshd", LOG_PID, LOG_AUTH);
  syslog(LOG_NOTICE,
         "pam_unix(su:session): session opened for user root by toady(uid=0)");
  closelog();

  return 0;
}
```

Correlation goal: Transform **alerts** into **attacks**.

Objectives

What ?

- Concentrate on high-level analysis
- Reduce noise created by false positives or harmless events
- Fight evasion
- Discover new attacks

Objectives

What ?

- Concentrate on high-level analysis
- Reduce noise created by false positives or harmless events
- Fight evasion
- Discover new attacks

How ?

- Use trust score to improve the reliability
- Combine elements from heterogeneous sources (use the **Meta**-IDS !)
- Reconstruct and understand the attack

Trust score (TS)

$$\boxed{TS = \text{severity of the alert} \times \text{accuracy of the alert}}$$

- 0 (false alarm) $< TS <$ 1 (known and verified attack)
- Initial value depending on the alert (analyzer and signature reliability)
- NIDS: high probability of false alerts $\Rightarrow$ low TS
- Will be adjusted during correlation steps
- Will be used to take the final decision

Understand an attack

Objectives :

- Reconstruct the sequence of events
- Detect the targets, protocols, tools, . . .
- Adapt the severity
- Reduce false positives
- Prepare for an eventual counter-measure
- Ensure the Security Policy is properly applied

Understand an attack

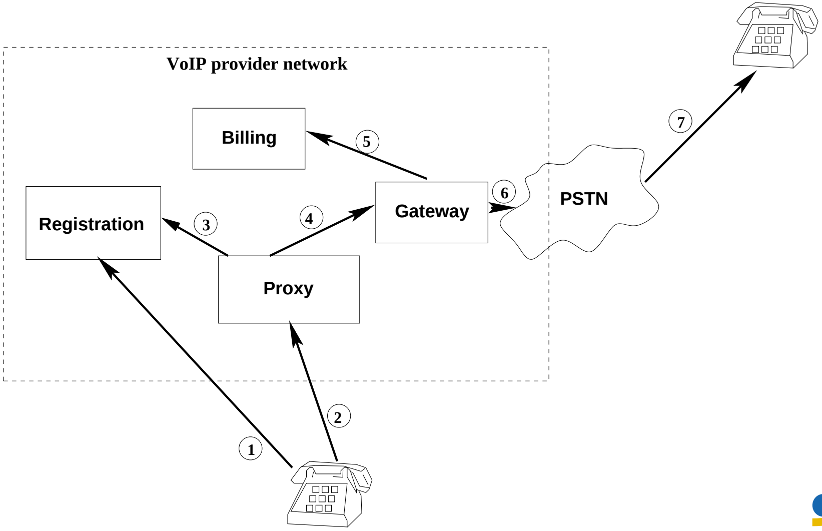
Objectives :

- Reconstruct the sequence of events
- Detect the targets, protocols, tools, . . .
