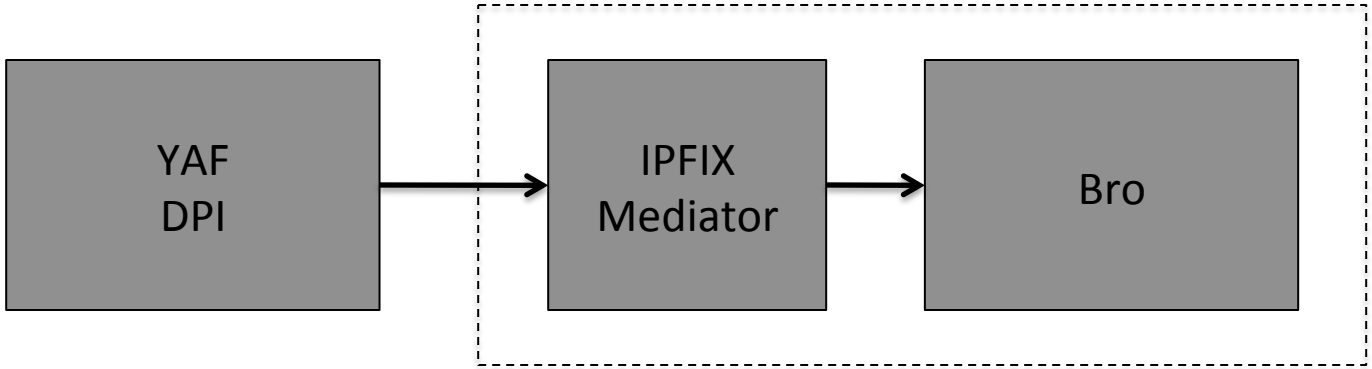
FLOCON 2014

# Network Security Monitoring with IPFIX and Bro

Randy Caldejon

# YAF2Bro Project



# Purpose

**Is it possible to create a framework for producing Actionable Intelligence with YAF and Bro?**

# Actionable Intelligence

"The necessary background information that will enable someone to deal quickly and efficiently with a particular situation."

-- Collins Dictionary

# YAF2BRO (Context)

# Context

1. Bro/SiLK Integration at 2013 Bro Workshop
George Warnagiris

2. Intel Framework Overview at 2013 Bro Workshop
Seth Hall

3. Publication of entitled, "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains"
Eric M. Hutchins, Michael J. Clopperty, Rohan M. Amin, Ph.D.

# Types of Indicators of Cyber Attack

**Atomic**: ip address, email address, http header

**Computed**: regular expressions, hash calculations, packet counters

**Behavioral**: combinatorial logic, activity correlation, complex event processing

# Primary Components for YAF2BRO

YAF
DPI

- Oriented toward performing microscopic analysis
- Detects of patterns in streams of packets
- Regular expression processing
- Focused on producing flow records and L7 extraction
- Produces atomic and computed indicators

---

Bro
Engine

- Oriented toward performing macroscopic analysis
- Detects patterns in stream of events
- Complex event processing
- Focused on policy violations
- Produces (atomic, computed, and) behavioral indicators

# YAF2BRO

## Framework for Flow-based Actionable Intelligence


Atomic
and computed
indicators

Behavioral
indicators

Threat
indicators

# Presentation

- Implementation of mediator
- Integration with Bro
- Example use case

# YAF2BRO (Implementation)

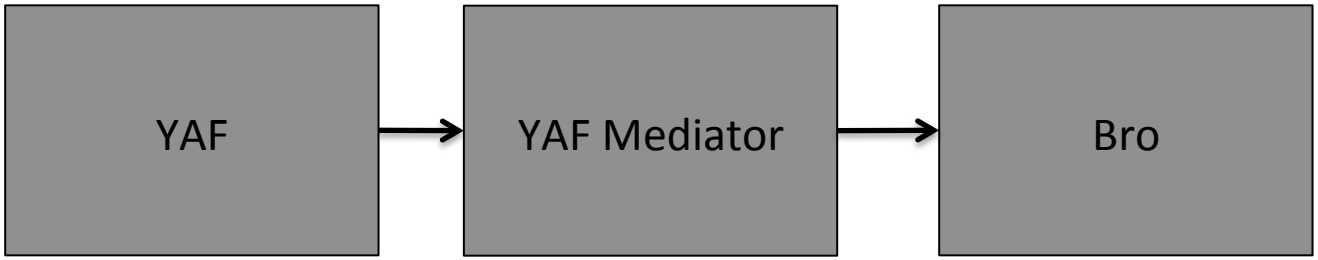
# Building Blocks

Libfixbuf Library (LGPL)

