Towards Integrated Malware Defence

Dr. Morton Swimmer
Trend Micro, Inc
(Formerly: John Jay College of Criminal Justice/CUNY)

Disclaimer:
This presentation is based on research done at John Jay and does not necessarily represent the opinion of my current employer

Threat Landscape

- Threats have been modularized
- Different groups specialize on different technologies
- Clearly there is a criminal ecosystem at work here

- Attacks are often multi-pronged
- It's not just the operating system
  - any application is game
- There is no silver bullet

- Furthermore, attacks are more frequent
- They are more targeted
- They have minimised the time between vulnerability discovery and abuse

The questions are now:

How can we build an effective defence infrastructure?

How can we get our products to work together?

What model can we use for product interactions?

Top-down

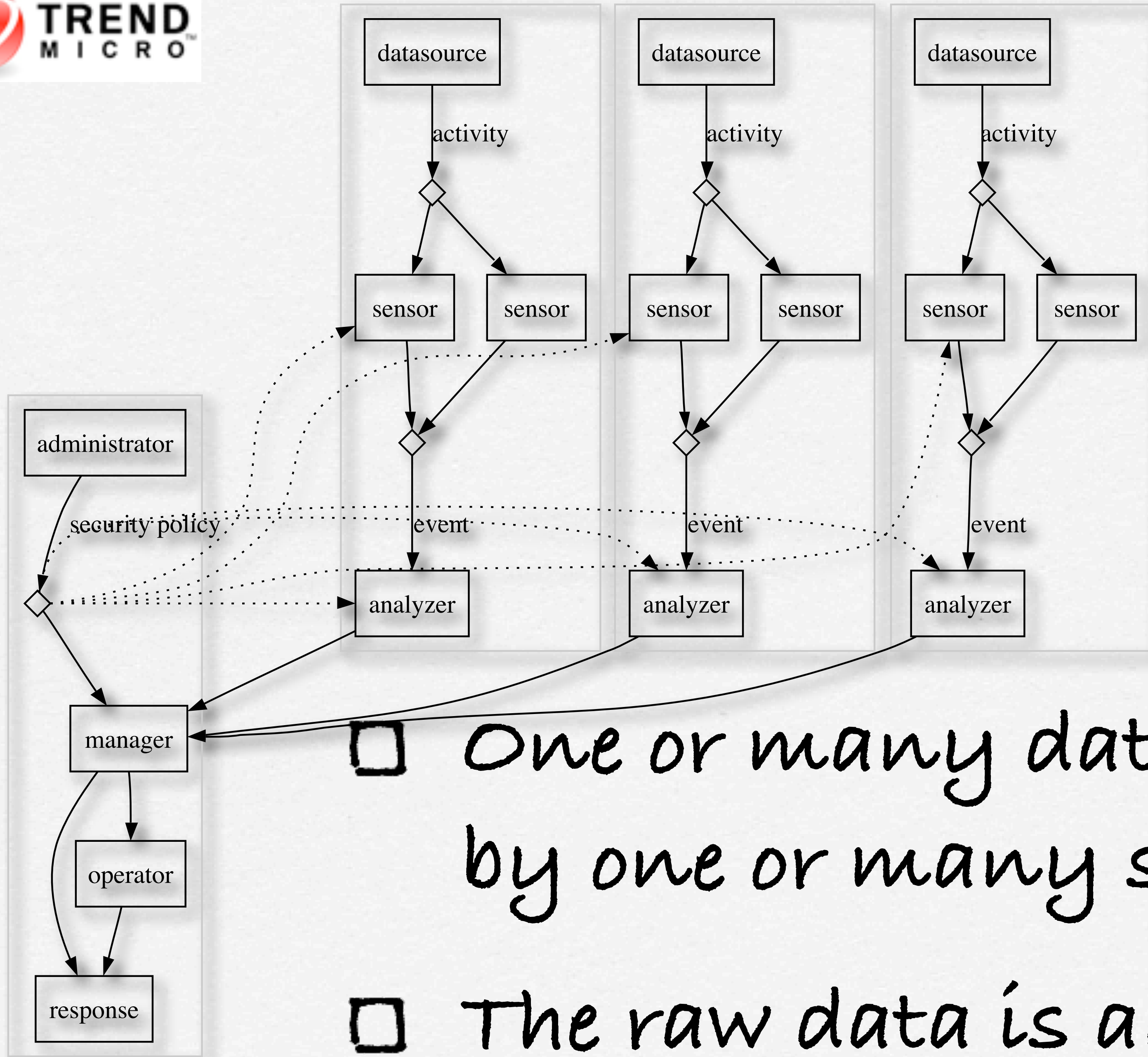
- We will look at a few models
- Then look at how semantic alerts can enable a more intelligent approach