- Adapt the severity
- Reduce false positives
- Prepare for an eventual counter-measure
- Ensure the Security Policy is properly applied

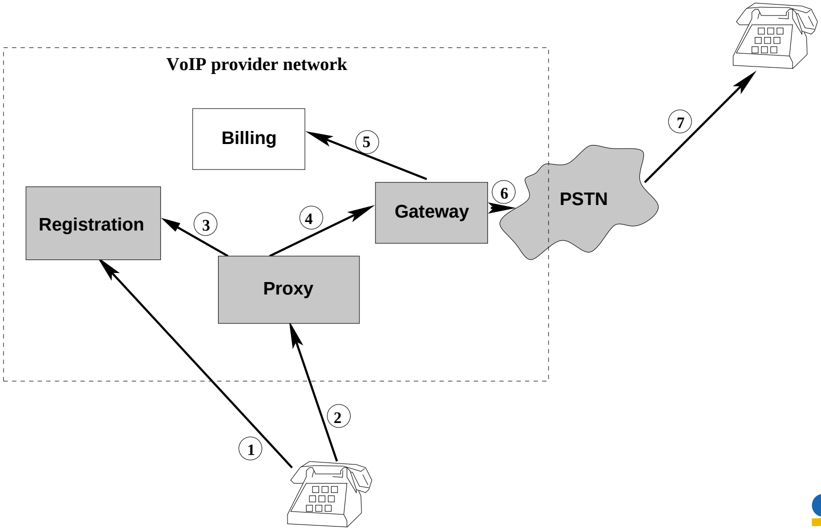
Tools:

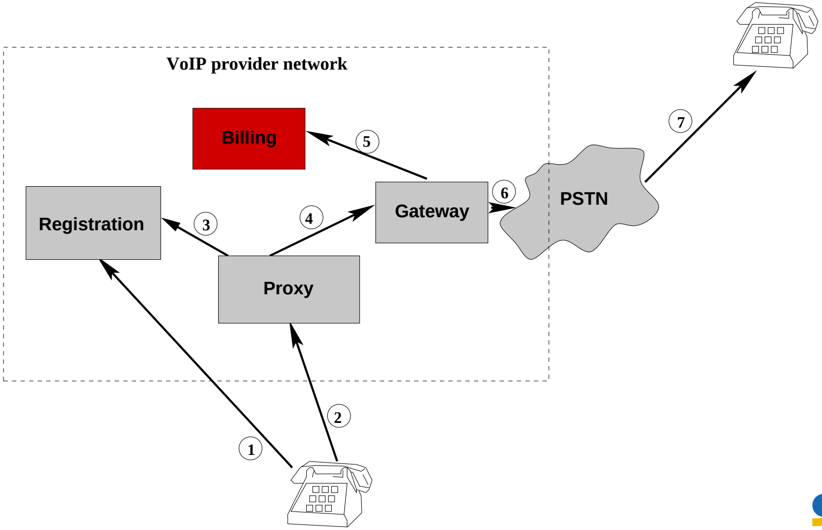
- Normalization, Centralization
- **Correlation**
- **Visualization**

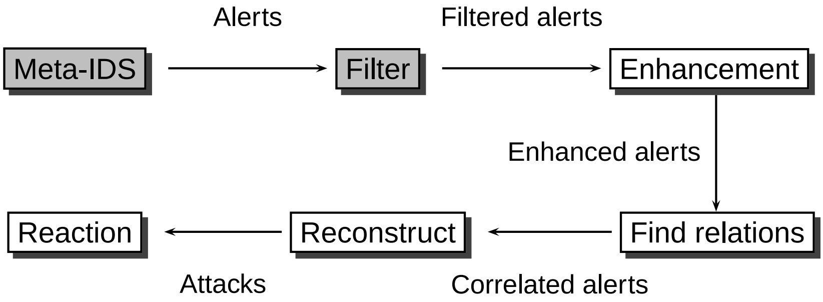
Correlation



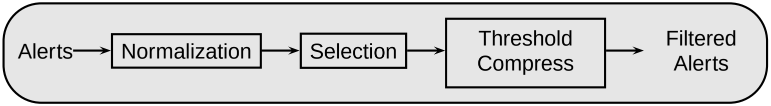
Correlation



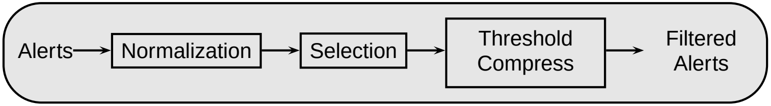

VoIP provider network
Billing
Registration
Gateway
Proxy
PSTN
1
2
3
4
5
6
7


Alerts
Filtered alerts
Meta-IDS
Filter
Enhancement
Enhanced alerts
Reaction
Reconstruct
Find relations
Attacks
Correlated alerts

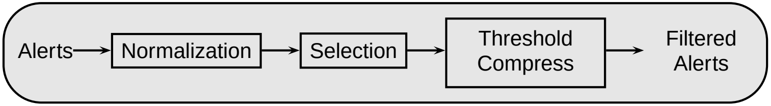
Filtering



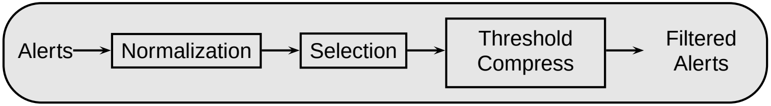
- Normalize input (*classification.text*, *analyzer type*)
- Apply initial filtering
- Compression: replace *n* alerts by one, keeping all information
- Threshold: if $n > threshold$, ignore other alerts (loosing information)



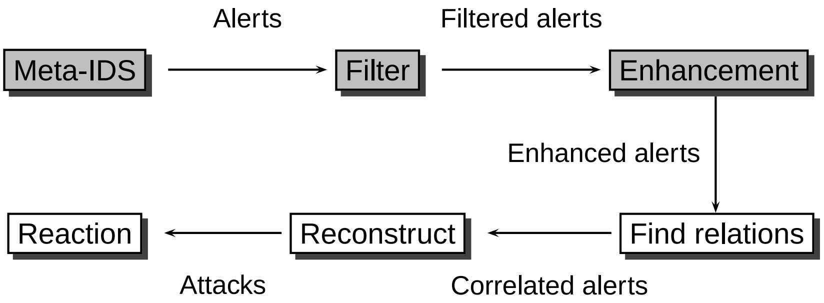
Alert	Filtered alert
SSHD authentication success	User login attempt completion: success



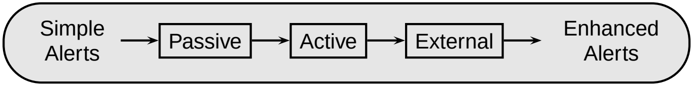
Alert	Filtered alert
SSHD authentication success	User login attempt completion: success
User login failed (Alice) User login failed (Alice)	User login attempt (2 × Alice) completion: failed



Alert	Filtered alert
SSHD authentication success	User login attempt completion: success
User login failed (Alice) User login failed (Alice)	User login attempt (2 $\times$ Alice) completion: failed
