BRITE

BGPSEC / RPKI Interoperability Tester

May 16, 2011
What is BRITE?

**BGPSEC / RPKI Interoperability Test & Evaluation**
- Distributed test and evaluation framework for:
  - RPKI / BGP Security implementation testing,
  - Configuration and deployment testing.
- Flexible XML based test / scenario scripting language.
- Can test all components / interfaces of BGPSEC system.
  - RPKI Validating Caches*
  - Cache to Router Protocol.
  - ROA Processing in BGP Router.
  - BGPSEC Path validation.**

**Distributed / automated test system**
- WWW interface to BRITE – http://brite.antd.nist.gov
- Multi-user infrastructure.
- Real time test monitoring & reporting.
- Other diagnostics:
  - Report, log files, and raw traffic traces are available for download and email.

* Fully available in June/July, ** Availability planned for late 2011 / early 2012
Need for Test & Evaluation

• **RPKI / ROA / BGPSEC**
  – Complicated global systems added to already complicated global BGP routing.
  – Several new protocols & modifications to BGP platforms.
  – Interactions / side effects with existing policy mechanism.

• **Need tool to support / Recommended Audience:**
  – Developers of ROA validation / BGPSEC implementations.
  – Early adaptors to explore new capabilities and components.
  – Customers to evaluate / compare product offerings.
  – Operators to verify test configuration settings.
  – Researchers to study different behavior and stress test system configurations.
BRITE Design Overview

- IUT
- RPKI Validation Cache
- ROA
- RSYNC
- White List Collector / Generator
- Traffic Generator
- Collector
- BRITE Test Controller

WEB Interface

- RSYNC
- PRKI/RTR Protocol
- BGP Protocol

NIST BGPSEC Team
BRITE Test Modeling Language (BML)

- BML is an XML based language to script BRITE tests.
- Tests are scripted using Modular approach
  - Script different modules in different files that can be referenced
  - Customized labeling of module references through aliasing
    - Customizable name space
    - Makes it easier to read / write a test
  - Allows to reuse modules across multiple tests
- Test allow multiple branches within the test flow.
  - Expectations depending on previous events.
- Test scripts are loaded into BRITE using the web interface.
- Report text is scripted within the test for both failure and success.
BML Components

• BML is an XML based language to script BRITE tests.
• Tests are scripted using Modular approach
  – Allows to reuse modules across multiple tests
• System Configuration
  – Describes how to configure the implementation under test (IUT) to be able to establish BRITE ↔ IUT communication.
• Router Configuration
  – Used to configure BGP routers and their roles within BRITE. A router can be used as COLLECTOR, FEED, or IUT.
• Test Data Block models
  – RPKI Tree (available June/July 2011):
    • Provides a complete RPKI tree at a given point in time. A test can contain multiple parallel trees. (example: ARIN tree, RIPE tree, …)
    • Changes within a tree are implemented using multiple static trees where one tree replaces the other during the test run. (example: CRLs added at a given point in time)
  – Whitelist Data:
    • Result of a cache validation using RPKI/Router protocol.
    • Data that can be send out by BRITE → BRITE simulates a validation cache.
    • Data that can be received by BRITE → BRITE tests a validation cache.
  – BGP Data:
    • BGP update templates that are used for both BGP receiving (collector) as well as sending (feed)
• Test Traffic Blocks
  – Whitelist traffic that uses data blocks
  – BGP traffic that uses data blocks.
BML – Configuration Example

- This example script contains the system configuration
  - RSYNC point to provide an RPKI tree.
  - BRITE Cache Client to monitor ROA validation.
  - BRITE Cache Server to provide validation results.
  - BGP Router.

```xml
<brite xmlns="http://www.annd.nist.gov/brite"/>
<configuration id="cfg_1">
  <rsync id="rsy_1" url="rsync://brite.annd.nist.gov/rsync1" reload="30"/>
  <rpki_sc id="rsc_iut" type="IUT_CACHE"/>
  <rpki_sc id="rsc_server" type="B_CACHE"/>
  <rpki_sc id="rsc_client" type="B_CLIENT"/>
  
  <router id="rtr_feed1" as=12"/>
  <router id="rtr_feed2" as=8"/>
  <router id="rtr_iut" type="IUT" as=100"/>
  <router id="rtr_collector" type="COLLECTOR" as="101"/>

<configuration>
</brite>
```
BML – Data Example