IDS architectures

- Classic IDS infrastructure, according to IEFT/IDWG
- Autonomous IDS infrastructure, for example ADN

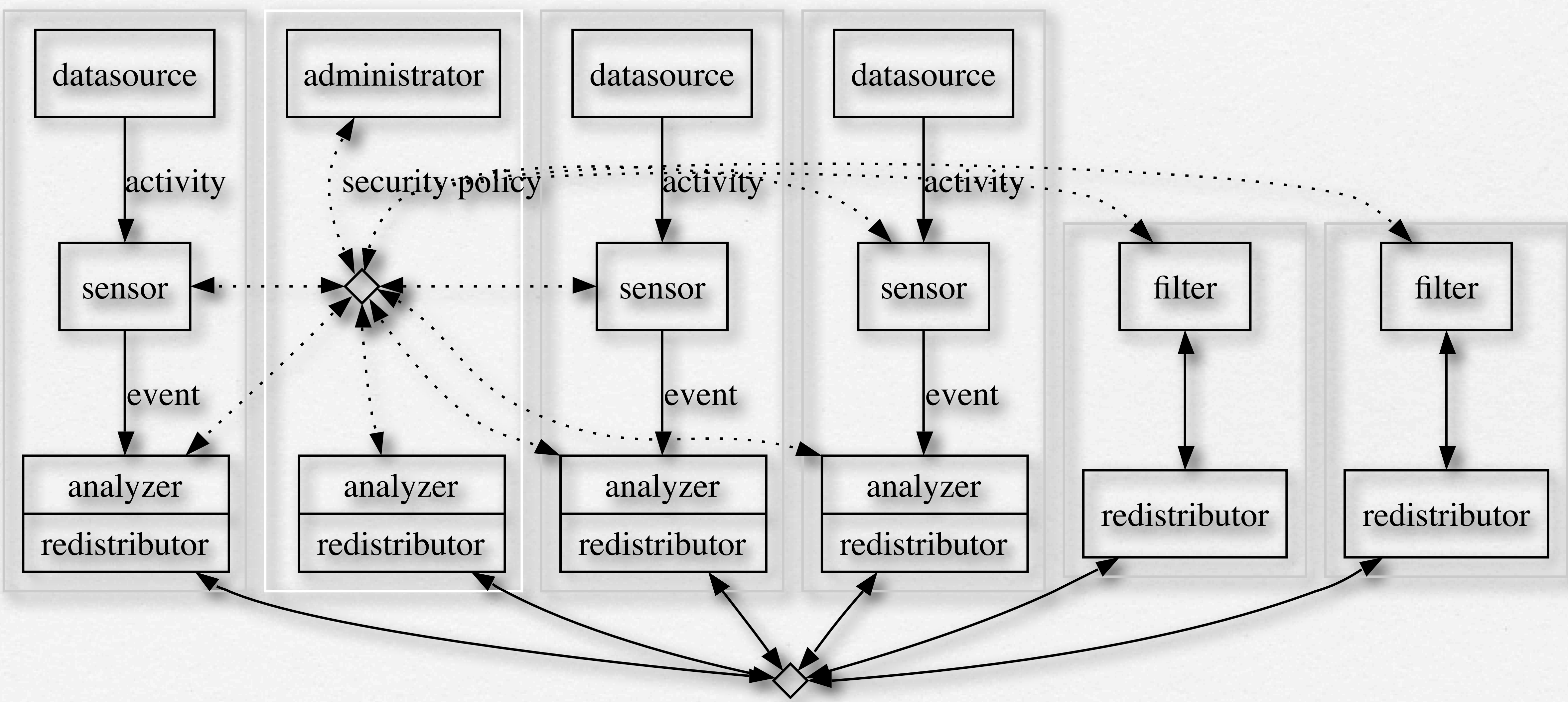
AntiVirus and Spam filters are considered a form of IDS in this discussion.
It is important that they are!

