Ovalyzer: an OVAL to Cfengine Translator

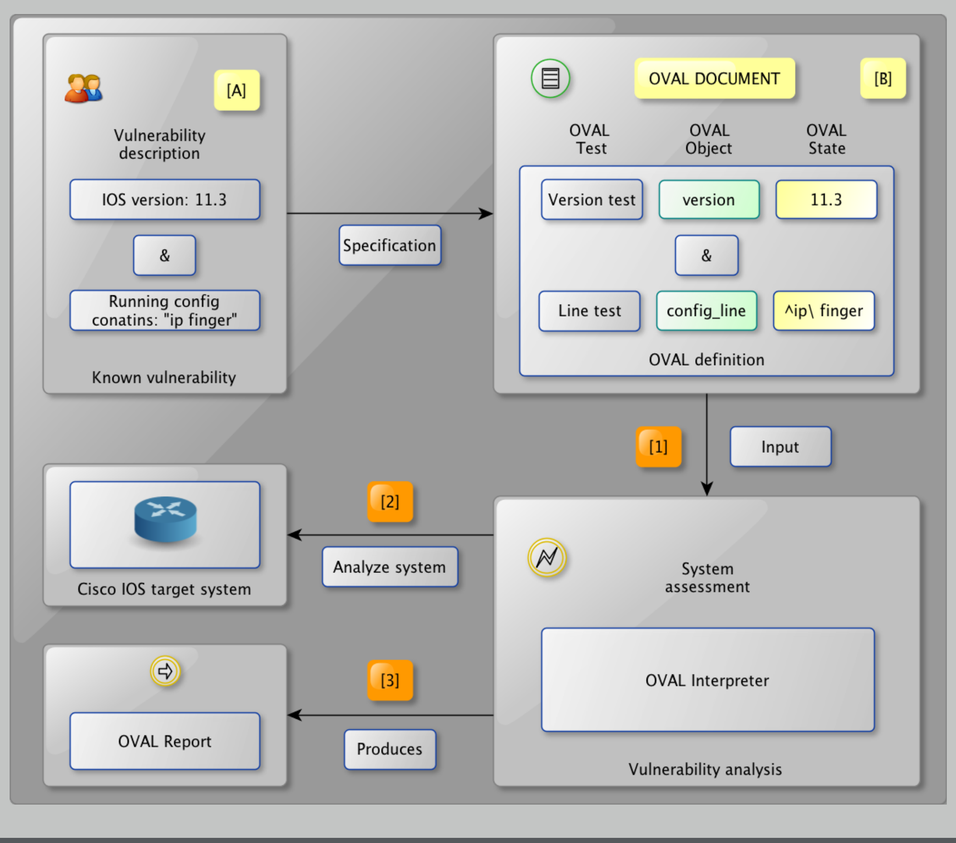
Martín Barrère, Rémi Badonnel and Olivier Festor
Madynes Research Team - LORIA - INRIA Nancy Grand Est, France

