# The Bro Network Security Monitor

## Network Forensics with Bro

Matthias Vallentin
UC Berkeley / ICSI
vallentin@icir.org

Bro Workshop 2011
NCSA, Champaign-Urbana, IL

INTERNATIONAL COMPUTER SCIENCE INSTITUTE

NCSA

# Outline

# Post-Facto Forensics

## Scenario

1. You observe symptoms of infections
   - Concrete: some hosts send a lot of spam
   - Abstract: many connections to `[insert malware country here]`
2. Apparently your IDS did not trigger :-(
   - Complex attack: poor/no detection strategy (APT)
   - Evasion
   - 0-day

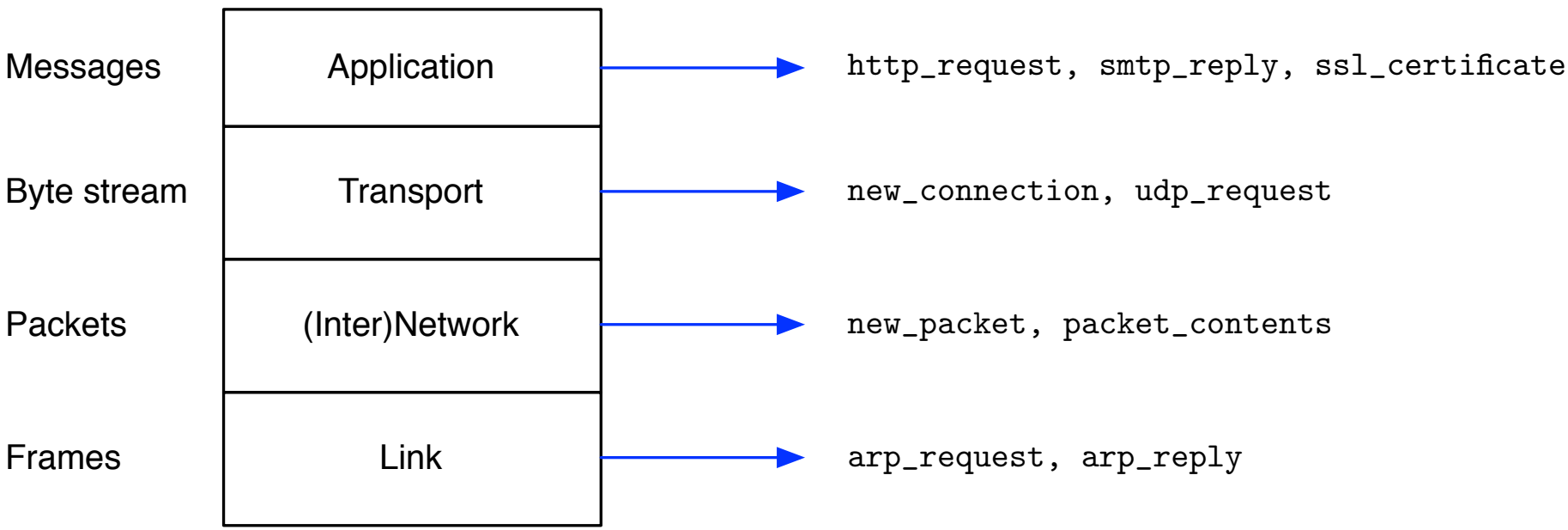
→ Post-facto **log analysis**

What makes Bro logs well-suited for this task?

# Where Do Bro Logs Come From?

## Bro event and data model

- **Rich-typed**: first-class networking types (`addr`, `port`, ...)
- **Deep**: across the whole network stack
- **Fine-grained**: detailed protocol-level information
- **Expressive**: nested data with container types (aka. semi-structured)



# Bro Logs?

# Bro Logs!

Events → Scripts → Logs

- **Policy-neutral** by default: no notion of good or bad
  - Recall the separation of scripts: `base` vs. `policy`
  - Forensic investigations highly benefit from *unbiased* information
  - Hence no use of the term "alert" → **NOTICE** instead
- **Flexible** output formats:
  1. ASCII
  2. Binary (coming soon)
  3. Custom