User login successful (Alice)	*dropped*


Alerts
Filtered alerts
Meta-IDS
Filter
Enhancement
Enhanced alerts
Reaction
Reconstruct
Find relations
Attacks
Correlated alerts

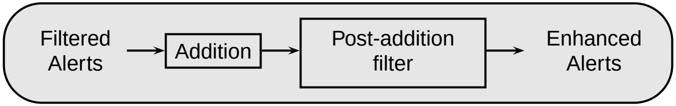
Enhancement (enlarge your alerts)



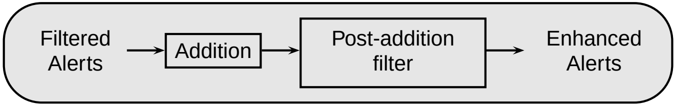
Passive Infomation Collection (PIC):

- Passive data (OS, applications, versions, inventory)
- Profiling (sancp)
- OSVDB, BID, CVE, patches, known exploits
- Current attacks (DShield)
- Passive . . . or not ! (*hint: Nessus*)

Post-enhancement filter



- Send alerts on spurious changes
- Re-evaluate alert with additional data
  - Delete alert or lower trust score if the target is not affected
  - Increase trust score if affected

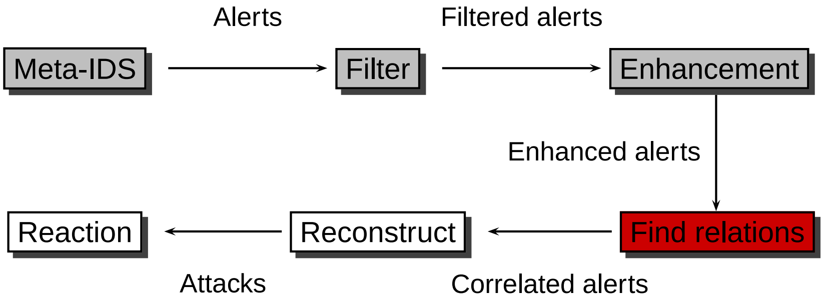


Filtered alert	Enhanced alert
"THCIISLame IIS SSL Exploit Attempt"	"THCIISLame IIS SSL Exploit Attempt" Host OS: Linux 2.6.24 Reference: isc.sans.org/diary.php?date=2004-07-17 Exploit www.thc.org/exploits/THCIISSLame.c *dropped*

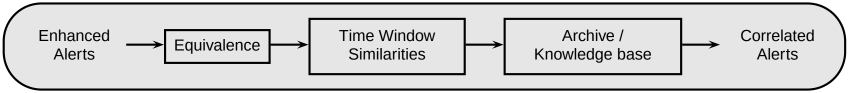
Attack definition

- An attack is a sequence of alerts or events with a particular relation
- $Attack = n \times$ alerts
- $n \geq 1$
- Classification of the *attack* can be done *after* the entire correlation

Find relations

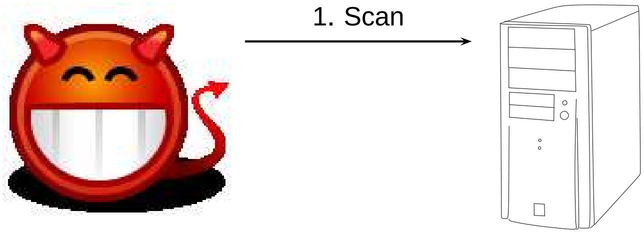


Find relations

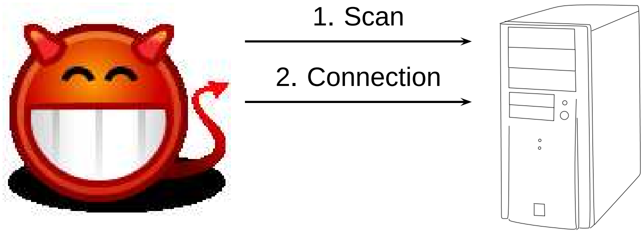


- Equivalence
- Similarities, during a time window (source, destination, attack vector, . . .)
