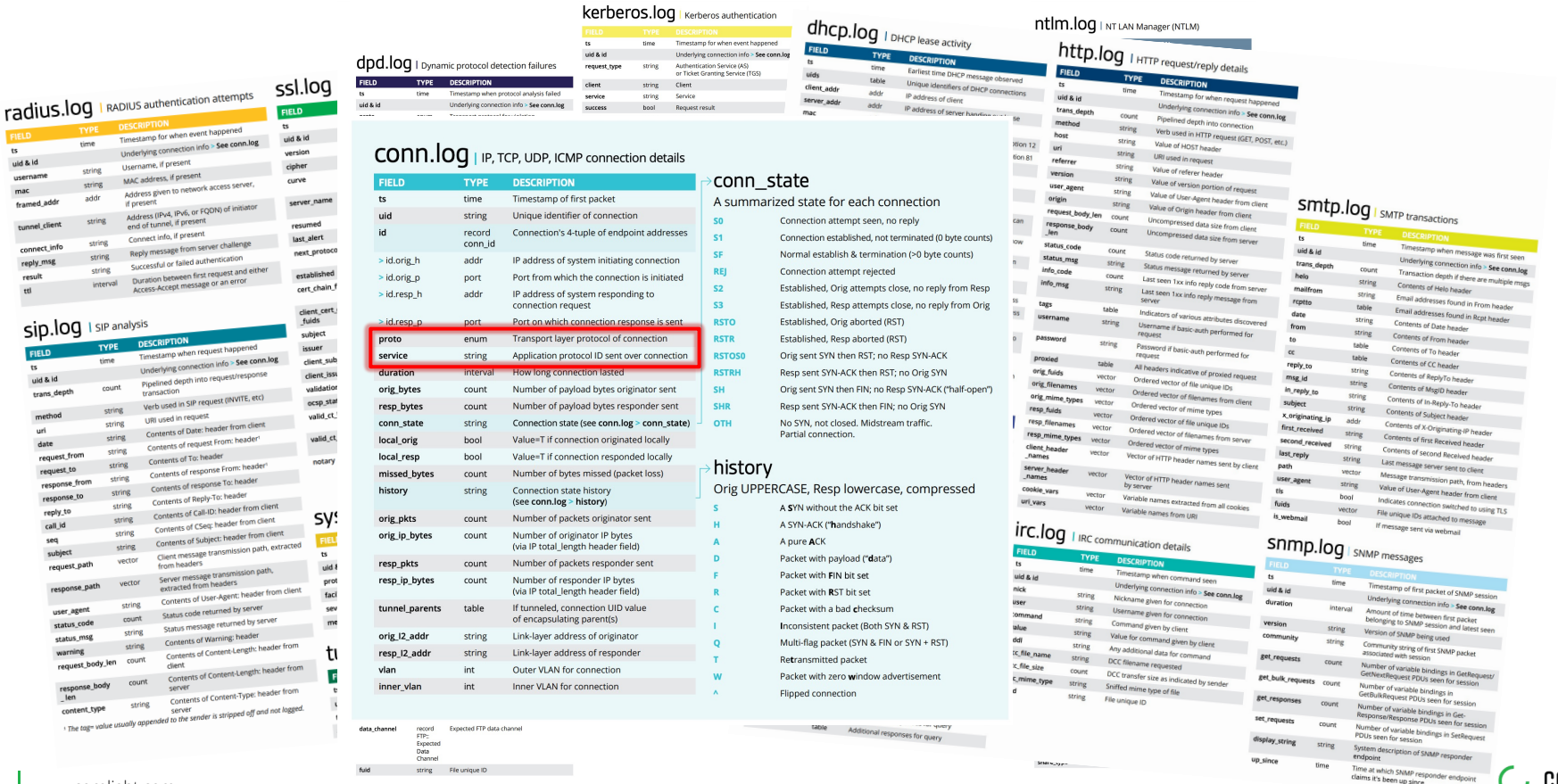
corelight
OPEN SOURCE

Protocol Identification in Zeek

Jan Grashöfer

Zeek Logs

corelight OPEN SOURCE



radius.log | RADIUS authentication attempts

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > See conn.log
username	string	Username, if present
mac	string	MAC address, if present
framed_addr	addr	Address given to network access server, if present
tunnel_client	string	Address (IPv4, IPv6, or FQDN) of initiator end of tunnel, if present
connect_info	string	Connect info, if present
reply_msg	string	Reply message from server challenge
result	string	Successful or failed authentication
ttl	interval	Duration between first request and either Access-Accept message or an error