IETF IDWG IDS Architecture



- One or many data sources monitored by one or many sensors
- The raw data is analyzed and sent on to a manager and operator who may initiate a response

ADN architecture



Discussion

- Classically, we rely on a central authority to correlate, judge and respond
- In ADN, we push tactical decision making down into the infrastructure
- Both rely on some form of alert correlation

Alert processing

- Correlation helps filter the useful from the useless
- But each sensor produces different data
- A sensor may produce data with different meaning in different contexts

IDS alert standard

- IDMEF (RFC 4765)
  - IETF's IDWG alert exchange format
- Semantics not addressed
- Standardizes parsing
- Each source still need individual interpretation


```xml
<?xml version="1.0" ?>
<IDMEF-Message version="1.0">
    <Alert ident="12773">
        <Analyzer analyzerid="snort00" model="snort" ...
        </Analyzer>
        <CreateTime ntpstamp="0xb9225b23.0x9113836a">
            1998-06-05T11:55:15Z
        </CreateTime>
        <Source>...</Source>
        <Target>... </Target>
        <Classification origin="vendor-specific">
            <name>msg=ICMP PING</name>
            <url>none</url>
        </Classification>
        <Classification origin="vendor-specific">
            <name>sid=384</name>
             <url>http://www.snort.org/snort-db/sid.html?sid=384</url>
        </Classification>
        <Classification origin="vendor-specific">
            <name>class=misc-activity</name>
            <url>none</url>
        </Classification>
        <Classification origin="vendor-specific">
            <name>priority=3</name>
            <url>none</url>
        </Classification>
        <Assessment>
            <Impact severity="high" />
        </Assessment>
        <AdditionalData meaning="sig_rev" type="string">
            5
        </AdditionalData>
        <AdditionalData meaning="Packet Payload" type="string">
            2A2A2020202020202020202000AAEA020097A4020075DA
        </AdditionalData>
    </Alert>
</IDMEF-Message>
```

TREND MICRO

Introducing semantics

- Machine understanding is currently impossible
- We approximate this with
  - controlled vocabularies
  - standard data model
- Inspired by Semantic Web

