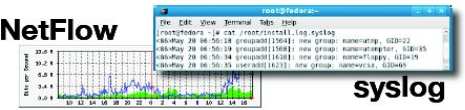
Zeek (Bro) Network Security Monitor

Sareena K P
RISE Lab

What is Bro?



Packet Capture

Traffic Inspection

Attack Detection

Log Recording

Flexibility
Abstraction
Data Structures

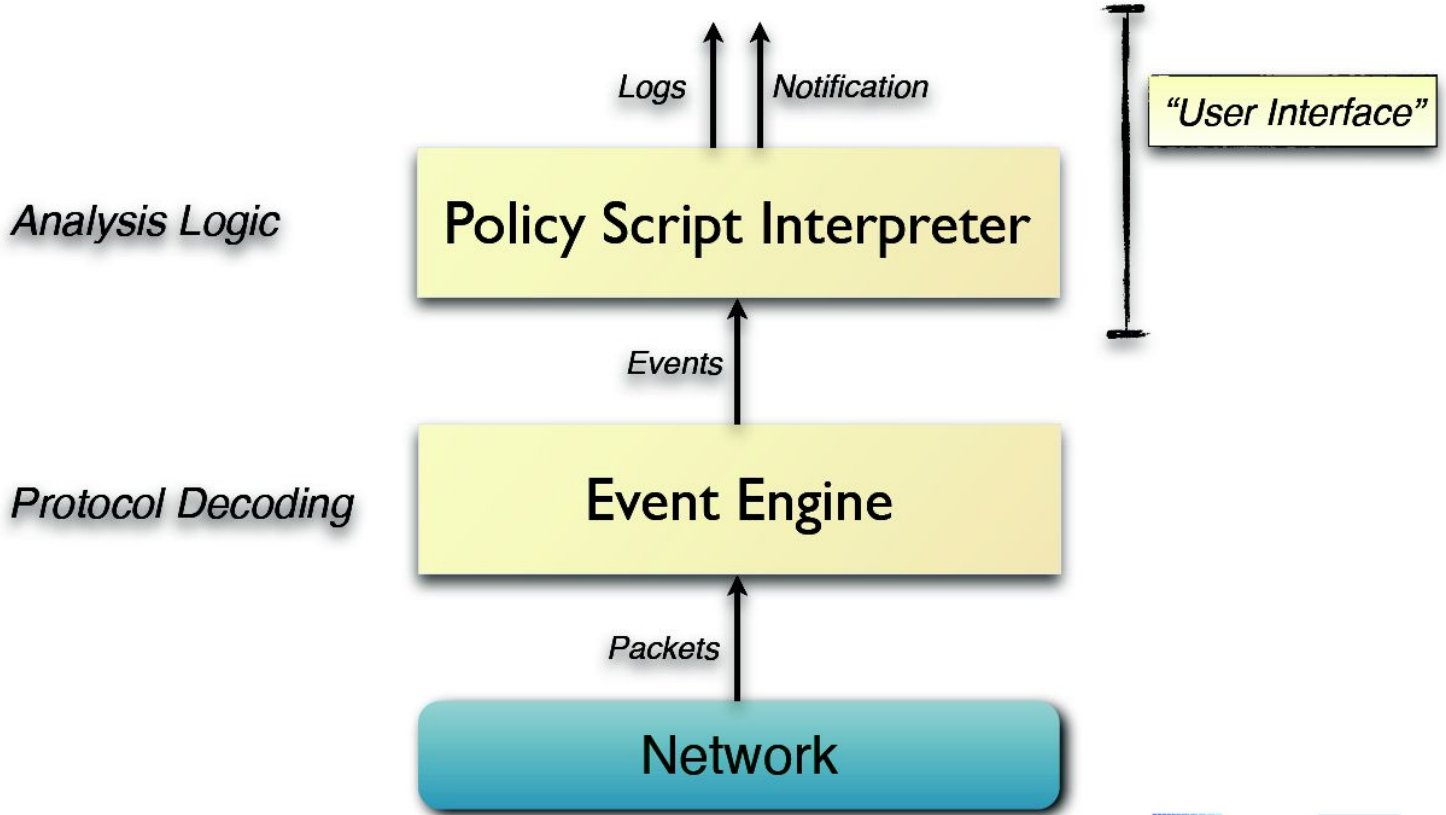
"Domain-specific Python"

Facilitates broader spectrum of very different approaches to find malicious activity

- semantic misuse detection
- anomaly detection
- behavioral analysis.

