BGPRV: A Library for Fast and Efficient Routing Data Manipulation

DETER/EMIST Workshop
June 15th, 2006 - Arlington, VA
Patrick McDaniel, Sophie Qiu, and Kevin Butler
BGP Security

• Community in the midst of comprehensive effort to evaluate and address the security limitations of BGP, the Internet interdomain routing protocol.

• Where are we?
  ‣ Enormous body of analysis of the behavior of BGP
  ‣ Many, many security protocol proposals
  ‣ Time-tested operational (protective) procedures
  ‣ Community interest, and involvement

Q: Why (technically) don’t we have secure BGP?

A: Massively complex protocol interactions involving thousands of independent (often untrustworthy) entities

A: No way to convincingly evaluate proposals/practices
Trace based experiment ...

Simulation Definition

Extract Topology

Extract Traffic Model

Project Attack

Visualization

Forensic Analysis

Simulation

(simulation scope and timeline)

(simulation model)

(results)

(modified attack)
Problem

• The experimental community has a large and increasing corpus of data to use to drive experiments ... (PREDICT)

• However, it is hard to manage data.
  ‣ A large cost of developing an experiment is in dealing with the source data.
  ‣ This is particularly true of BGP data
    • RouteViews, RIPE, etc., all use MRT or other formats which are collected in obtuse structures on remote servers
    • Tools for managing data specialized/rudimentary
    • No comprehensive API for manipulating
Requirements

• Designers of an experiment need flexible access to the real source traces ...
  ‣ formats should be transparent ...
  ‣ details of collection should be abstracted away ...
  ‣ optimize over many uses ...
  ‣ fast, reliable, etc. ...
  ‣ simplicity

• Goal state: experimenters should be completely ignorant of the location, format, and access methods of corpus of data
BGPRV.pm

• A perl module that abstracts away all of the details of the RouteViews repository
  ‣ Random access stream of entire history
  ‣ Handles both RIB and UPDATE data
    • routing snapshot (RIB), flows (UPDATES)
  ‣ Automatically interrogates RouteViews repository
    • keeps state of repository in persistent store
    • replicates as needed all data
  ‣ Applications programmer interface
    • makes all files, replication, formats transparent to the user
    • converts everything to ASCII

• Note: not only RV, but useful for any structured repository
#!/usr/bin/perl
use BGPRV;

# Create the RV object
my ($mobj, @mrt);
$mobj = BGPRV->new;

# Check for CLP, then init and walk the stream
if ( $ARGV[0] eq "-f" ) { $mobj->online(0); shift; }
$mobj->initMRTstream( $ARGV[0], $ARGV[1] );
while ( $mobj->getNextMRT(@mrt) ) {

    # Process the BGP Announcements
    if ( defined $mrt[$BGPRV::UPD_FIELDS{ANNOUNCES}] ) {
        foreach my $prefix (@{$mrt[$BGPRV::UPD_FIELDS{ANNOUNCES}]}) {
            print "$mrt[$BGPRV::UPD_FIELDS{TIME}]|A|" .
                "$mrt[$BGPRV::UPD_FIELDS{SRCIP}]|" .
                "$BGPRV::ORIGINS[$mrt[$BGPRV::UPD_FIELDS{ORIG}]]|" .
                "$prefix|$mrt[$BGPRV::UPD_FIELDS{PATH}]\n";
        }
    }
}
}
Replication

• Replication: mirroring all remote files locally:

  1. Scan all UPDATE and RIB directories
     • once a day, current month once an hour (with dates and times)
     • write to $RV_REPOSITORY/rv_state.dat
     • get all files, inclusively for all dates within the MRT stream range
     • [All web related tools built upon Perl lwp (libwww-perl) interfaces]

  2. Read each record
     1. Parse MRT formats
        • Convert all MRT data to ASCII
        • Create hash table (w nesting)

  3. Return hash to the caller program
bgpstab: BGP stability

- Observes the stability of address space advertisements over a year
  - Stable prefixes
  - Unstable prefixes
  - Pathological instability
  - Categorization by AS type
  - Characterizations of security apparatus
    - Cryptographic methods, route filters
    - sBGP, IRV, RCP, and many, many others ....


bgptopo: topology extractor

• Extracts the IDR state of the network (or subset)
  ‣ Topology uses
  ‣ Prefix assignments
  ‣ Prefix transitions

• Usage
  ‣ Topology from center-point AS X, diameter Y over period Z
  ‣ bgptopo [-f] <START> <END> <CENTER AS> <DIAMETER> <MAXLINKS>
  ‣ Example:

    ./bgptopo -f "05/24/06 00:00:00" "05/24/06 01:00:00" 5413 3 10

• Note: compiles directly into lseb compatible topo file
bgpevent: event extractor

• How do we extract the important events from a corpus of data -- build the “script” for the evolving environment

• Inprogress: dual of bgptopo, extracts events from corpus
  ‣ Develops maps of state transition of known topology
  ‣ Currently
    • prefix advertisements
    • prefix withdrawals
    • link state changes
    • working: route filters, policy ...

• Research challenges: how do you reliably determine the difference between failures, changes in configuration, and other events from indirect observations

Extract Traffic Model
Future work ...

- Implementation enhancements
  - **Namespaces**: RouteViews, Ripe

- Experimental sources: use non-BGP event information
  - E.g., AT&T Ruby system provides details of interface level logs throughout AS7018
    - Useful for more reliable event introspection
  - E.g., Router configurations, CISCO COI
    - Useful for determining “real” topology, more realistic events

- Further integration with existing experimental systems
  - Iseb (large-scale replayable eBGP experiments)
Availability

• BGPRV and (some) tools are currently available ...
  ‣ Some use in community, more welcome
  ‣ Please see

http://siis.cse.psu.edu/tools.html

or email

siis@cse.psu.edu