Babe: A system for end to end brokering of diverse data types

Blake Matheny

February 2003

A LaTeX Presentation

What is the purpose of a logging mechanism?

To relay relevant information to an appropriate agent in the amount of time most suitable for that event.

What's the problem?

- Logging mechanisms do not do this, they:
  - Relay data to any agent that understands a certain data format and speaks a certain protocol regardless of content
- What's wrong with this?
  - Heterogeneous application layer data formats lead to information loss or at best information overload
  - Protocol diversity leads to unusable data
  - Not all data is needed by every agent
  - Makes forensics and correlation difficult or impossible

How about an example?

You have several machines running syslog (string) and some of these machines are running tripwire (encrypted TCP stream?) as well.
You have set up an ID system on one machine. This ID system takes XML over TCP.
You are only interested in seeing when a certain user logs in (syslog) on any host or when /etc/passwd changes on host X (tripwire).
When either of those happens you want to be paged, otherwise the data should be converted to XML and sent to the ID system.
Your pager only understands SMS.

How about an example?

You have several machines running syslog (string) and some of these machines are running tripwire (encrypted TCP stream?) as well.
You have set up an ID system on one machine. This ID system takes XML over TCP.
You are only interested in seeing when a certain user logs in (syslog) on any host or when /etc/passwd changes on host X (tripwire).
When either of those happens you want to be paged, otherwise the data should be converted to XML and sent to the ID system.
Your pager only understands SMS.

Does this sound like fun or what?

Some good ideas

- Firewall paradigm
  - Network filters are good, so are application filters (content inspection)
  - **The combination is better**
- Routing paradigm
  - Source routing can be useful
  - How about information routing? (this is where Babe comes in)

Babe architecture

- Modules
  - Input - Collect data from sources
  - Data Format - Convert ASCII data to an intermediary language or from intermediary language to ASCII
  - Output - Send data to requested destinations
- Core
  - When new input data is received from an input module, pass it to a format module or process ourselves with FSM
  - Use filters on intermediary language
  - Filters evaluate as boolean expressions
  - Send to one or more output modules, or discard based on rule-set

Inputs

- Retrieve data for some protocol
  - For example, collect SMTP messages
- Return ASCII text to the core
  - Some protocol information may be retained (HTTP GET, SMTP HELO, etc)
  - Whether or not this happens is largely dependent on what is trying to be accomplished (modules can have config files to specify this)
  - Largely unnecessary for diverse data types, necessary for a proxy
- Currently have named pipe and TCP input modules, a sniffer module is on the way

Outputs

- Output data for some protocol
  - For example, output to a TACACS+ or ID system
- Takes ASCII text from the core
  - If protocol information was retained, use it
  - Otherwise, put the information into the data segment of your protocol and send it on its way
- Currently have a text file output module. Code for input modules is almost completely reusable

Data example

Assume we have an input source that produced data that looks like this:
Feb 19 01:08:04 duke sshd(15483): Accepted password for bmatheny from 127.0.0.1 port 33946 ssh2

If we wanted the date and the user we can easily represent this in some pseudo language as:

%s %2d %2d:%2d:%2d %*s Accepted password for %s %*s
$\lambda_1 = Month, \lambda_2 = Day,$ etc.

What if we actually have some structure though, such as in this example:

<cpl>
<Incoming><**Address type**=v4><**address** is="127.0.0.1"/></**Address**></Incoming>
<Outgoing><**Address type**=v4><**address** is "10.10.10.200"/></**Address**></Outgoing>
</cpl>

Obviously it is ideal to do something like:
$\lambda_1 = src\_address$ when the parent is Incoming and $\lambda_2 = dst\_address$ when the parent is Outgoing

What do we want to do with our data?

Sample Configuration File

```
filter   host_filter  {
  hostname = { "basm.cerias.purdue.edu", "earthsea.cerias.purdue.edu" }; };
filter   user_filter  {
  username = "bmatheny"; };
input esp {
   library  = "esp";
   format_id = string;
   named_pipe = "/tmp/esp.pipe"; };
output textfile {
   library  = " textfile ";
   format_id = string;
   output_filename = "/tmp/esp.output"; };
log chain1 {
   {  host_filter  &&  user_filter  },
   { syslog },
   { network },
   { xml },
   { false } };
```

Requirements

- Need bijection between original data and intermediary language
  - Must maintain structure
  - Transformation must be lossless
- Mappings must be user definable
- Mapping definitions are two-way functions (i.e. $f(x) = y$ and $f(y) = x$)

Data Types

- Data abstraction
- Three types of data formats
  - Unstructured (anything the system has no apriori knowledge of)
  - Semi-structured (html, syslog)
  - Structured (xml, uml, most programming languages)
- Out of n surveyed security data formats, more than half were one of the formats shown in the example (or a slight variation on)
- We believe that we can model the second two types of data formats, while staying within our requirements, with one common language

Where do things stand?

- What's done?
  - We do data proxying (smtp->smtp)
  - We can do network filtering (if source == x send to y)
- What's not done?
  - Transformations
  - Portable regular expression engine for filters
  - Threads for multiple input/output sources
  - Lots more

What's up with the transformations?

- We found a solution!

What's up with the transformations?

- We found a solution!
- We threw that solution in the trash
- Currently investigating non-deterministic fuzzy FSMs and their applications for transformational grammars
  - AT&T has done some interesting work, as has Xerox
  - Looks good, operates as a two way 'function'

What lies ahead?

- Filtering based on temporally isolated events
- Ontology system for NLP and NLG

Research 'opportunities' for budding scientists

- Compiler design
- Mathematical modeling of attacks
- Ontological systems and their role in NLP and NLG
- Data mining
- Digital Libraries

Questions?

http://www.nongnu.org/babe/

This discussion was sponsored by the Center for Sleep Depravation Studies