Source: https://www.zeek.org/documentation/slides/index.html

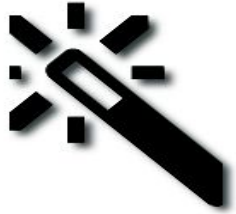
Architecture



What can Bro do?

"Network Ground Truth"

Alerts



Custom Logic

Source: https://www.zeek.org/documentation/slides/index.html

BRO Logs

Built-in functionality for a range of analysis and detection tasks

sudo bro -i wlan0

sudo bro -r sample.pcap

Logs Generated

- Conn.log
- SSH.log
- HTTP.log
- DNS.log
- Files.log
- Software.log

BRO Logs

conn.log

ts	1393099191.817686	Timestamp
uid	Cy3S2U2sbarorQgmw6a	Unique ID
id.orig_h	177.22.211.144	Originator IP
id.orig_p	43618	Originator Port
id.resp_h	115.25.19.26	Responder IP
id.resp_p	25	Responder Port
proto	tcp	IP Protocol
service	smtp	App-layer Protocol
duration	1.414936	Duration
orig_bytes	9068	Bytes by Originator
resp_bytes	4450	Bytes by Responder
conn_state	SF	TCP state
local_orig	T	Local Originator?
missed_bytes	0	Gaps
history	ShAdDaFf	State History
tunnel_parents	(empty)	Outer Tunnels

http.log

ts	1393099291.589208
uid	CKFUW73bIADw0r9pl
id.orig_h	17.22.7.4
id.orig_p	54352
id.resp_h	24.26.13.36
id.resp_p	80
method	POST
host	com-services.pandonetworks.com
uri	/soapservices/services/SessionStart
referrer	-
user_agent	Mozilla/4.0 (Windows; U) Pando/2.6.0.8
status_code	200
username	anonymous
password	-
orig_mime_types	application/xml
resp_mime_types	application/xml

What can Bro do?

"Watch this!"

Recorded in notice.log.
Can trigger actions.

Eg. Suspicious Logins

SSH::Watched_Country_Login

Login from an unexpected country.

SSH::Interesting_Hostname_Login

Login from an unusual host name.

Source: https://www.zeek.org/documentation/slides/index.html

What Can it Do?

Log Files

Custom Logic

"Don't ask what Bro can do.
*Ask what **you want** it to do."*

Zeek - Syntax

- Static type system (i.e., the type of data a variable holds is fixed)
- Regular expression using flex's syntax

```
#pattern matching
print /one|two|three/ == "two";  # T
print /one|two|three/ == "ones"; # F (exact matching)
print /one|two|three/ in "ones"; # T (embedded matching)
print /[123].*/ == "2 two";  # T
```

- Set of domain-specific types : Examples are `time`, `interval`, `port`, `addr`, and `subnet`.

Interactive Learning --- **http://try.bro.org**

Zeek Events

Special flavour of function

- They may be scheduled and executed at a later time, so that their effects may not be realized directly after they are invoked.
- They return no value -- they can't since they're not called directly but rather scheduled for later execution.
- Multiple bodies can be defined for the same event, each one is deemed an "event handler". When it comes time to execute an event, all handler bodies for that event are executed in order of `&priority`.

```
global myevent: event(s: string);
global n = 0;
event myevent(s: string) &priority = -10
{
  ++n;
}
event myevent(s: string) &priority = 10
        {
        print "myevent", s, n;
        }
event bro_init() {
        print "bro_init()";
        event myevent("hi");
        schedule 5 sec { myevent("bye") };
        }
event bro_done() {
        print "bro_done()";}
```

Zeek Hooks

Customization points for modules, as they allow to outsource decisions to site-specific code.