1. Introduction

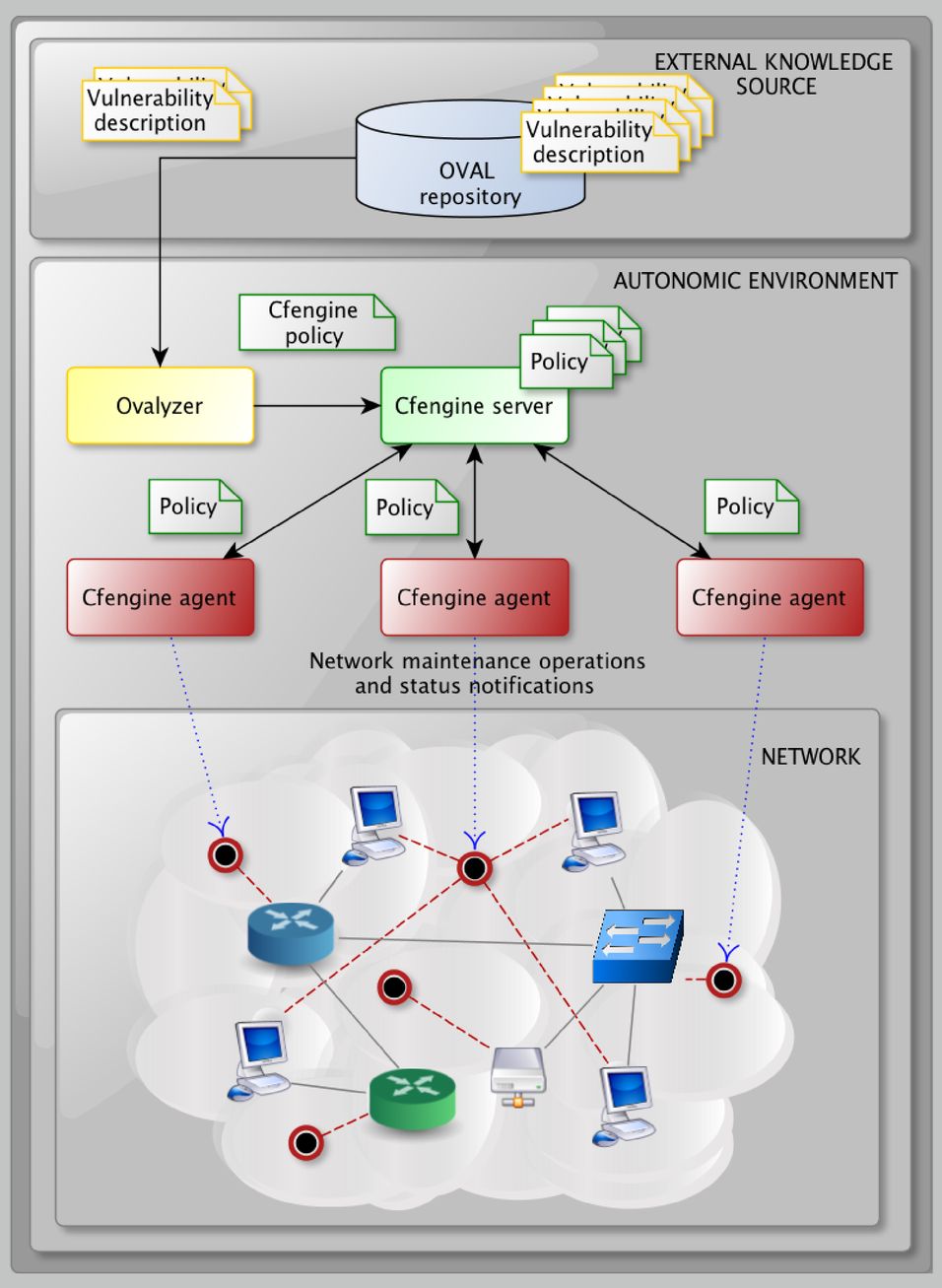
- Growing complexity of networks and systems management
  - ▷ delegation of management functionalities to the networks themselves
  - ▷ but, human errors and autonomic mechanisms may generate vulnerable configuration states
- **How do we increase vulnerability awareness in autonomic networks and systems?**
- We propose a strategy for supporting the integration and assessment of vulnerability descriptions into the autonomic management plane [1]
  - ▷ exploiting OVAL descriptions warning about current threats and system vulnerabilities [2]
  - ▷ considering the widely used Cfengine autonomic configuration system for performing policy-based network and system administration [3]

2. OVAL language overview

- OVAL (Open Vulnerability and Assessment Language) is an XML-based standard language for describing vulnerabilities, analyzing and reporting them, and exchanging security related information