- This example script contains raw data
  - Whitelist data.
  - BGP update data.

```xml
<brite xmlns="http://www.antd.nist.gov/brite">
  ...
  <whitelist id="wlt_1">
    <data id="dat_1" time="01:00">add 7 10.10.0.0/16 24</data>
    <data id="dat_4" time="02:00">del 7 10.10.0.0/16 24</data>
  </whitelist>

  <bgp id="bgp_1">
    <data id="dat_1" time="00:00">
      <path>{IUT} {R} 3 5</path>
      <prefix>10.10.0.0/16</prefix>
    </data>
    <data id="dat_2" time="00:30">
      <path>{IUT} {R} 6 1 7</path>
      <prefix>10.10.0.0/16</prefix>
    </data>
  </bgp>
</brite>
```
BML – Traffic Example

- This example script contains traffic
  - Send the whitelist data.
  - Send the BGP traffic.

```xml
<brite xmlns="http://www.antd.nist.gov/brite">
  ...
  <traffic id="tfc_1" type="wl" require="wl" server="@fil_config:cfg_1:rsc_server">
    <action>@fil_data:wlt_1:*</action>
  </traffic>
  
  <traffic id="tfc_2" type="wl" require="bgp" server="@fil_config:cfg_1:rtr_feed1">
    <action>@fil_data:wlt_1:dat_1</action>
  </traffic>
  
  <traffic id="tfc_3" type="wl" require="bgp" server="@fil_config:cfg_1:rtr_feed2">
    <action>@fil_data:wlt_1:dat_2</action>
  </traffic>
</brite>
```
BML – GOALS Example (1)

• This example script contains GOALS / Expectations
  – Whitelist data to be received.

```xml
<goals id="gls_1">
  <goal id="gol_1" label="WL Ann. from IUT Cache" type="wl">
    <require>@fil_data:wlt_1:dat_1</require>
    <report/>
    <failure>Whitelist announcement for ROA [9,10.10.0.0/16] missing!</failure>
  </report>
  <goal id="gol_2" label="WL Withd. from IUT Cache" type="wl">
    <require_gol type="success">gls_1:gol_1</require_gol>
    <require>@fil_data:wlt_1:dat_2</require>
    <report/>
    <failure>Whitelist withdrawal for ROA[7,10.10.0.0/16] missing!</failure>
  </report>
</goal>
```

...
BML – GOALS Example (2)

• This example script contains GOALS / Expectations
  – Whitelist data to be received.

```
...  
<goal id="gol_3" type="bgp">
  <require>@fil_data:bgp_1:dat_2</require>
  <allow>@fil_data:bgp_1:dat_1 before @fil_data:bgp_1:dat_2</allow>
  <report>
    <failure>
      BGP update [{IUT} {R} 6 1 7, 10.10.0.0/16] missing!
      After receiving the ROA Whitelist entry for prefix
      10.0.0.0/16 and origin AS 7, the expected behavior is to
      prefer the longer path originated by AS7.
      1: Check if the whitelist entry was received!
      2: Check if ROA processing is enabled.
    </failure>
  </report>
</goal>
</goals>
```
Simulated Topology

Scenario:
@t1: AS7 Originates 129.6.0.0/16
@t2: AS49 Originates 129.6.0.0/16
@t3: Whitelist {129.6.0.0/16-24, 49, (expiration t5)}

Expectations (Goals):
@t1+: G1: Receive (129.6.0.0/16, AS7)
@t3+: G2: Receive (129.6.0.0/16, AS49)
@t5+: G3: Receive (129.6.0.0/16, AS7)
BRITE Web Interface (1)

Tests available to the User.

Select a test to be started
BRITE Web Interface (2)

Test Progress

Test Timeline

Events:
M = Multiple
A = Activation
B = BGP
W = Whitelist

Experiment Log

Goal Tree
Wait to be activated
Currently processing
Finished successful
NIST BRITE

http://brite.antd.nist.gov