Broccoli Library (BSD)

# Implementation

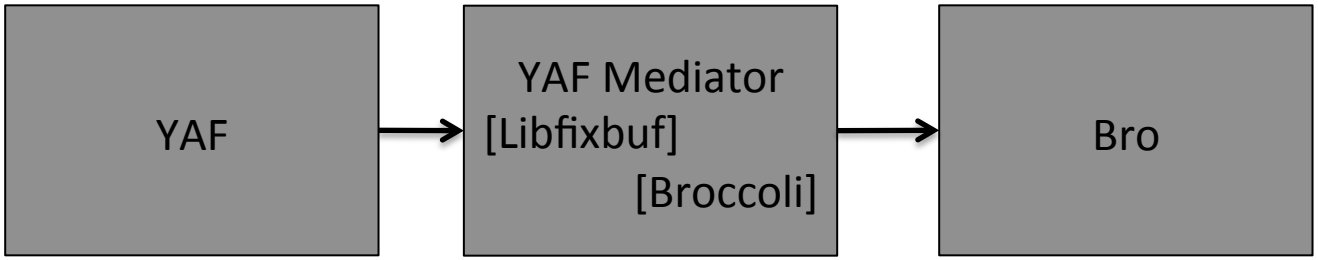
YAF-to-MySQL




- Implemented by Emily Sarneso

# Implementation

YAF-to-Bro




- Basically refactored YAF to MySQL Mediator

# First Module: yaf2bro.c

```
case YAF_DNS_FLOW_TID:
    {
        yfDNSFlow_t *dnsflow = NULL;
        dnsflow = (yfDNSFlow_t *)FBSTMLNEXT(stml, dnsflow);
        yfMyDNSInsert(conn, dnsflow, stml->tmplID, flowID);
    }
    break;
```

---

```
case YAF_DNS_FLOW_TID:
    {
        yfDNSFlow_t *dnsflow = NULL;
        dnsflow = (yfDNSFlow_t *)FBSTMLNEXT(stml, dnsflow);
        yfBroDNSEvent(conn, (yfIpfixFlow_t *) &rec, dnsflow,
                      observationDomain, flowIDString);
    }
    break;
```

# Second Module: yafBroEvents.c

```
1) gboolean yfBroConnectionEvent(
      BroConn *conn,
      yfIpfixFlow_t *ipfixRec,
      uint16_t tcpTmplID,
      yfTcpFlow_t *tcpRec,
      uint16_t observationDomain,
      const char* flowID);
```

# Second Module: yafBroEvents.c

```
2) gboolean yfBroDNSEvent(
      BroConn *conn,
      yfIpfixFlow_t *ipfixRec,
      yfDNSFlow_t *dnsflow,
      uint16_t observationDomain,
      const char* flowID);

3) gboolean yfBroSSLHandShakeEvent(
      BroConn *conn,
      yfIpfixFlow_t *ipfixRec,
      yfSSL2Flow_t *sslflow,
      uint16_t observationDomain,
      const char* flowIDString);
```

# Second Module: yafBroEvents.c

```
4) static gboolean yfBroInsertConnectionRecord (BroRecord *rec,
      yfIpfixFlow_t *ipfixRec,
      const char* flowID);
```

# Firing a Bro event with Broccoli

yfBroConnectionEvent ()

# Broccoli Communications Library

Enables applications to speak the Bro communication protocol.

- ✓ Application or Agent
- ✓ Send or receive events

# Broccoli data types

```
#define    BRO_TYPE_BOOL   1
#define    BRO_TYPE_INT   2
#define    BRO_TYPE_COUNT   3
#define    BRO_TYPE_COUNTER   4
#define    BRO_TYPE_DOUBLE   5
#define    BRO_TYPE_TIME   6
#define    BRO_TYPE_INTERVAL   7
#define    BRO_TYPE_STRING   8
#define    BRO_TYPE_PATTERN   9
#define    BRO_TYPE_ENUM   10
#define    BRO_TYPE_TIMER   11
#define    BRO_TYPE_PORT   12
#define    BRO_TYPE_IPADDR   13
#define    BRO_TYPE_SUBNET   14
#define    BRO_TYPE_ANY   15
#define    BRO_TYPE_TABLE   16
#define    BRO_TYPE_UNION   17
#define    BRO_TYPE_RECORD   18
#define    BRO_TYPE_LIST   19
#define    BRO_TYPE_FUNC   20
#define    BRO_TYPE_FILE   21
#define    BRO_TYPE_VECTOR   22
#define    BRO_TYPE_ERROR   23
#define    BRO_TYPE_PACKET   24
#define    BRO_TYPE_SET   25
```