ssl.log

FIELD
ts
uid & id
version
cipher
curve
server_name
resumed
last_alert
next_protoco
established
cert_chain_f
client_cert_ fuids
subject
issuer
client_sub
client_issu
validatior
ocsp_stat
valid_ct_
valid_ct_
notary

sip.log | SIP analysis

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when request happened
uid & id		Underlying connection info > See conn.log
trans_depth	count	Pipelined depth into request/response transaction
method	string	Verb used in SIP request (INVITE, etc)
uri	string	URI used in request
date	string	Contents of Date: header from client
request_from	string	Contents of request From: header[1]
request_to	string	Contents of To: header
response_from	string	Contents of response From: header[1]
response_to	string	Contents of response To: header
reply_to	string	Contents of Reply-To: header
call_id	string	Contents of Call-ID: header from client
seq	string	Contents of CSeq: header from client
subject	string	Contents of Subject: header from client
request_path	vector	Client message transmission path, extracted from headers
response_path	vector	Server message transmission path, extracted from headers
user_agent	string	Contents of User-Agent: header from client
status_code	count	Status code returned by server
status_msg	string	Status message returned by server
warning	string	Contents of Warning: header
request_body_len	count	Contents of Content-Length: header from client
response_body_len	count	Contents of Content-Length: header from server
content_type	string	Contents of Content-Type: header from server

[1] The tag= value usually appended to the sender is stripped off and not logged.

dpd.log | Dynamic protocol detection failures

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when protocol analysis failed
uid & id		Underlying connection info > See conn.log

kerberos.log | Kerberos authentication

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > See conn.log
request_type	string	Authentication Service (AS) or Ticket Granting Service (TGS)
client	string	Client
service	string	Service
success	bool	Request result

dhcp.log | DHCP lease activity

FIELD	TYPE	DESCRIPTION
ts	time	Earliest time DHCP message observed
uids	table	Unique identifiers of DHCP connections
client_addr	addr	IP address of client
server_addr	addr	IP address of server handing out lease
mac		

ntlm.log | NT LAN Manager (NTLM)

http.log | HTTP request/reply details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when request happened
uid & id		Underlying connection info > See conn.log
trans_depth	count	Pipelined depth into connection
method	string	Verb used in HTTP request (GET, POST, etc.)
host	string	Value of HOST header
uri	string	URI used in request
referrer	string	Value of referer header
version	string	Value of version portion of request
user_agent	string	Value of User-Agent header from client
origin	string	Value of Origin header from client
request_body_len	count	Uncompressed data size from client
response_body_len	count	Uncompressed data size from server
status_code	count	Status code returned by server
status_msg	string	Status message returned by server
info_code	count	Last seen 1xx info reply code from server
info_msg	string	Last seen 1xx info reply message from server
tags	table	Indicators of various attributes discovered
username	string	Username if basic-auth performed for request
password	string	Password if basic-auth performed for request
proxied	table	All headers indicative of proxied request
orig_fuids	vector	Ordered vector of file unique IDs
orig_filenames	vector	Ordered vector of filenames from client
orig_mime_types	vector	Ordered vector of mime types
resp_fuids	vector	Ordered vector of file unique IDs
resp_filenames	vector	Ordered vector of filenames from server
resp_mime_types	vector	Ordered vector of mime types
client_header_names	vector	Vector of HTTP header names sent by client
server_header_names	vector	Vector of HTTP header names sent by server
cookie_vars	vector	Variable names extracted from all cookies
uri_vars	vector	Variable names from URI

