

# rBroccoli

Seth Hall

Bro Workshop 2006

# What is this?

- Brings the power of Bro's Broccoli library to the world of a scripting language.
- Fast development and prototyping.
- Developed using SWIG. Don't expect to get much reuse for building bindings in other languages.

# Why Ruby?

- Language features make development of rBroccoli relatively straight forward.
- It's my language of choice.



# Typemapper

- Created out of need to map C types to the correct ruby class.
- Predefine fields in records and arguments in events.
- Future: would be nice to only do the conversion if the data is actually accessed.



# Callbacks (or Event Handlers)

- Anonymous code blocks are commonly used in Ruby and help to give a good syntax for callback creation.
- Methods to code blocks need to be typed ahead of time through the rBroccoli typemapper.

# Example...in C

```
static void
bro_pong(BroConn *conn, void *data, double *src_time, double *dst_time, uint32 *seq)
{
    double now = bro_util_current_time();

    printf("pong event from %s: seq=%i, time=%f/%f s\n",
        host_str, *seq, *dst_time - *src_time,
        now - *src_time);

    conn = NULL;
    data = NULL;
}
```

# Example...in Ruby

```
Bro::Typemap.define_event("pong", [:time, :time, :count])
# Register the pong event handler
bc.event_handler_for "pong" do |src_time, dst_time, seq|
  now = Bro::current_time_f
  puts "pong event: seq=#{seq}, time=#{dst_time-src_time}/#{now-src_time} s"
end
```

# Sending Events

```
(1..10).each do |seq|  
  ev = Bro::Event.new("ping")  
  ev.insert(Bro.current_time_f)  
  ev.insert(seq)  
  bc.send(ev)  
end
```

# Creating and accessing records

```
Bro::Typemap.define_record("ping_data", {:seq => :count,  
                                         :src_time => :time})
```

```
rec = Bro::Record.new("ping_data")  
rec.insert("seq", 5)  
rec.insert("src_time", Bro::current_time_f)
```

```
print rec.seq      # ==> 5|
```

Questions/Requests?