# Bro IPFIX Record

```
type conn_id: record
{
    orig_h: addr; # originator's IP
    orig_p: port; # originator's port
    resp_h: addr; # responder's IP
    resp_p: port; # responder's port
};
```

```
type ipfix: record
{
    id:       conn_id;# bro connection id
    uid:      string;  # unique string id
    start:    time;    # start of flow time
    end:      time;    # end of of flow time
    pkt:      count;   # forward packet count
    rpkt:     count;   # reverse packet count
    oct:      count;   # forward octet count
    roct:     count;   # reverse octet count
    reason:   count;   # end reason
};
```

# Bro IPFIX Record

```
type ipfix: record
{
    id:       conn_id;# bro connection id
    uid:      string;   # unique string id
    start:    time;     # start of flow time
    end:      time;     # end of of flow time
    pkt:      count;    # forward packet count
    rpkt:     count;    # reverse packet count
    oct:      count;    # forward octet count
    roct:     count;    # reverse octet count
    reason:   count;    # end reason
};
```

```
global ipfix_conn_event: event(conn: ifpfix);
```

# yfBroInsertConnectionRecord.c

```
BroRecord *ipfix = bro_record_new();
if (! ipfix ) {
        printf("Broccoli record allocation error\n");
        return FALSE;
}
BroRecord *conn_id = bro_record_new();
 if (!conn_id) {
        printf("Broccoli record allocation error\n");
        return FALSE;
 }
```

# yfBroInsertConnectionRecord.c

```
BroAddr orig;
if (ipfixRec->sourceIPv4Address)
{
   memcpy(&orig.addr, BRO_IPV4_MAPPED_PREFIX,
                  sizeof(BRO_IPV4_MAPPED_PREFIX));
                  orig.addr[3] = htonl(ipfixRec->sourceIPv4Address);
}
.
.
.
bro_record_add_val(conn_id, "orig_h",  BRO_TYPE_IPADDR, NULL, &orig);
```

# yfBroInsertConnectionRecord.c

```
.
.
.
  BroPort sPort;
  sPort.port_num = ipfixRec->sourceTransportPort;
  sPort.port_proto = ipfixRec->protocolIdentifier;

  bro_record_add_val(conn_id, "orig_p",  BRO_TYPE_PORT, NULL, &sPort);
```

# yfBroInsertConnectionRecord.c

```
.
.
.
bro_record_add_val(ipfix , "conn_id", BRO_TYPE_RECORD, NULL, conn_id);
bro_record_free(conn_id );
```

# yfBroInsertConnectionRecord.c

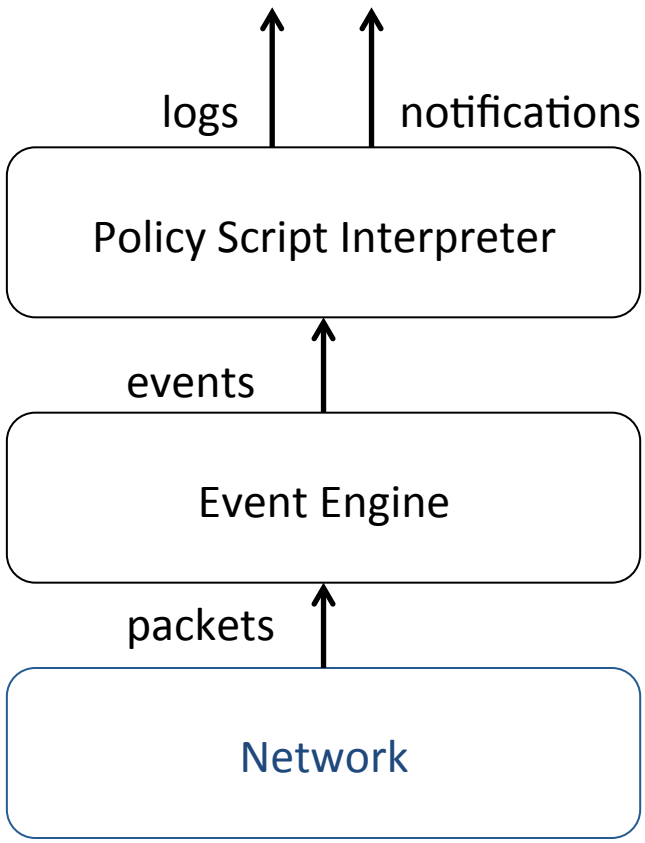
```
bro_record_add_val(ipfix , "pkt",  BRO_TYPE_COUNT, NULL,
                   &ipfixRec->packetTotalCount);
bro_record_add_val(ipfix , "rpkt",  BRO_TYPE_COUNT, NULL,
                   &ipfixRec->reversePacketTotalCount);
bro_record_add_val(ipfix , "oct",  BRO_TYPE_COUNT, NULL,
                   &ipfixRec->octetTotalCount);
bro_record_add_val(ipfix , "roct",  BRO_TYPE_COUNT, NULL,
                   &ipfixRec->reverseOctetTotalCount);
```