# Log Analysis

- **What** do we do with Bro's quality logs?
  - Process (ad-hoc analysis)
  - Summarize (time series data, histogram/top-k, quantile)
  - Correlate (machine learning, statistical tests)
  - Age (elevate old data into higher levels of abstraction)
- **How** do we do it?
  - All eggs in one basket
    - SIEM: Splunk, ArcSight, NarusInsight, . . . $$$
    - ELSA (Martin Holste)
    - VAST (under development)
  - In-situ processing
    - Tools of the trade (`bro-cut`, `awk`, `sort`, `uniq`,. . . )
    - MapReduce / Hadoop

# Outline

# Use Case #1: Classic Incident Response

- **Goal**: fast and comprehensive analysis of security incidents
- Often begins with an external piece of **intelligence**
  - "IP X serves malware over HTTP"
  - "This MD5 hash is malware"
  - "Connections to `128.11.5.0/27` at port 42000 are malicious"
- Analysis style: Ad-hoc, interactive, several refinements/adaptions
- Typical operations
  - **Filter**: project, select
  - **Aggregate**: `mean`, `sum`, `quantile`, `min/max`, `histogram`, `top-k`, `unique`

$\Rightarrow$ Concrete starting point, then widen scope (bottom-up)

# Use Case #2: Network Troubleshooting

- **Goal**: find root cause of component failure
- Often no specific hint, merely symptomatic feedback
  - "I can't access my Gmail"
- Typical operations
  - **Zoom**: slice activity at different granularities
    - Time: seconds, minutes, days, ...
    - Space: layer 2/3/4/7, host, subnet, port, URL, ...
  - Study **time series** data of activity aggregates
  - Find abnormal activity
    - "Today we see 20% less outbound DNS compared to yesterday"
    - Infer **dependency graphs**: use joint behavior from past to asses present impact [KMV+09]
    - Judicious machine learning [SP10]

$\Rightarrow$ No concrete starting point, narrow scope (top-down)

# Use Case #3: Combating Insider Abuse

- **Goal**: uncover policy violations of personnel
- Analysis procedure: connect the dots
- Insider attack:
  - Chain of **authorized** actions, hard to detect individually
  - E.g., data exfiltration
    1. User logs in to internal machine
    2. Copies sensitive document to local machine
    3. Sends document to third party via email
- Typical operations
  - Compare activity profiles
    - "Jon never logs in to our backup machine at 3am"
    - "Seth accessed 10x more files on our servers today"

$\Rightarrow$ Relate temporally distant events, behavior-based detection

# Outline

# Example #1: Kaminsky Attack

1. Issue: vulnerable resolvers do not randomize DNS source ports
2. Identify relevant data: DNS, resolver address, UDP source port
3. Jot down your analysis ideas:
   - "For each resolver, no connection should reuse the same source port"
   - "For each resolver, connections should use random source ports"
4. Express analysis:
   - "Count the number of unique source ports per resolver"
5. Use your toolbox:
   - ```
     bro-cut id.resp_p id.orig_h id.orig_p < dns.log \
         | awk '$1 == 53 { print $2, $3 }' \  # Basic DNS only
         | sort | uniq -d  \                  # Duplicate source ports
         | awk '{ print $1 }' | uniq  # Extract unique hosts
     ```
6. Know your limitations:
   - No measure of PRNG quality (Diehard tests, Martin-Löf randomness)
   - Port reuse occurs eventually $\rightarrow$ false positives
7. Close the loop: write a Bro script that does the same
     ```

# Example #1: Kaminsky Attack

## Kaminsky Attack Detector

```
const local_resolvers = { 7.7.7.7, 7.7.7.8 }
global ports: table[addr] of set[port] &create_expire=1hr;

event dns_request(c: connection, ...)
    {
    local resolver = c$id$orig_h;
    if ( resolver !in local_resolvers )
        return;

    local src_port = c$id$orig_p;
    if ( src_port !in ports[resolver] )
        {
        add ports[resolver][src_port]:
        return;
        }