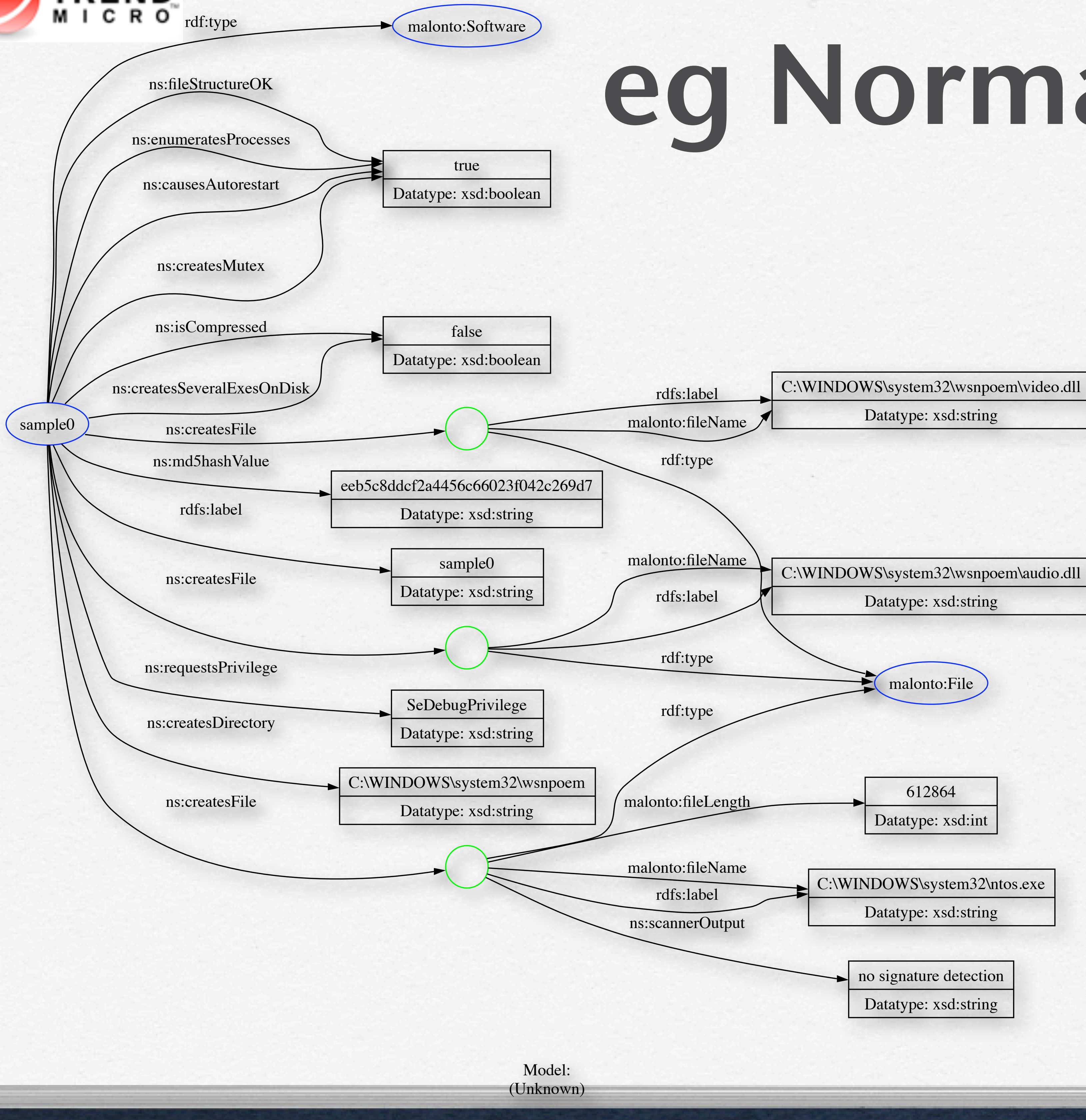
Semantic Web technology

- The puzzle pieces falling into place
- RDFS for simple vocabularies, OWL for ontologies, RDF for descriptions
- RDQL, SPARQL, ... for queries
- Pellet, Racer, FaCT, ... for reasoning

RDF = Resource Description Language

- Technically: a graph
- Composed of triples
  - subject - predicate - object
- Serialized in many forms
  - RDF/XML, N3, Turtle, ...
  - but it is the model that is important