- executes immediately when invoked
- Termination determines if further handlers get executed. If the end of the body, or a `return` statement, is reached, the next hook handler will be executed. If, however, a hook handler body terminates with a `break`statement, no remaining hook handlers will execute.

```
priority 10 myhook handler, hi
break out of myhook handling, hi
```

```
hook myhook(s: string) &priority = 10 {
        print "priority 10 myhook handler", s;
        s = "bye";   }

hook myhook(s: string){
        print "break out of myhook handling", s;
        break;       }

hook myhook(s: string) &priority = -5      {
        print "not going to happen", s;       }

event bro_init()   {
        local ret: bool = hook myhook("hi");
        if ( ret )      {
        print "all handlers ran"; }}
```

Scan Detector

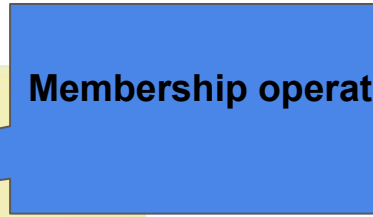
Task: Count failed connection attempts per source address.

```
global attempts: table[addr] of count &default=0;

event connection_rejected(c: connection)
{
    local source = c$id$orig_h;      # Get source address.

    local n = ++attempts[source];    # Increase counter.

    if ( n == SOME_THRESHOLD )       # Check for threshold.
        NOTICE(...);                 # Alarm.
}
```



Excessive DNS Requests

Track the number of DNS Requests - **SumStats**

```
SumStats::observe("dns.lookup", [$host=c$id$orig_h], [$str=query]);
```

```
local r1 = SumStats::Reducer($stream="dns.lookup",apply=set(SumStats::UNIQUE));
```

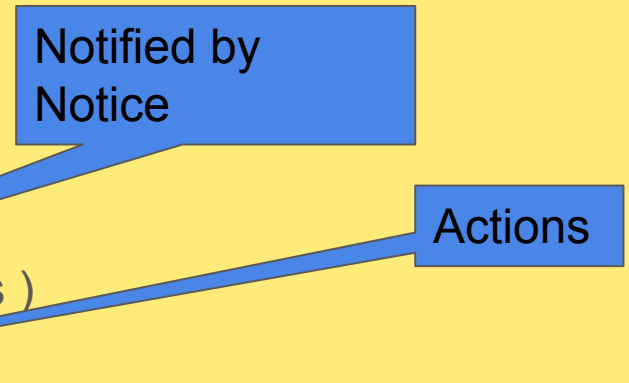
```
SumStats::create([$name="dns.requests.unique", $epoch=6hrs,  $reducers=
                  set(r1), $epoch_result(ts: time, key: SumStats::Key, result:
                  SumStats::Result) = ….]);
```