    NOTICE(...);
    }
```

# Example #2: NUL-byte in Certificate

1. Issue: `paypal.com\0.attacker.com` → `paypal.com`
   - Bug manifests only on *client side*, not during certificate registration
2. Identify relevant data: common name (CN) field
3. Jot down analysis ideas:
   - "ASN.1-encoded certificates should not contain non-ASCII characters"
4. Express analysis:
   - "Look for `\0` in CN"
   - "Look for non-ASCII chars in CN"
5. Use your toolbox:
   - ```
     bro-cut subject uid < ssl.log \
         | awk -f cn.awk '{ cn = extract_cn($1); \
                            if (cn ~ /\x00/)
                                print $2 }
     ```
6. Know your limitations
   - Clients may already be patched → user agent, `software.bro`
   - MITM occurs downstream of monitor
7. Close the loop: write a Bro script that does the same
     ```

# Example #2: NUL-byte in Certificate

## Detect NUL-byte in CN

```
event x509_certificate(c: connection, cert: X509, is_server: bool,
                       chain_idx: count, chain_len: count, der_cert: string)
    {
    local cn = "";
    local s = split(cert$subject, /,/); # looks like "k1=v1,k2=v2,..."
    for ( i in s )
        {
        local kv = split(s[i], /=/);
        if ( kv[1] == "CN" )
            {
            cn = kv[2];
            break;
            }
        }

    if ( /\x00/ in cn )
        NOTICE(...);
    }
```

# Example #2: NUL-byte in Certificate

8. Think beyond:
   - "What about other CN weirdness? Mismatching wildcard and SNI?"

## Mismatching `server_name` and wildcarded CN suffix

```
bro-cut uid server_name subject < ssl.log | awk -f cn.awk '{           \
    cn = extract_cn($3);                                              \
    if (cn == "" || $2 == "-")                                        \
        next;                                                         \
                                                                      \
    wildcard = index(cn, "*");                                        \
    if (wildcard > 0)                                                 \
    {                                                                 \
        suffix = substr(cn, wildcard + 2, length(cn) - wildcard - 1); \
        if (index($2, suffix) > 0)                                    \
            next;                                                     \
    }                                                                 \
    else if ($2 == cn)                                                \
        next;                                                         \
                                                                      \
    print $1, $2, cn;                                                 \
}'
```

# Example #3: Duqu Detector

1. Issue: APT
2. Identify relevant data $\triangleq$ network behavior
   - I HTTPS exchange (WinHTTP)
   - II HTTP GET request with `PHPSESSIONID` cookie $\rightarrow$ 54x54 white GIF
   - III HTTP POST upload `default.jpg` $\rightarrow$ 200 OK
   - $\rightarrow$ Also peer-to-peer C&C SMB if external C&C not reachable
3. Jot down analysis ideas:
   - "Follow the behavior defined by the protocol FSM"
4. Toolbox: direct use of Bro
5. Know your limitations
   - APT is highly adaptive $\rightarrow$ hard to describe

# Example #3: Duqu Detector

duqu.bro

```
module HTTP;
export {
    redef enum Notice::Type += {
        Potential_Duqu_Infection
    };

    redef record Info += {
        cookie: string &optional;
        content_type: string &optional;
    };

    type DuquState: enum { ## The Duqu FSM.
        GIF_REQUEST,
        GIF_REPLY,
        JPEG_REQUEST,
        JPEG_REPLY
    };
}
global duqus: table[addr] of DuquState; ## Duqu-infected hosts.
```