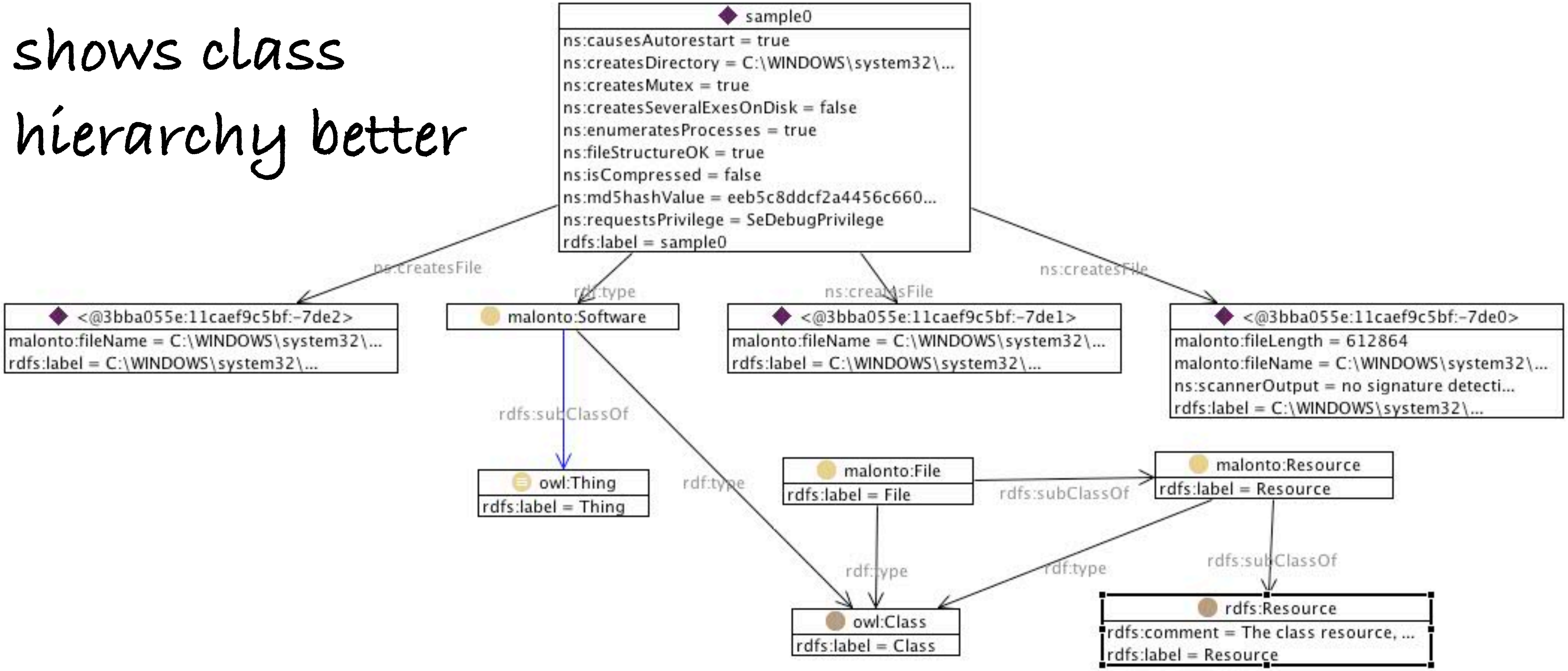
eg Norman Sandbox



- RDF model a malware analysis report
- used as a stand-in for a general Honeypot