# yfBroInsertConnectionRecord.c

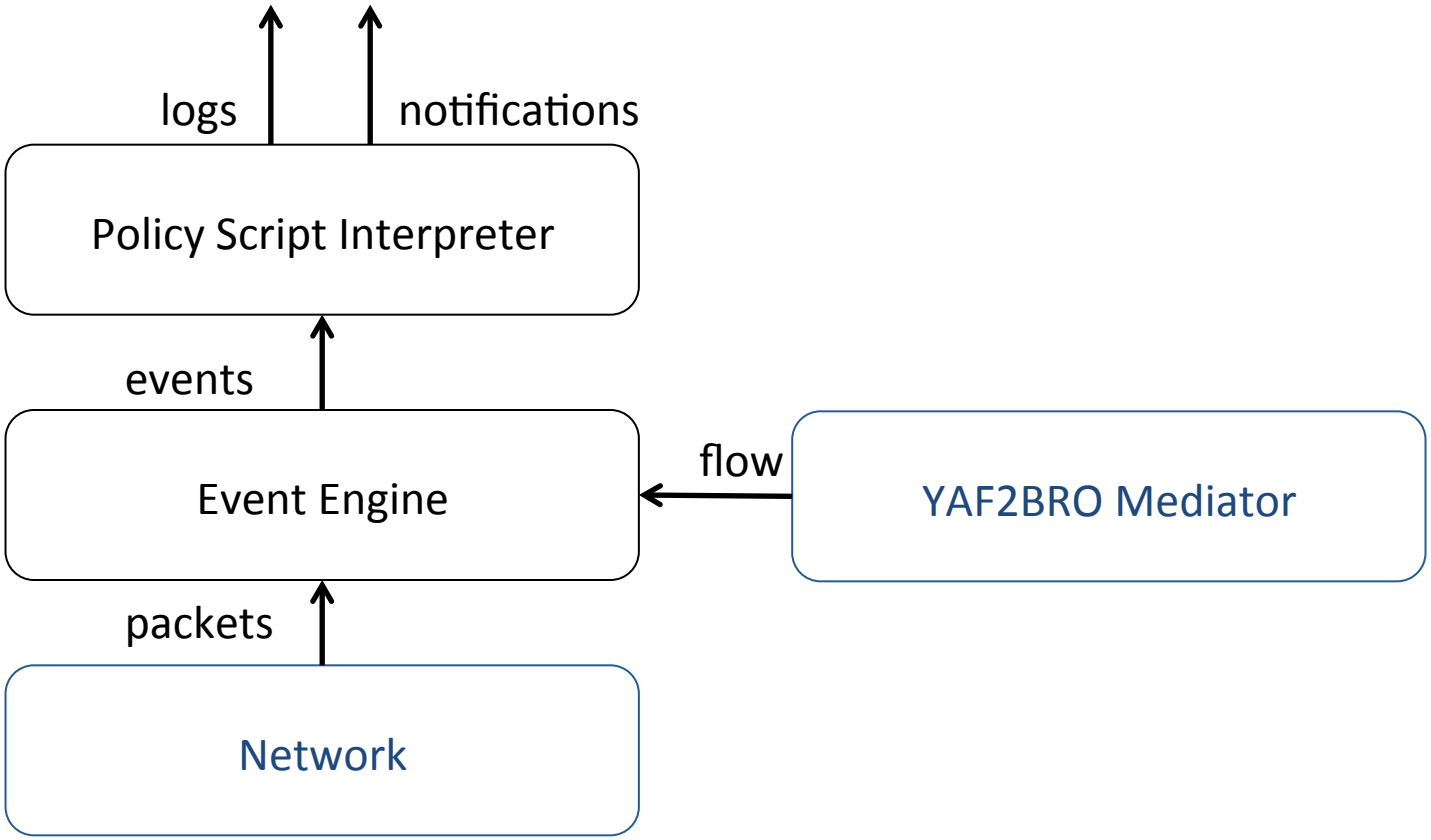
```
BroEvent *ev = bro_event_new("ipfix_conn_event");
if (!ev) {
        printf("Broccoli event allocation error\n");
        return FALSE;
}
.
.
.
bro_event_add_val(ev, BRO_TYPE_RECORD, NULL, rec);
bro_event_send(conn, ev);
bro_event_free(ev);
```