- Archive / knowledge database (known patterns)
- Search on a long time range
- Regular events

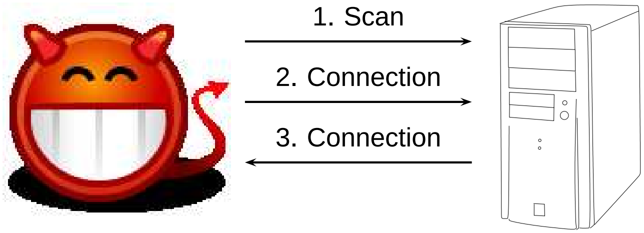
Find relations



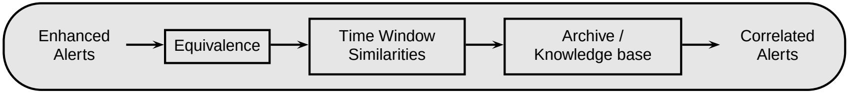
Find relations



Find relations

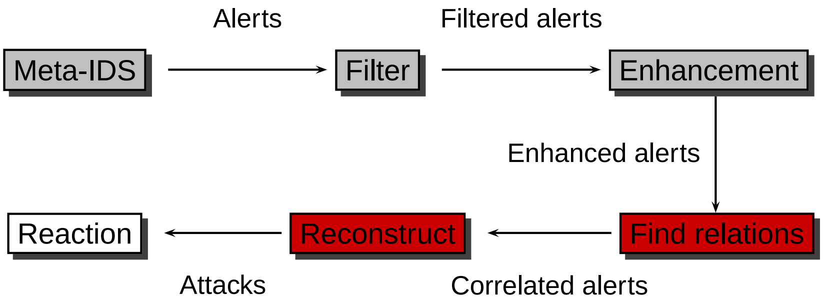


Find relations

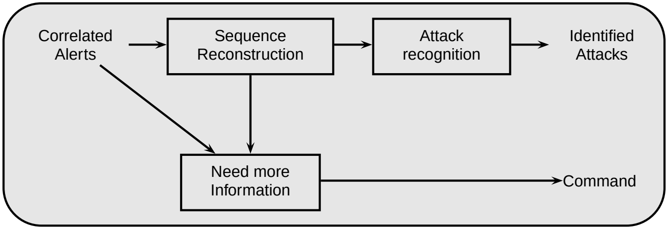


Enhanced Alert	Correlated alert
Port scan + Incoming connection + Outgoing connection *source/dest*	Sequence 3 elements
OSSEC SSHD authentication success (Alice) + Prelude LML User login successful (Alice)	SSH login attempts (1 $\times$ Alice)

Find relations

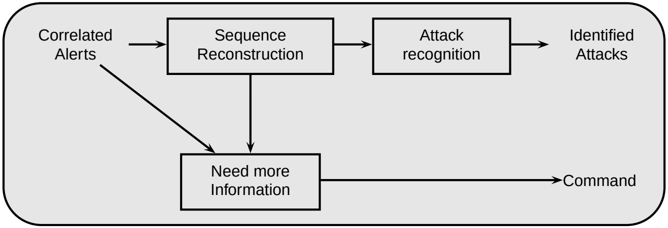


Attack reconstruction



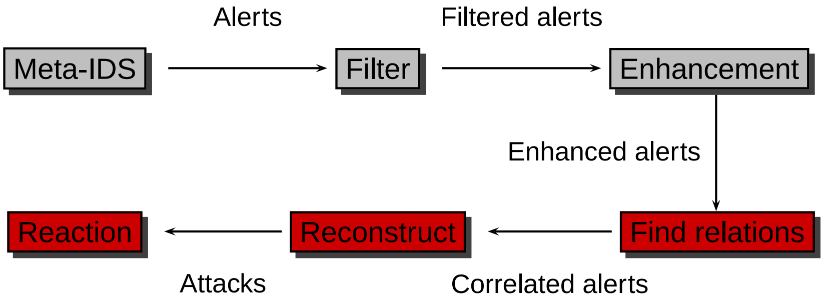
- Try to reconstruct the attack (events and timeline)
- Match vs patterns of known attacks

Find relations

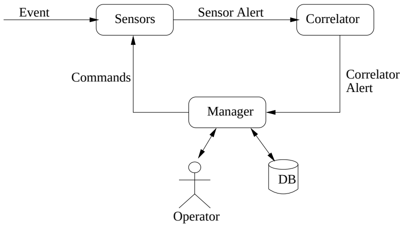


Correlated Alert	Attack
Sequence: Scan + Incoming connection + Outgoing connection	Attack High success probability *known pattern*

