Using the New Analyzers

In the past few years, each new Bro release has introduced a few new protocol analyzers. These analyzers significantly enhance Bro's capabilities - generating logs for those protocols out of the box and allowing you to write scripts for traffic using those protocols. These logs are often ignored simply because analysts and potential script writers might not be familiar with that particular protocol or how to interpret the newly available data.
The goal of these exercises isn't to comprehensively go through the new analyzers, aiming instead to impart some basic skills for navigating through the documentation, tuning scripts and logs that ship with Bro, and customizing Bro for your environment. While the examples will use the new analyzers, the general skills are adaptable to all current and future analyzers.
Note: make sure you have libgeoip-dev installed in case you work inside your own Bro installation.

A) Working with the Documentation

Accessing the Documentation

Bro documentation is hosted on the Bro site. The documentation for the latest stable version is available at: <http://www.bro.org/sphinx/index.html> and the git master version is at: <http://www.bro.org/sphinx-git/index.html>. The documentation is also included in the Bro source code, in the doc subdirectory.

Navigating the Documentation

The documentation is split up into 3 main parts:

- `Introduction` covers the general Bro philosophy, background information, and getting Bro up and running.
- `Using Bro` focuses on Bro from an incident responder's viewpoint, discussing topics such as how to work with the Bro logs, detecting an example attack, etc. There's also a section on writing some basic Bro scripts, mainly driven by example.
- `Reference` provides in-depth documentation about each of the Bro frameworks, the scripts that ship with Bro, the Bro "built-in functions," and Bro subcomponents such as BroControl. For our purposes, the script reference is especially useful. Unlike the rest of the documentation, this is generated automatically from special comments in the source code itself.

The main points of interest in the Script Reference are:

- Protocol and File analyzer references. These go through the events that the analyzers generate, and any helper functions that they might provide. The event reference is especially useful, since to write a script, we need to determine which event(s) to handle and the parameters for those events.

  Of particular note is that the type of each parameter is a link to further documentation on that type. If the type is a record, this documentation is critical to know which fields to access.

- The Package Index links to in-depth documentation for the protocol and file analyzers (`base/protocols/*` and `base/files/*`), frameworks (`base/frameworks/*`) and policy scripts (`policy/*`). The difference between the protocol and file references and the package documentation is that the former documents the C++ analyzer, while the later documents the accompanying scripts.

- Built-in function (BIF) documentation (accessed through Package Index -> `base/bif`). For example, if your script does any kind of string manipulation, you'll probably need to reference `base/bif/strings.bif.bro`.

Finally, there is a search feature which will search the documentation.

Documentation Exercises

Level beginner
Getting comfortable with accessing the documentation is such a critical component of Bro scripting that it helps to reinforce all the information that was just presented.

Exercise 1

Which version(s) of SNMP does the analyzer support?

Exercise 2

Which DHCP events include the host name option?

Exercise 3

Which field of the RADIUS log should you look at to determine if an authentication attempt succeeded or failed?
Hint: *The data in the log is actually a record. By convention, the name of that record is "ModuleName::Info". Look for the documentation for the type "RADIUS::Info".*

Exercise 4

Which BIF can you use to determine if a given string contains only ASCII characters? What data type does the BIF return?

B) Customizing Analyzer Behavior

Exercise 5

GridFTP includes a setting for the number of bytes transferred before guessing that a connection is a GridFTP data channel. Using the documentation, find this setting and write code to redefine it to 5 MB.
Hint: *GridFTP is part of the FTP package, so look for FTP in the Package Index.*

C) Extending a Log

Level intermediate

Exercise 6

You would like to extend your radius.log to include a new field with the country code that users are attempting to access your VPN from. Fill out the code in `radius_cc.bro` to add this. You can test this with the PCAP `radius_cc.pcap`.

D) Creating a Notice

Bro's Notice Framework is the central mechanism to Bro's alerting, automation and exemption capabilities. Bro has a rather unique philosophy when it comes to "alerting." Policy scripts flag certain network events as *potentially interesting*. When this occurs, a notice is generated, and then it's up to the local configuration to determine what, if anything, should be done with each notice.
By default, notices are logged to the notice.log, but other actions can be attached to notices as well - such as e-mailing the incident response team or using the Exec module to call a script and block an IP address. Duplicate notices can be suppressed, or a notice can be completely silenced.
The notice framework is a complex beast, but is also one of the key configuration and extension points for Bro. For these exercises, we'll only be touching on a few of the most commonly used features.

