Bro Scripts
Agenda Thursday

Block 1: Bro-Overview and introduction.
  • Structure, setup, administration.
  • Exercise: find your way around in the training VM.

Block 2: Bro-logs, network logs.
  • Introduction on logs in Bro.
  • Exercise: use Bro logs to find the attack.

Block 3: Working with Bro scripts.
  • Exercise: access and use included and external scripts.
Block 3 Outline

- Using included scripts
- Working with external scripts
- First glimpse on the Bro scripting language
Objectives for this block

• Being able to find and include Bro scripts
• Get familiar with the different sources of Bro scripts
• Understand the basics of the Bro scripting language
Bro comes with >10,000 lines of script code. Prewritten functionality that’s just loaded.

Scripts generate everything we have seen. Amendable to extensive customization and extension.

Growing community writing 3rd party scripts. Bro could report Mandiant’s APT1 indicators within a day.
How to tell Bro which scripts to load

Where (standard scripts)

<prefix>/share/bro

load scripts within a script

@load <path-to-script>

from the command line

bro <options> <scripts...>

Documentation:
http://www.bro.org/sphinx/scripts/index.html
Script directory walk through

• **base/**
  • Everything loaded by default. Scripts meant to:
    • enable analyzers, collect state, generate protocol logs, provide reusable frameworks and function libraries.
  • base/ is not in the default $BROPATH!

• **policy/**
  • Not loaded by default.
  • Place for scripts that not everyone may want to load.
  • Pick and choose.

• **site/**
  • Location for local configuration.
  • No overwrite during installation.
  • BroControl loads site/local.bro as top-level site script.
module CaptureLoss;

export {
  redef enum Log::ID += { LOG };

  redef enum Notice::Type += {
    ## Report if the detected capture loss exceeds the percentage
    ## threshold.
    Too_Much_Loss
  };

  type Info: record {
    ## Timestamp for when the measurement occurred.
    ts:           time &log;
    ## The time delay between this measurement and the last.
    ts_delta:     interval &log;
    ## In the event that there are multiple Bro instances logging
    ## to the same host, this distinguishes each peer with its
    ## individual name.
    peer:         string &log;
    ## Number of missed ACKs from the previous measurement interval.
    gaps:         count &log;
    ## Total number of ACKs seen in the previous measurement interval.
    acks:         count &log;
    ## Percentage of ACKs seen where the data being ACKed wasn’t seen.
    percent_lost: double &log;
  };

  ## The interval at which capture loss reports are created.
  const watch_interval = 15mins &redef;

  ## The percentage of missed data that is considered “too much”
  ## when the :bro:enum:`CaptureLoss::Too_Much_Loss` notice should be
  ## generated. The value is expressed as a double between 0 and 1 with 1
  ## being 100%.
  const too_much_loss: double = 0.1 &redef;
}

## Report if the detected capture loss exceeds the percentage threshold.
Too_Much_Loss

## The interval at which capture loss reports are created.
const watch_interval = 15mins &redef;

## The percentage of missed data that is considered “too much”
## when the :bro:enum:`CaptureLoss::Too_Much_Loss` notice should be
## generated. The value is expressed as a double between 0 and 1 with 1
## being 100%.
const too_much_loss: double = 0.1 &redef;
module CaptureLoss;

export {
…

## The interval at which capture loss reports are created.
const watch_interval = 15mins &redef;

## The percentage of missed data that is considered "too much"
## when the :bro:enum:`CaptureLoss::Too_Much_Loss` notice should be
## generated. The value is expressed as a double between 0 and 1 with 1
## being 100%.
const too_much_loss: double = 0.1 &redef;

}
module CaptureLoss;

export {
  redef enum Log::ID += { LOG };

  redef enum Notice::Type += {
    ## Report if the detected capture loss exceeds the percentage
    ## threshold.
    Too_Much_Loss
  };
}
module CaptureLoss;

export {
    ...
    type Info: record {
    ## Timestamp for when the measurement occurred.
    ts: time &log;
    ## The time delay between this measurement and the last.
    ts_delta: interval &log;
    ## In the event that there are multiple Bro instances logging
    ## to the same host, this distinguishes each peer with its
    ## individual name.
    peer: string &log;
    ## Number of missed ACKs from the previous measurement interval.
    gaps: count &log;
    ## Total number of ACKs seen in the previous measurement interval.
    acks: count &log;
    ## Percentage of ACKs seen where the data being ACKed wasn't seen.
    percent_lost: double &log;
    }
    ...
    
}
Bro script resources

GitHub

Search

Repositories

LiamRandall/bro-scripts
Bro scripts to be shared with the community
Last updated a year ago

sethhall/bro-scripts
Various analysis scripts for the Bro Intrusion Detection System
Last updated 8 months ago

srunnels/bro-scripts
Last updated 2 years ago

ohshitcompanyname/Bro-Scripts
For Bro
Last updated 4 months ago
Using External Scripts

Demo