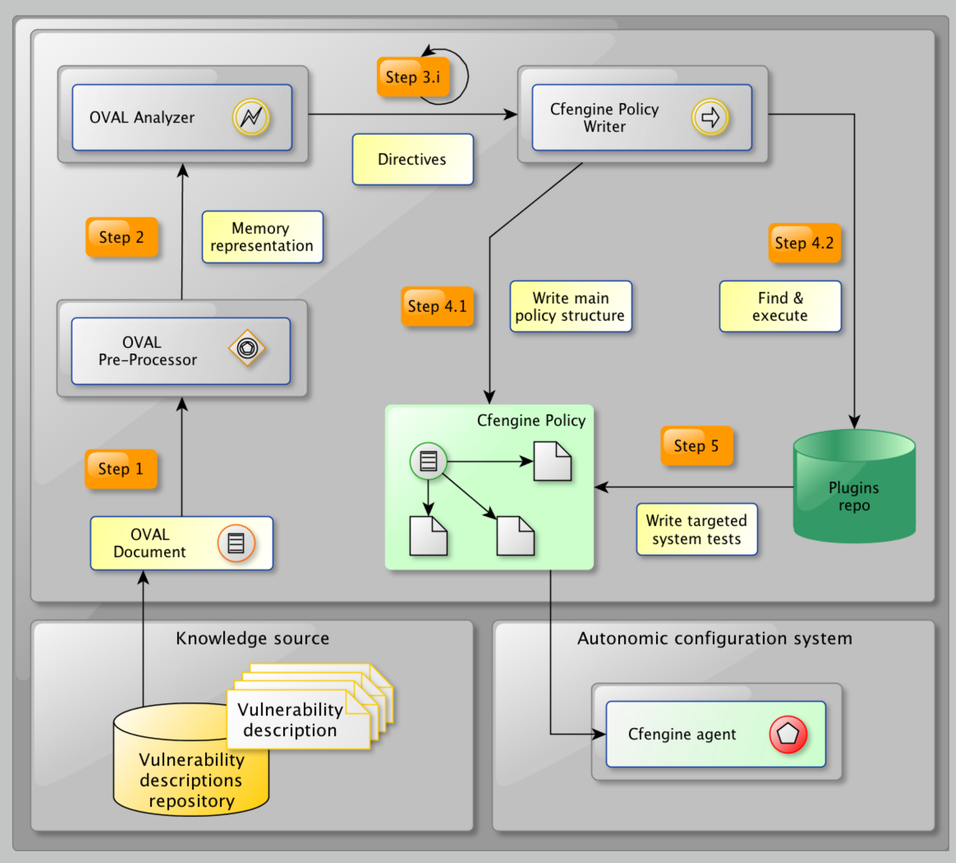


3. Ovalyzer-based approach



4. Ovalyzer functional architecture

- Ovalyzer is an extensible Java-based tool for translating OVAL descriptions to Cfengine policy rules
- Main characteristics
  - ▷ automatic data model regeneration using the JAXB framework [4]
  - ▷ seamless functional extension over a plugin-based architecture
- High-level operational view



5. OVAL to Cfengine rules generation example

OVAL document

```
<?xml version="1.0" encoding="UTF-8"?>
<oval_definitions>
<definitions>
    <definition id="oval:org.mitre.oval:def:15">
      <criteria operator="AND">
          <criterion comment="IOS vulnerable version"
            test_ref="oval:org.mitre.oval:tst:1"/>
          <criterion comment="IP finger service test."
            test_ref="oval:org.mitre.oval:tst:2"/>
      </criteria>
    </definition>
  </definitions>
  <tests>
    <version55_test id="oval:org.mitre.oval:tst:1" >
      <object object_ref="oval:org.mitre.oval:obj:100"/>
      <state state_ref="oval:org.mitre.oval:ste:200"/>
    </version55_test>
    <line_test id="oval:org.mitre.oval:tst:2" >
      <object object_ref="oval:org.mitre.oval:obj:101"/>
      <state state_ref="oval:org.mitre.oval:ste:201"/>
    </line_test>
  </tests>
  <objects>
    <version55_object id="oval:org.mitre.oval:obj:100"/>
    <line_object id="oval:org.mitre.oval:obj:101">
      <show_subcommand> show running-config
      </show_subcommand>
    </line_object>
  </objects>
  <states>
    <version55_state id="oval:org.mitre.oval:ste:200" >
      <version_string operation="pattern match"> 11\.3.*
      </version_string>
    </version55_state>
    <line_state id="oval:org.mitre.oval:ste:201">
      <show_subcommand> show running-config
      </show_subcommand>
      <config_line operation="pattern match">^ip\ finger
      </config_line>
    </line_state>
  </states>
</oval_definitions>
```

Cfengine policy

```
import:
   any::
   oval:org.mitre.oval:def:15

_____________________________


control:
   ...
   definitionId    = ( "oval:org.mitre.oval:def:15" )

   object100 = ( PrepModule(module:retrieveObject,"\
     "show version\" \"out/obj:100\"")  )
   object101 = ( PrepModule(module:retrieveObject,"\
     "show running-config\" \"out/obj:101\"")  )

   actionsequence = ( shellcommands methods )
   ...
   ste_200_versionstring = ( "11\.3.*" )
   ste_201_configline = ( "^ip\ finger" )

shellcommands:
   ...

methods:
   EvalTest1("out/obj:100",${ste_200_versionstring})
   action=oval:org.mitre.oval:tst:1
   returnclasses=ResultTest
   ...

   EvalTest2("out/obj:101",${ste_201_configline})
   action=oval:org.mitre.oval:tst:2
   returnclasses=ResultTest
   ...

alerts:
   every::

   (EvalTest1_ResultTest.EvalTest2_ResultTest)::
   "${definitionId} - Result: TRUE"

   !(EvalTest1_ResultTest.EvalTest2_ResultTest)::
   "${definitionId} - Result: FALSE"
```

6. Conclusions and future work

- Design of an architecture for supporting and automating vulnerability management activities
- Development of Ovalyzer, an extensible OVAL to Cfengine translation system (focus on Cisco IOS related vulnerability descriptions)
- Future work
  - ▷ interactions between generated policies and current maintenance operations in real environments
  - ▷ framework enhancement by considering distributed vulnerability descriptions and remediation actions (XCCDF [5])

References

- **[1]** M. Barrère, R. Badonnel, and O. Festor. Supporting Vulnerability Awareness in Autonomic Networks and Systems with OVAL. IEEE CNSM'11, October 2011.
- **[2]** The OVAL language. http://oval.mitre.org/. Last visited on April 10, 2012.
- **[3]** Cfengine. http://www.cfengine.org/. Last visited on April 10, 2012.
- **[4]** JAXB, Java Architecture for XML Binding. http://www.oracle.com/. Last visited on April 10, 2012.
- **[5]** XCCDF. http://scap.nist.gov/specifications/xccdf/. Last visited on April 10, 2012.

{barrere,badonnel,festor}@inria.fr