# YAF2BRO (Integration)

# Bro Internal Architecture



# Bro Internal Architecture



# Bro script - yaf.bro

```
global ipfx_log = open_log_file("ipfix_conn");
global ipfx_dns_log = open_log_file("ipfix_dns");
global ipfx_ssl_log = open_log_file("ipfix_ssl");

event ipfix_conn_event(c: ipfix_conn)
{
	print ipfx_log, c;
}
event ipfix_dns_event(dns: ipfix_dns)
{
	print ipfx_dns_log, dns;
}
event ipfix_ssl_event(ssl: ipfix_ssl)
{
	print ipfx_ssl_log, ssl;
}
```

# Bro script - yaf.bro

```
type ipfix_dns: record
{
        conn:     ipfix;
        qname:  string; # Query or Response Name
        qrtype:   count; # Query/Response Type
        qr:          bool;    # Query/Response header field - query (0) or a response (1)
        ttl:         count;  # Time To Live
        aa:         bool;    # Authoritative header field - valid when qr = 1
        rcode:    count;  # NXDomain or Response Code -  3:Name Error, 2:Server
                                #Failure, 1:Format Error, and 0:No Error
        rsection: count; # Resource Record Section Field -  0:Question Section,
                                 # 1:Answer Section, 2:Name Server Section, and 3:Additional
                                 #Section
        tid:           count; # Transaction ID
        data:       string &optional; # data based on qrtype
};
```

# Bro script - yaf.bro

```
type ipfix_x509_certificate: record
{
   serial:        string;
   not_after:        string;
   not_before:      string;
   issuer:        string;
   subject:      string;
};

type ipfix_ssl: record
{
      conn:             ipfix;
      server_cipher:  count;
      client_version: count;
      compression:   count;
      certificate:    ipfix_x509_certificate &optional;
};
```

# ipfix_dns.log

[conn=[id=[orig_h=67.77.165.24, orig_p=47470/udp, resp_h=198.6.1.4, resp_p=53/udp], uid=IPFIX00:5594, start=1389439332.474, end=1389439332.544, pkt=1, rpkt=1, oct=59, roct=189, reason=1], qname=www.isg-apple.com.akadns.net., qrtype=5, qr=T, ttl=48, aa=F, rcode=0, rsection=1, tid=58574, data=www.apple.com.edgekey.net.]

[conn=[id=[orig_h=67.76.165.24, orig_p=47470/udp, resp_h=198.6.1.4, resp_p=53/udp], uid=IPFIX00:5594, start=1389439332.474, end=1389439332.544, pkt=1, rpkt=1, oct=59, roct=189, reason=1], qname=www.apple.com.edgekey.net., qrtype=5, qr=T, ttl=149, aa=F, rcode=0, rsection=1, tid=58574, data=e3191.dscc.akamaiedge.net.]