Filtering Packets

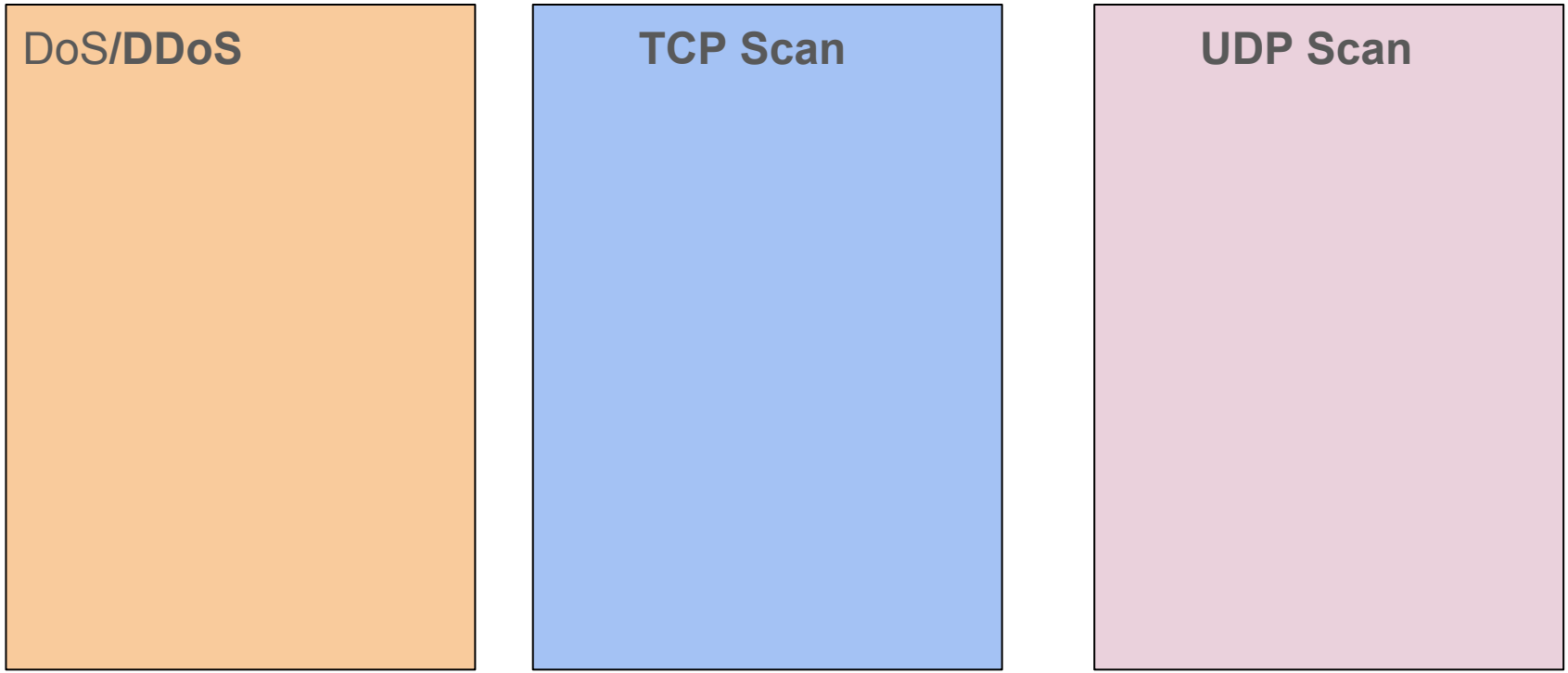
```
event NetControl::init()        {
        local debug_plugin = NetControl::create_debug(T);
        NetControl::activate(debug_plugin, 0);
        }
hook Notice::policy(n: Notice::Info){
        if ( n$note == DNSEXCESS::ExcessiveRequests )
                add n$actions[Notice::ACTION_DROP]; }
```

Filtering Packets

```
event NetControl::init()        {
        local debug_plugin = NetControl::create_debug(T);
        NetControl::activate(debug_plugin, 0);
        }
hook Notice::policy(n: Notice::Info){
        if ( n$note == DNSEXCESS::ExcessiveRequests )
                add n$actions[Notice::ACTION_DROP]; }
```



Stateful filters



Stateful filters

DoS/**DDoS**

Persistent communication from any host to a destination that does not provide replies

High rate of outgoing packets;

TCP Scan

UDP Scan

Stateful filters

DoS/**DDoS**

Persistent communication from any host to a destination that does not provide replies

High rate of outgoing packets;

TCP Scan

Significant number of half-open TCP connections over time

UDP Scan

Stateful filters

DoS/**DDoS**

Persistent communication from any host to a destination that does not provide replies

High rate of outgoing packets;

TCP Scan

Significant number of half-open TCP connections over time

UDP Scan

The ratio of successful versus unsuccessful communication attempts from the network.

Stateful Filters

Email SPAM

The number of email messages from the network;

Malware

Number of failed DNS queries

Installation

- VM will be provided for the tutorial.
- Download

  sudo apt-get install bro

- Installation from source - https://docs.zeek.org/en/stable/install/install.html
  - sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python-dev swig zlib1g-dev
  - ./configure
  - Sudo make
  - Sudo make install
  - export PATH=/usr/local/bro/bin:$PATH

Thank You.