Exercise 7

Level intermediate
Looking through your new and improved radius.log, you see some odd VPN connections from Romania. Even though you're sure it's totally legit, you decide you want to generate a notice when you see authentication attempts from Romania. Fill out the script `romania_vpn.bro`, then test it with the previous PCAP.

E) Refining a Notice

Level advanced+

Exercise 8

After testing our previous notice, we realize that it's too noisy in our environment. As it turns out, we see a lot of authentication attempts from Romania, but very few successful logins, which is what we really want to focus on. While we could edit our previous script, let's refine this a slightly different way. Fill out the code in `refine_vpn_cc.bro` to restrict it to successful logins only.

```
$ bro -r radius_cc.pcap refine_vpn_cc
$ bro -r radius_cc_fail.pcap refine_vpn_cc
```

F) Determining Which Events to Handle

Level beginner
In most of our exercises so far, we've been handling the `log_radius` event. Many of the analyzer base scripts have such an event. This is often an easy event to handle, since the information provided is the same as what's in the logs.
However, such events might not be the correct event to handle for your script. Apart from going through the documentation to see the possible events, there's a helper script which will print information for all the events. Let's try it.

Exercise 9

```
$ bro -r radius_cc.pcap misc/dump-events
```

Which events are generated? How many times is each event generated?

Exercise 10

A limitation to `misc/dump-events` is that, for performance reasons, Bro will only generate events that are handled in a script. `dump-events` will only display events that are handled in this way. We can work around this by handing the event, of course.

```
# new_packet.bro

# We actually don't need to do anything with the event
    , as long as we're handling it.
event new_packet(c: connection, p: pkt_hdr) { }


$ bro -r radius_cc.pcap new_packet misc/dump-events
```

Conclusion

At this point, you should be able to find and understand the documentation for analyzers, types, and BIFs. The exercises were designed to walk through some basic skills to become a Bro "power user." These skills are enough to really be able to customize Bro to your environment - tuning existing scripts, adding context via additional log fields, creating and tuning notices, and being able to determine which events to handle, so you can go from PCAP to script.
The exercises focused on RADIUS since it's one of the easier protocols to understand, and doesn't require any in-depth knowledge about the protocol. Of course, there were many other analyzers developed recently, which are just as useful. A quick recap of these is:

Application Protocols

- DHCP - While Bro has had a DHCP analyzer for a while, it got a bit of a refresh. DHCP leases will now show up in `dhcp.log`. There's also a new set of policy scripts for creating a `known_devices.log`, in the same vein as `known_hosts.log`. Instead of IPs, MAC addresses are logged, as well as the DHCP host name option, if available. DHCP details.
- DNP3 - A new analyzer for the Distributed Network Protocol, commonly used in Industrial Control Systems (ICS). The base script will generate a `dnp3.log`, with requests and replies. DNP3 details.
- Modbus - Another new analyzer for an ICS protocol. The base script will generate `modbus.log`, with the function being accessed and any exception code that is returned. Modbus has two policy scripts as well, one for tracking changes to registers and one for tracking known Modbus master and slave devices. Modbus details.
- RADIUS - A new analyzer for the RADIUS protocol, commonly used for network authentication. Common use cases are VPN and 802.1x wireless and wired authentications. Will produce a `radius.log`, with the authentication attempts and results. RADIUS details.
- SNMP - One of the more ubiqutous network protocols, SNMP is used for everything from network access auditing, service monitoring, and network device configuration. The new analyzer will create an `snmp.log`, with the session details (including the SNMP community for v1 and v2c), the number of variable bindings that were queried, returned, and modified, and a system description and uptime for the SNMP responder endpoint. SNMP details.

Encapsulation Protocols

- GRE - Generic Routing Encapsulation. Bro can now decapsulate certain types of GRE tunnels.
- GTPv1 - GPRS Tunnelling Protocol version 1. Bro can now decapsulate GTPv1 tunnels as well, which are used in mobile technologies like GSM and LTE.

These tunnels are decapsulated just like other tunnels (Teredo, 6to4, etc.). For more information, see the Tunnels Framework.