[conn=[id=[orig_h=67.77.165.24, orig_p=47470/udp, resp_h=198.6.1.4, resp_p=53/udp], uid=IPFIX00:5594, start=1389439332.474, end=1389439332.544, pkt=1, rpkt=1, oct=59, roct=189, reason=1], qname=e3191.dscc.akamaiedge.net., qrtype=1, qr=T, ttl=17, aa=F, rcode=0, rsection=1, tid=58574, data=23.66.205.15]

# ipfix_ssl.log

[conn=[id=[orig_h=67.77.165.24, orig_p=40598/tcp, resp_h=17.151.226.11, resp_p=443/tcp], uid=IPFIX00:5522, start=1389438041.126, end=1389438272.106, pkt=20, rpkt=17, oct=2560, roct=5772, reason=3], server_cipher=4, client_version=3, compression=0, certificate=[serial=4c:20:39:e5:d:31:33:30:37:31:30:30:30:32:38:35, not_after=130710002856Z, not_before=130710002856Z, issuer=cn=Entrust Certification Authority - L1C, ou=(c) 2009 Entrust, Inc., o=Entrust, Inc., c=US, subject=cn=*.icloud.com, o=Apple Inc., l=Cupertino, s=California, c=US]]

[conn=[id=[orig_h=67.77.165.24, orig_p=49117/tcp, resp_h=166.78.79.129, resp_p=993/tcp], uid=IPFIX00:5547, start=1389438924.996, end=1389438932.096, pkt=183, rpkt=176, oct=10119, roct=156987, reason=3], server_cipher=47, client_version=3, compression=0, certificate=[serial=8:67:d5:d:31:32:30:39:32:34:31:32:35:38:35:31, not_after=120924125851Z, not_before=120924125851Z, issuer=cn=RapidSSL CA, o=GeoTrust, Inc., c=US, subject=cn=secure.emailsrvr.com, ou=Domain Control Validated - RapidSSL(R)]]

# ipfix_conn.log

[conn=[id=[orig_h=67.77.165.24, orig_p=48706/tcp, resp_h=17.151.226.17, resp_p=443/tcp], uid=IPFIX00:5639, start=1389440088.772, end=1389440105.234, pkt=18, rpkt=15, oct=2449, roct=5556, reason=3], app=443, rtt=85, isn=756119802, rsn=3702282550, iflags=S, riflags=AS, uflags=APF, ruflags=APF]

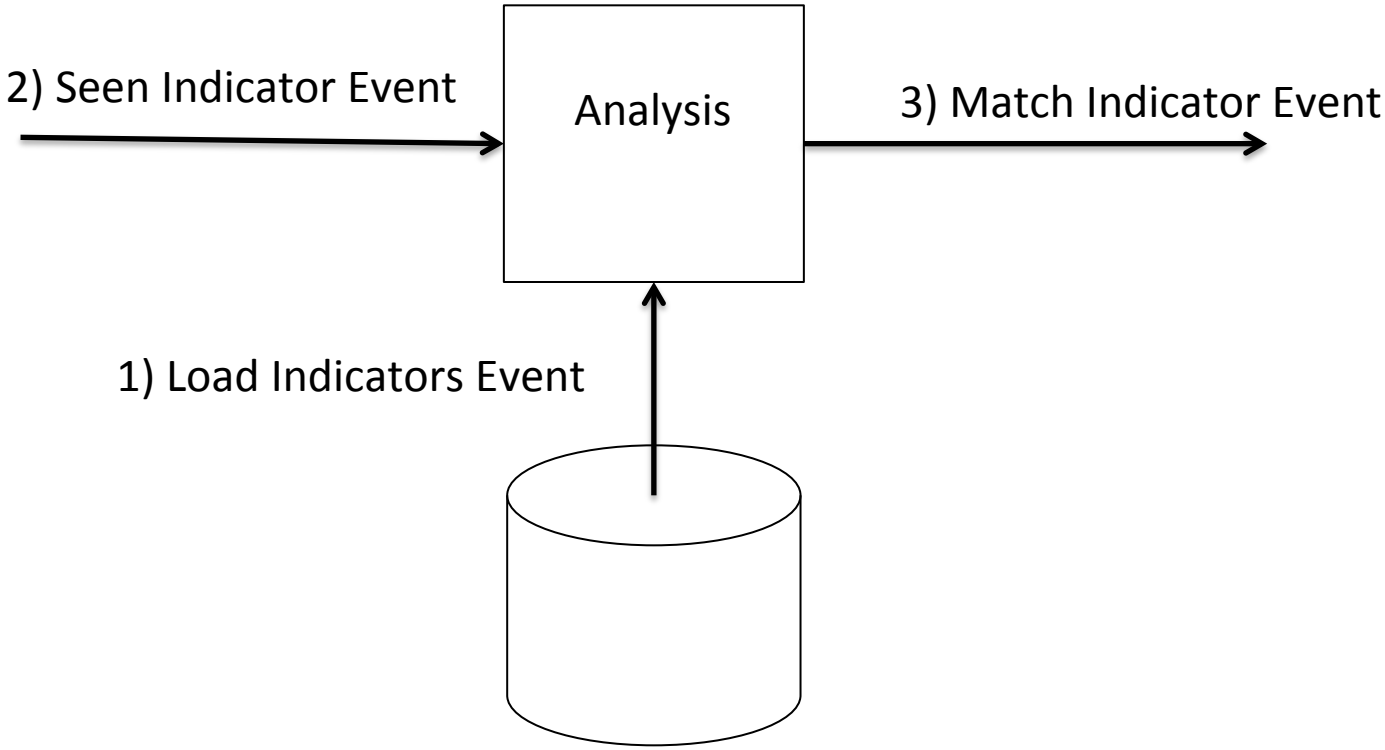
[conn=[id=[orig_h=67.77.165.24, orig_p=37722/tcp, resp_h=17.151.226.15, resp_p=443/tcp], uid=IPFIX00:5640, start=1389440085.73, end=1389440105.234, pkt=18, rpkt=15, oct=3074, roct=5837, reason=3], app=443, rtt=90, isn=1218727344, rsn=4141625483, iflags=S, riflags=AS, uflags=APF, ruflags=APF]