Find relations



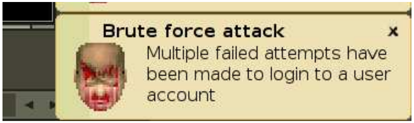
Trust Score evaluation



- Attack is reconstructed and identified
- Trust Score is part of the decision to react
- Ability to capture the whole session by sending commands to agents

Reaction

- Report problem (mail)
- Archive
- Prepare a visualization
- Counter-measure
  - (try to) block attack (*dangerous !*)
  - Collect more information
  - Send commands to agents
- Notify


Brute force attack
Multiple failed attempts have been made to login to a user account

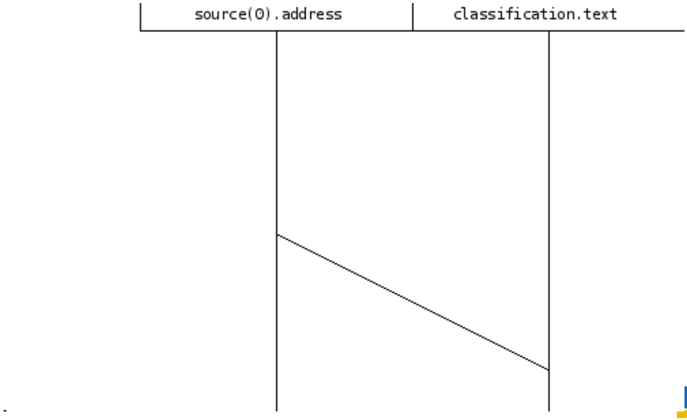
IDS visualization

- Required to manage large amount of data
- Helps to focus on what is important
- Uses the human correlation engine
- Helps to write correlation signatures

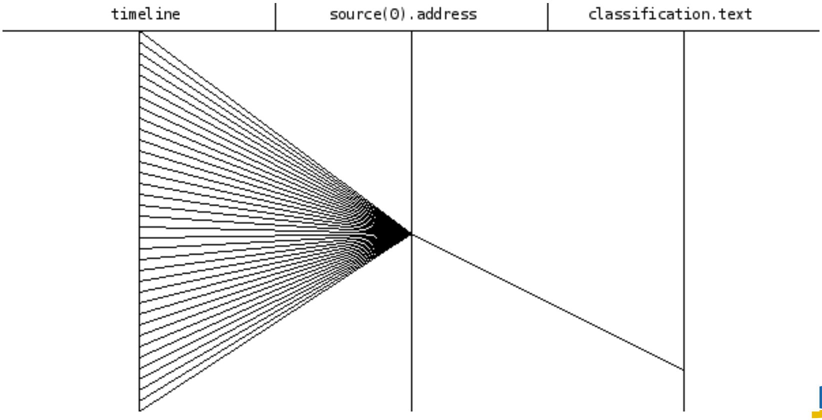
Problem

- Alert are complex objects
- Numerous criteria (N-dimensional plot)
- How to graph correctly?