Another view

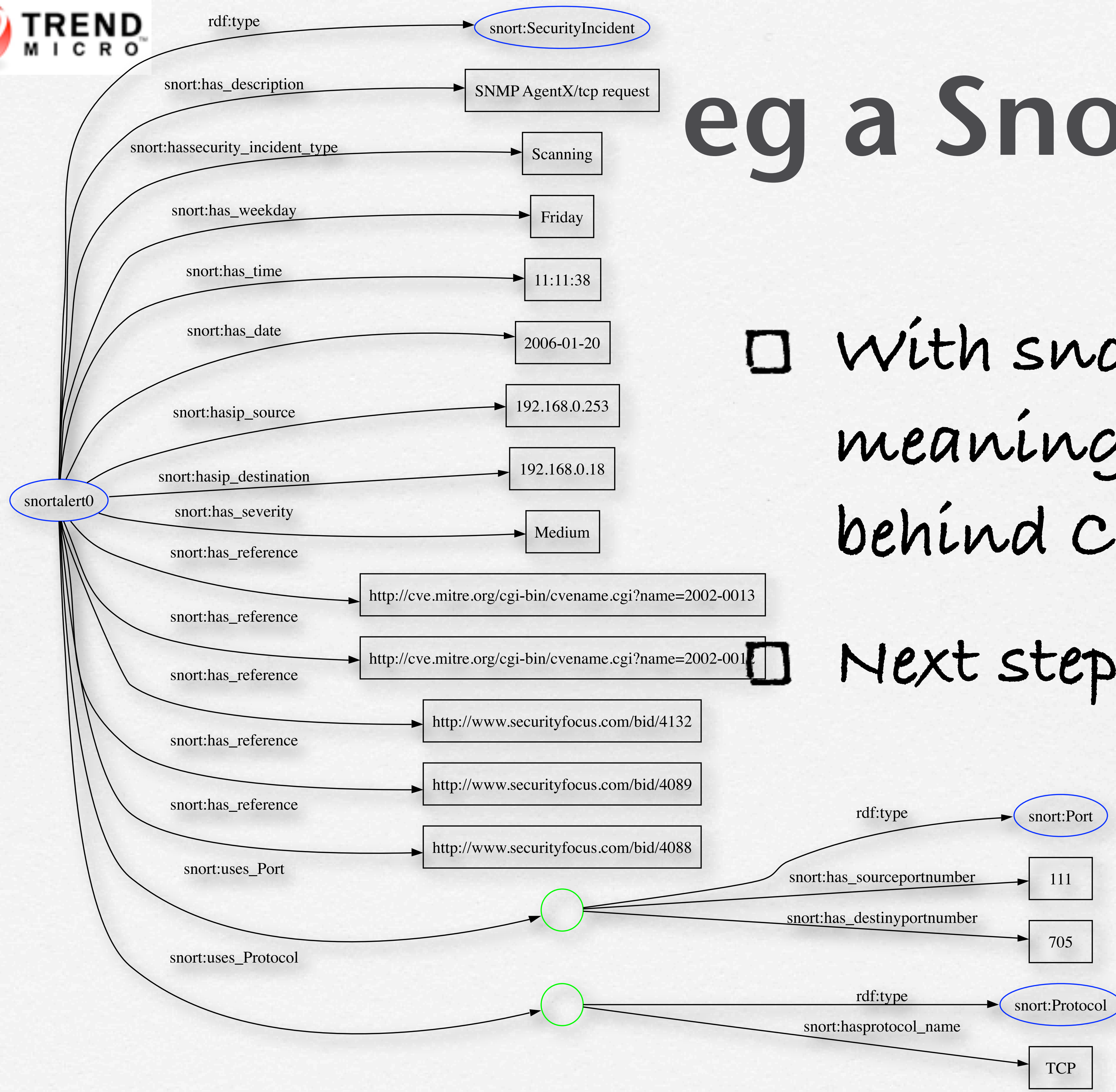
shows class hierarchy better



- The model as Turtle data
- (You don't want to see the RDF/XML)

```
:sample0
      a       malonto:Software ;
      rdfs:label "sample0"^^xsd:string ;
      ns:causesAutorestart
              "true"^^xsd:boolean ;
      ns:createsDirectory "C:\\WINDOWS\\system32\\wsnpoem"^^xsd:string ;
      ns:createsFile
              [ a       malonto:File ;
                rdfs:label "C:\\WINDOWS\\system32\\wsnpoem\\audio.dll"^^xsd:string ;
                malonto:fileName "C:\\WINDOWS\\system32\\wsnpoem\\audio.dll"^^xsd:string
              ] ;
      ns:createsFile
              [ a       malonto:File ;
                rdfs:label "C:\\WINDOWS\\system32\\ntos.exe"^^xsd:string ;
                malonto:fileLength "612864"^^xsd:int ;
                malonto:fileName "C:\\WINDOWS\\system32\\ntos.exe"^^xsd:string ;
                ns:scannerOutput "no signature detection"^^xsd:string
              ] ;
      ns:createsFile
              [ a       malonto:File ;
                rdfs:label "C:\\WINDOWS\\system32\\wsnpoem\\video.dll"^^xsd:string ;
                malonto:fileName "C:\\WINDOWS\\system32\\wsnpoem\\video.dll"^^xsd:string
              ] ;
      ns:createsMutex "true"^^xsd:boolean ;
      ns:createsSeveralExesOnDisk
              "false"^^xsd:boolean ;
      ns:enumeratesProcesses
              "true"^^xsd:boolean ;
      ns:fileStructureOK "true"^^xsd:boolean ;
      ns:isCompressed "false"^^xsd:boolean ;
      ns:md5hashValue "eeb5c8ddcf2a4456c66023f042c269d7"^^xsd:string ;
      ns:requestsPrivilege
              "SeDebugPrivilege"^^xsd:string .
```

eg a Snort alert



- With snort, a lot of the meaning is hidden behind CVE
- Next step: CVE in RDF

```
:snortalert0
    snort:has_date "2006-01-20" ;
    snort:has_description "SNMP AgentX/tcp request" ;
    snort:has_reference
     "http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012",
     "http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013",
     "http://www.securityfocus.com/bid/4088",
     "http://www.securityfocus.com/bid/4089",
     "http://www.securityfocus.com/bid/4132" ;
    snort:has_severity "Medium" ;
    snort:has_time "11:11:38" ;
    snort:has_weekday "Friday" ;
    snort:hasip_destination "192.168.0.18" ;
    snort:hasip_source "192.168.0.253" ;
    snort:hassecurity_incident_type "Scanning" ;
    snort:uses_Port [
        snort:has_destinyportnumber "705" ;
        snort:has_sourceportnumber "111" ;
        a <snort:Port>
    ] ;
    snort:uses_Protocol [
        snort:hasprotocol_name "TCP" ;
        a <snort:Protocol>
    ] ;
    a <snort:SecurityIncident> ;
    rdfs:label "sample0"^^xsd:string .
```