[conn=[id=[orig_h=67.77.165.24, orig_p=21352/udp, resp_h=208.67.222.222, resp_p=53/udp], uid=IPFIX00:5653, start=1389440085.68, end=1389440085.718, pkt=1, rpkt=1, oct=78, roct=94, reason=1], app=53, rtt=19, isn=<uninitialized>, rsn=<uninitialized>, iflags=<uninitialized>, riflags=<uninitialized>, uflags=<uninitialized>, ruflags=<uninitialized>]

# YAF2BRO (Use Case)

# Bro Intelligence Framework

Acting on atomic and computed indicators



# Load Event: Emerging Threats IQRisk

```
@load frameworks/intel/seen
.
.
.
redef Intel::read_files +=
{
	# Emerging Threats IQRisk List
	# Domain name reputation list
	"/opt/yaf/data/domainrepdata.dat",
	# IP reputation list
	"/opt/yaf/data/iprepdata.dat"
};
```

For more information: http://www.emergingthreats.net/intelligence/beyond-ip-reputation/

# Load Event: domainrepdata.dat

```
#fields indicator       indicator_type  meta.source     meta.desc
-0nnu7.blogercontent.com        Intel::DOMAIN   CnC     102
-420-5.suras-ip.com     Intel::DOMAIN   CnC     77
-67i8p0o5i.yourarchivesstoarge.com      Intel::DOMAIN   CnC     27
-88eacdcou.cloudstorepro.com    Intel::DOMAIN   CnC     57
-icon-sushi.sd.softonic.com.br  Intel::DOMAIN   SpywareCnC      77
-o3yo.kolabatory.com    Intel::DOMAIN   CnC     102
-og3le4.firoli-sys.com  Intel::DOMAIN   CnC     42
.
.
.
```

# Load Event: iprepdata.dat

```
#fields indicator       indicator_type  meta.source     meta.desc
1.0.199.10      Intel::ADDR     P2P     50
1.0.236.255     Intel::ADDR     P2P     50
1.0.244.76      Intel::ADDR     Bot     50
1.0.244.76      Intel::ADDR     P2PCnC  110
1.0.254.217     Intel::ADDR     P2P     30
1.1.1.1 Intel::ADDR     Blackhole       120
.
.
.
```

# Seen Event: ipfix_conn_event()

```
event ipfix_conn_event(c: ipfix_conn)
{
    # Report IP address to check it against known intelligence for matches.
    Intel::seen([$indicator="IPFIX", $conn=ipfix2connection(c$conn),
                $host=c$conn$id$orig_h, $indicator_type=Intel::ADDR,
                $where=Conn::IN_ORIG]);
    Intel::seen([$indicator="IPFIX", $conn=ipfix2connection(c$conn),
                $host=c$conn$id$resp_h, $indicator_type=Intel::ADDR,
                $where=Conn::IN_RESP]);

    print ipfx_log, c;
}
```