Visualization dilemma: take the right parameters for the right graph



Visualization dilemma: take the right parameters for the right graph



Relevant parameters from IDMEF paths

- Source (*alert.source(0).node.address(0).address*)
- Destination (*alert.target(0).node.address(0).address*)
- Impact (*alert.assessment.impact.severity*)
- Completion (*assessment.impact.completion*)
- Attack vector (*alert.classification.text*)
- Agent type (*analyzer(0).class*)

Code 1/3

- Based on Prelude IDS
- High-level language
- Python + Prelude Easy bindings

```
svn co http://svn.prelude-ids.org/libprelude/
    branches/libprelude-easy-bindings
```

Code 2/3

How to get alerts

```
from PreludeEasy import *

client = ClientEasy("pig", Client.IDMEF_READ)
client.AddConnection("192.168.33.215")
client.Start()
idmef = client.RecvIDMEF()
```

Code 3/3

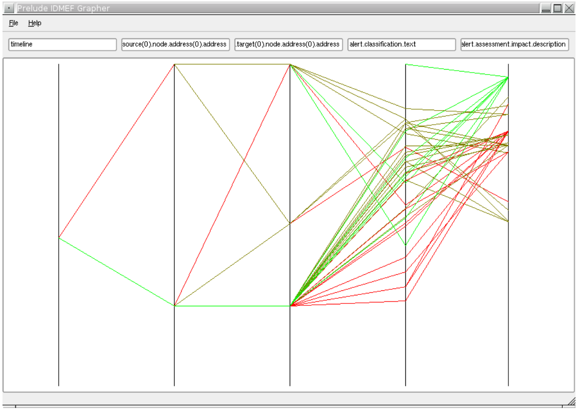
Graph Objects (GO!)

```
pen = QtGui.QPen()
pen.setColor(colorize_impact_severity(idmef))

line1_y = GetYPos(
      idmef.Get("alert.target(0).node.address(0).address"))
line2_y = GetYPos(
      idmef.Get("alert.classification.text"))

scene.addLine(
      line1_x, line1_y,
      line2_x, line2_y,
      pen)
```

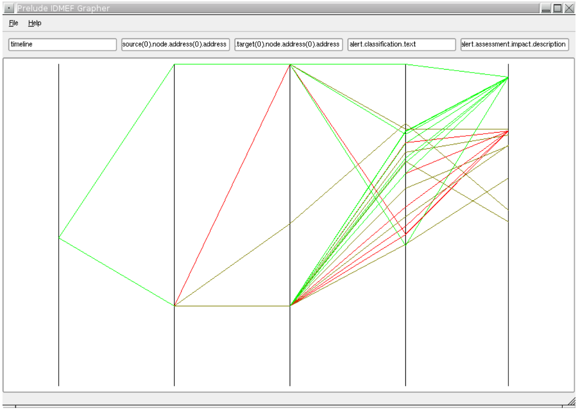
Prelude IDMEF Grapher (pig)

- Shows IDMEF paths
- Uses Prelude IDMEF pool
- Interesting to quickly understand a scanner
- Snort and LML are used as agents

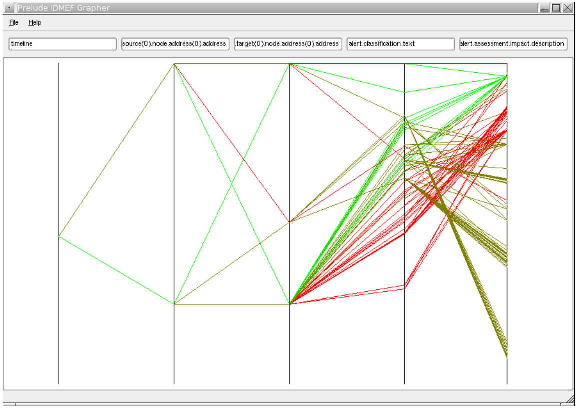
Saint: 166 alerts generated


Prelude IDMEF Grapher
File
Help
timeline
source(0).node.address(0).address
target(0).node.address(0).address
alert.classification.text
alert.assessment.impact.description

Retina: 76 alerts generated

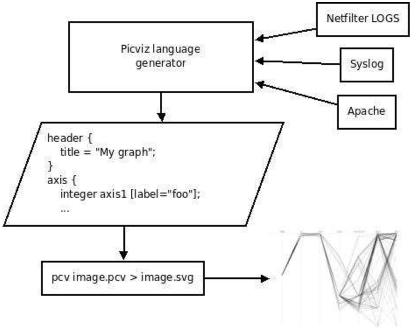

Prelude IDMEF Grapher
File
Help
timeline
source(0).node.address(0).address
target(0).node.address(0).address
alert.classification.text
alert.assessment.impact.description

Nessus: 1019 alerts generated


Prelude IDMEF Grapher
File
Help
timeline
source(0).node.address(0).address
target(0).node.address(0).address
alert.classification.text
alert.assessment.impact.description

Automate the image generation with Picviz

```
svn co https://picviz.svn.sourceforge.net/
    svnroot/picviz/trunk picviz
```



```
header {
    title = "My graph";
}
axis {
    integer axis1 [label="foo"];
    ...
```

Remerciements

- Yoann Vandoorselaere
- Pablo Neira Ayuso
- INL staff

Questions ?