The same as a Turtle file

Combining the data

- We want to be able to query the big picture
- First we need the vocabularies
- Then we need the queries

Vocabularies

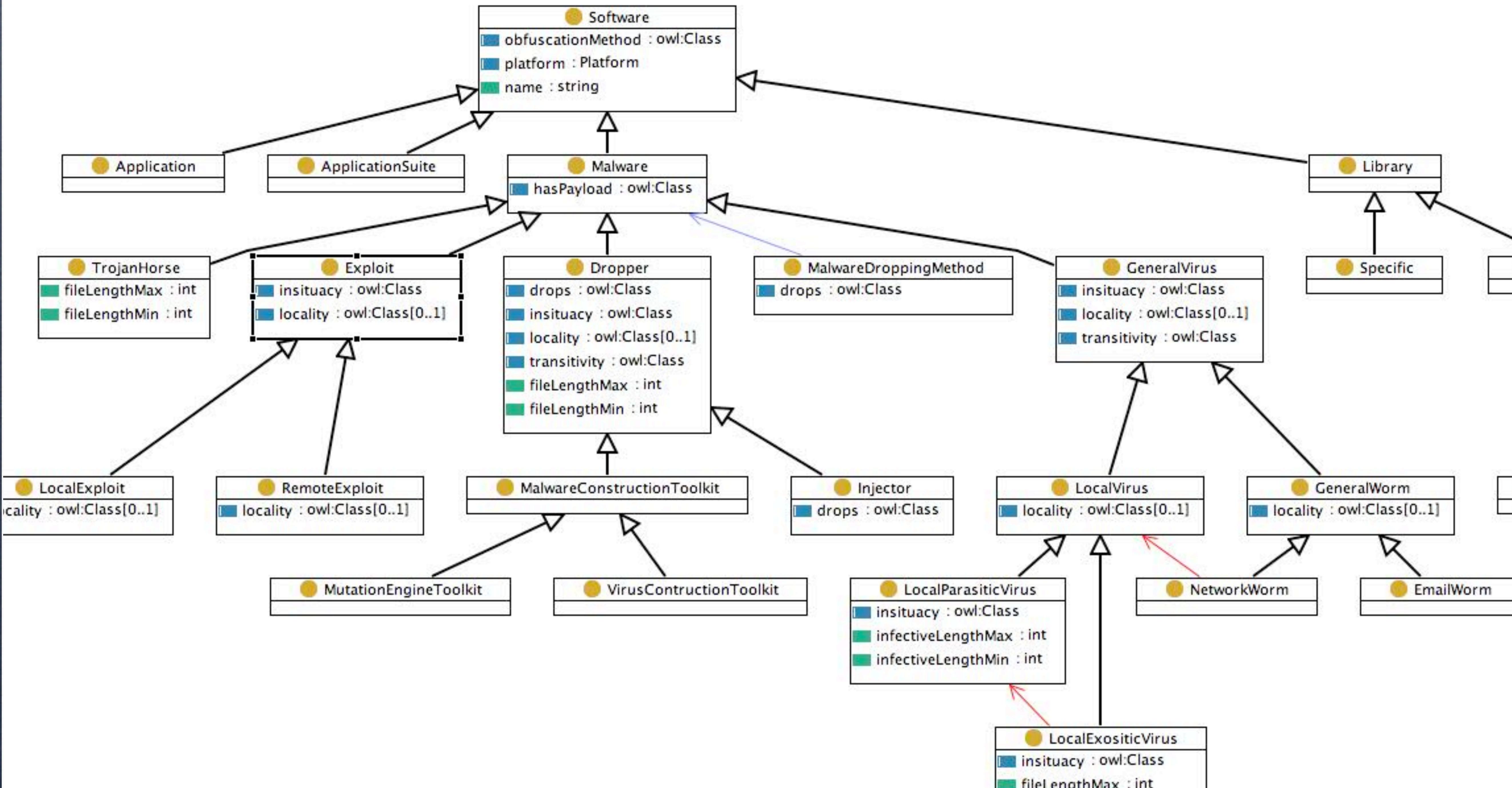
- RDFS = RDF Schema
- Defines Terms (Concepts) and Properties
- Organizes these into simple class hierarchies
- Like a dictionary, it defines a vocabulary that a group of individuals can agree on