# Seen Event: ipfix_dns_event()

```
# A, AAAA, CNAME
global QueryRequestReportType: vector of count = vector (1, 28, 5);

event ipfix_dns_event(dns: ipfix_dns)
{
	if (dns$qrtype in QueryRequestReportType)
	{
		if (dns$qr)
			Intel::seen([$indicator=dns$qname,
				$conn=ipfix2connection(dns$conn),
				$indicator_type=Intel::DOMAIN, $where=DNS::IN_REQUEST]);
		else
			Intel::seen([$indicator=dns$qname,
				$conn=ipfix2connection(dns$conn),
				$indicator_type=Intel::DOMAIN, $where=DNS::IN_RESPONSE]);
	}
	print ipfx_dns_log, dns;
}
```

# Seen Event: ipfix_ssl_event()

```
event ipfix_ssl_event(ssl: ipfix_ssl)
{
	if ( /emailAddress=/ in ssl$certificate$subject )
	{
		local email = sub(ssl$certificate$subject, /^.*emailAddress=/, "");
		email = sub(email, /,.*$/, "");
		Intel::seen([$indicator=email,
			$indicator_type=Intel::EMAIL,
			$conn=ipfix2connection(ssl$conn),
			$where=Intel::IN_ANYWHERE]);
	}
	.
	.
	.
```

# Seen Event: ipfix_ssl_event()

```
    .
    .
    .
    # ----------------------------------------
    # report public key
    # ----------------------------------------
    #
    # Intel::seen([$indicator=sha1_hash(der_cert),
    #             $indicator_type=Intel::CERT_HASH,
    #             ipfix2connection(ssl$conn),
    #                   $where=Intel::IN_ANYWHERE]);

    print ipfx_ssl_log, ssl;
}
```

# Match events: intel.log

```
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path intel
#open 2014-01-11-02-47-09
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p fuid file_mime_type file_desc seen.indicator seen.indicator_type
seen.where sources
#types time string addr port addr port string string string string enum enum table[string]
1389457351.876417 IPFIX00:17148 67.77.165.24 44972 108.161.189.192 80 - - - 108.161.189.192 Intel::ADDR
Conn::IN_RESP EXE_Source
1389457498.439943 IPFIX00:17406 114.80.226.94 6000 67.77.165.24 22 - - - 114.80.226.94 Intel::ADDR
Conn::IN_ORIG Brute_Forcer,Scanner
1389457578.225571 IPFIX00:17463 67.77.165.24 47622 66.235.138.224 80 - - - 66.235.138.224 Intel::ADDR
Conn::IN_RESP SpywareCnC
89459443.646463 IPFIX00:18086 59.51.114.74 6000 67.77.165.24 3128 - - - 59.51.114.74 Intel::ADDR Conn::IN_ORIG
Scanner
1389460089.078897 IPFIX00:18253 65.255.46.196 3607 67.77.165.24 3389 - - - 65.255.46.196 Intel::ADDR
Conn::IN_ORIG Brute_Forcer,Scanner
1389461388.573946 IPFIX00:18530 92.63.96.106 49884 67.77.165.24 80 - - - 92.63.96.106 Intel::ADDR Conn::IN_ORIG
Compromised,Brute_Forcer,Scanner
1389461443.664085 IPFIX00:18542 92.63.96.106 52057 67.77.165.24 443 - - - 92.63.96.106 Intel::ADDR
Conn::IN_ORIG Compromised,Brute_Forcer,Scanner
1389461496.436428 IPFIX00:18556 92.63.96.106 54672 67.77.165.24 8080 - - - 92.63.96.106 Intel::ADDR
Conn::IN_ORIG Compromised,Brute_Forcer,Scanner
.
.
.
```

# YAF2BRO (Lessons Learned)

# Final Thoughts

**Is it possible to create a framework for producing Actionable Intelligence with YAF and Bro?**

# Final Thoughts

**Pro**

- ✓ Solution works without having to modify YAF or Bro sources
- ✓ Very complimentary to SiLK
- ✓ Suited for sites with existing deployment of YAF
- ✓ Fairly easy to modify scripts to match site policies and requirements
- ✓ Access to cool frameworks like SumStat, Exec, etc.

**Con**

- ✓ Need support for Tables, Vectors, and Sets in Broccoli
- ✓ YAF SSL DPI should include hash of public key
- ✓ YAF2BRO is still work in progress; still need to support the rest of the L7 extracted protocols fields, i.e. http, p0f

Coming soon…

# Source code?

Randy Caldejon
@packetchaser
https://github.com/packetchaser

# Questions?