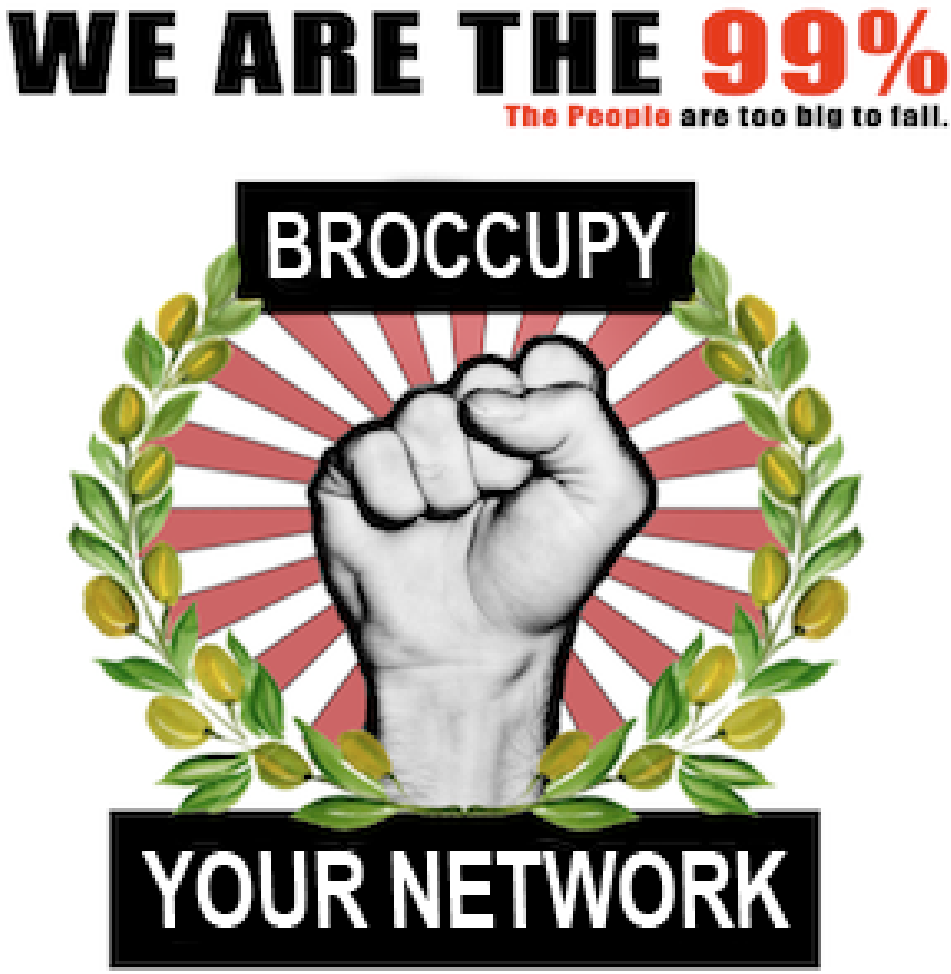
# Example #3: Duqu Detector

duqu.bro

```
event http_request(c: connection, method: string,
    unescaped_URI: string, ...)
    {
    if ( method == "GET" &&
        /^PHPSESSIONID=[[:alnum]]+$/ in c$http$cookie &&
        /([0-9]+){3}\.[0-9]/ in c$http$host && unescaped_URI == "/" )
        duqus[c$id$orig_h] = GIF_REQUEST;
    #...
    }

event http_reply(c: connection, version: string, code: count, ...)
  {
  if ( c$id$orig_h in duqus && duqus[c$id$orig_h] == GIF_REQUEST &&
     version == "HTTP/1.1" && code == 200 &&
     c$http$content_type == "image/gif" )
        {
        duqus[c$id$orig_h] = GIF_REPLY;
        NOTICE([$note=Potential_Duqu_Infection, ...]);
        }
    }
```

# Questions?


WE ARE THE 99%
The People are too big to fail.
BROCCUPY
YOUR NETWORK


Bro

# Next: You Try, We Assist!

## 12-1pm

Lunch (please read the exercise background story)

## 1-2pm: Exercise

Intelligence-Based Incident Response

## 2-2:50pm: Guest Talk

Bro@LBL: Operational Insights (Aashish Sharma & Jim Mellander)

## 3:10-4pm: Exercise

Advanced HTTP Traffic Analysis

## 4:10-4:35pm: Guest Talk

Analyzing and Visualizing Bro Logs with Splunk (Justin Azoff)

# References I


Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitendra Padhye, and Paramvir Bahl.
Detailed Diagnosis in Enterprise Networks.
In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 243–254, New York, NY, USA, 2009. ACM.

Robin Sommer and Vern Paxson.
Outside the Closed World: On Using Machine Learning for Network Intrusion Detection.
In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 305–316, Washington, DC, USA, 2010. IEEE Computer Society.