Ontologies

- An ontology is a formal explicit description of concepts in a domain of discourse, properties of each concept describing various features and attributes of the concept, and restrictions on properties

- Defined using
    - classes (aka concepts)
    - properties (aka slots or roles)
    - facets (aka role restrictions)
- Ontology classes can be defined by properties and facets alone

Terms from malonto



Using vocabularies

- Most likely, we need a domain specific vocabulary for each sensor
- As much as possible it should be based on an existing and established vocabulary
- I use malonto (my own) and RESIST mainly

Simple Queries

- Query language: SPARQL
- Others
  - RDQL, SeRQL, XsRQL, Versa

```
SELECT ?subject
WHERE {
      ?subject ns:fileStructureOK "true"^^xsd:boolean .
      ?subject ns:isCompressed  "false"^^xsd:boolean .
}
```

sample0

Looking further

```
SELECT ?subj ?alert
WHERE  {
   ?subj ns:networkConnect ?remote.
   ?remote ns:address "www.evil.net"^^xsd:string ;
      ns:port ?target_port.
   ?alert nids:has_severity "Medium".
   ?alert nids:uses_Port ?alert_port.
   ?alert_port nids:has_destination_port_number ?target_port.
}
```

- querying over multiple sources
- looking for results from honeypot targeting an network address and looking for that port in snort logs

Reasoning

- Useful for testing hypothesizes
- Useful for finding root causes
- Important to restrict vocabularies to OWL-DL (Descriptive Logics)
- DL Reasoning can be NPEXPTIME complex, but heuristics are well explored
- Pellet, FaCT, Racer, ...

Caveats

- Finding a good representation alert/report data is hard
- Finding a good vocabulary definition is equally hard
  - and it has to be restricted DL!

RDF/OWL design

give examples

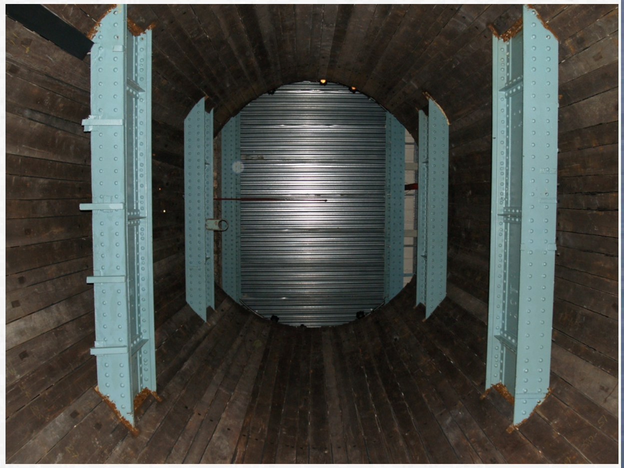
- Start with the end in mind
  - what sort of queries may be wanted
- Base new ontologies on existing established ones
  - RDFS, Cyc, RESIST, ...

- No compelling reason why original data can't remain in native format
- Converters used to map to RDF model
  - Ideally, the converters should be vendor supplied
  - Only the vendor knows the true 'meaning' of their data

What's missing?

- For each sensor deployment, an encoding of its context
  - Provenance data
- Inclusion of other forms of alert data
- Better base ontologies/vocabularies

Conclusions



- Shifting focus from raw alerts to meaningful alerts
- Allows a new level of querying and correlation
- Will first be used to augment existing alert handling systems until rules libraries are complete

- Reasoning systems will be used for hypothesis testing and root cause analysis

- Towards an autonomous network of security subsystems working together

Questions?

- -> morton@swimmer.org