smtp.log | SMTP transactions

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when message was first seen
uid & id		Underlying connection info > See conn.log
trans_depth	count	Transaction depth if there are multiple msgs
helo	string	Contents of Helo header
mailfrom	string	Email addresses found in From header
rcptto	table	Email addresses found in Rcpt header
date	string	Contents of Date header
from	string	Contents of From header
to	table	Contents of To header
cc	table	Contents of CC header
reply_to	string	Contents of ReplyTo header
msg_id	string	Contents of MsgID header
in_reply_to	string	Contents of In-Reply-To header
subject	string	Contents of Subject header
x_originating_ip	addr	Contents of X-Originating-IP header
first_received	string	Contents of first Received header
second_received	string	Contents of second Received header
last_reply	string	Last message server sent to client
path	vector	Message transmission path, from headers
user_agent	string	Value of User-Agent header from client
tls	bool	Indicates connection switched to using TLS
fuids	vector	File unique IDs attached to message
is_webmail	bool	If message sent via webmail

irc.log | IRC communication details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when command seen
uid & id		Underlying connection info > See conn.log
nick	string	Nickname given for connection
user	string	Username given for connection
command	string	Command given by client
value	string	Value for command given by client
addl	string	Any additional data for command
dcc_file_name	string	DCC filename requested
dcc_file_size	count	DCC transfer size as indicated by sender
dcc_mime_type	string	Sniffed mime type of file
fuid	string	File unique ID

snmp.log | SNMP messages

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp of first packet of SNMP session
uid & id		Underlying connection info > See conn.log
duration	interval	Amount of time between first packet belonging to SNMP session and latest seen
version	string	Version of SNMP being used
community	string	Community string of first SNMP packet associated with session
get_requests	count	Number of variable bindings in GetRequest/ GetNextRequest PDUs seen for session
get_bulk_requests	count	Number of variable bindings in GetBulkRequest PDUs seen for session
get_responses	count	Number of variable bindings in Get-Response/Response PDUs seen for session
set_requests	count	Number of variable bindings in SetRequest PDUs seen for session
display_string	string	System description of SNMP responder endpoint
up_since	time	Time at which SNMP responder endpoint claims it's been up since

conn.log | IP, TCP, UDP, ICMP connection details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp of first packet
uid	string	Unique identifier of connection
id	record conn_id	Connection's 4-tuple of endpoint addresses
> id.orig_h	addr	IP address of system initiating connection
> id.orig_p	port	Port from which the connection is initiated
> id.resp_h	addr	IP address of system responding to connection request
> id.resp_p	port	Port on which connection response is sent
proto	enum	Transport layer protocol of connection
service	string	Application protocol ID sent over connection
duration	interval	How long connection lasted
orig_bytes	count	Number of payload bytes originator sent
resp_bytes	count	Number of payload bytes responder sent
conn_state	string	Connection state (see conn.log > conn_state)
local_orig	bool	Value=T if connection originated locally
local_resp	bool	Value=T if connection responded locally
missed_bytes	count	Number of bytes missed (packet loss)
history	string	Connection state history (see conn.log > history)
orig_pkts	count	Number of packets originator sent
orig_ip_bytes	count	Number of originator IP bytes (via IP total_length header field)
resp_pkts	count	Number of packets responder sent
resp_ip_bytes	count	Number of responder IP bytes (via IP total_length header field)
tunnel_parents	table	If tunneled, connection UID value of encapsulating parent(s)
orig_l2_addr	string	Link-layer address of originator
resp_l2_addr	string	Link-layer address of responder
vlan	int	Outer VLAN for connection
inner_vlan	int	Inner VLAN for connection

conn_state

A summarized state for each connection

S0	Connection attempt seen, no reply
S1	Connection established, not terminated (0 byte counts)
SF	Normal establish & termination (>0 byte counts)
REJ	Connection attempt rejected
S2	Established, Orig attempts close, no reply from Resp
S3	Established, Resp attempts close, no reply from Orig
RSTO	Established, Orig aborted (RST)
RSTR	Established, Resp aborted (RST)
RSTOS0	Orig sent SYN then RST; no Resp SYN-ACK
RSTRH	Resp sent SYN-ACK then RST; no Orig SYN
SH	Orig sent SYN then FIN; no Resp SYN-ACK ("half-open")
SHR	Resp sent SYN-ACK then FIN; no Orig SYN
OTH	No SYN, not closed. Midstream traffic. Partial connection.

history

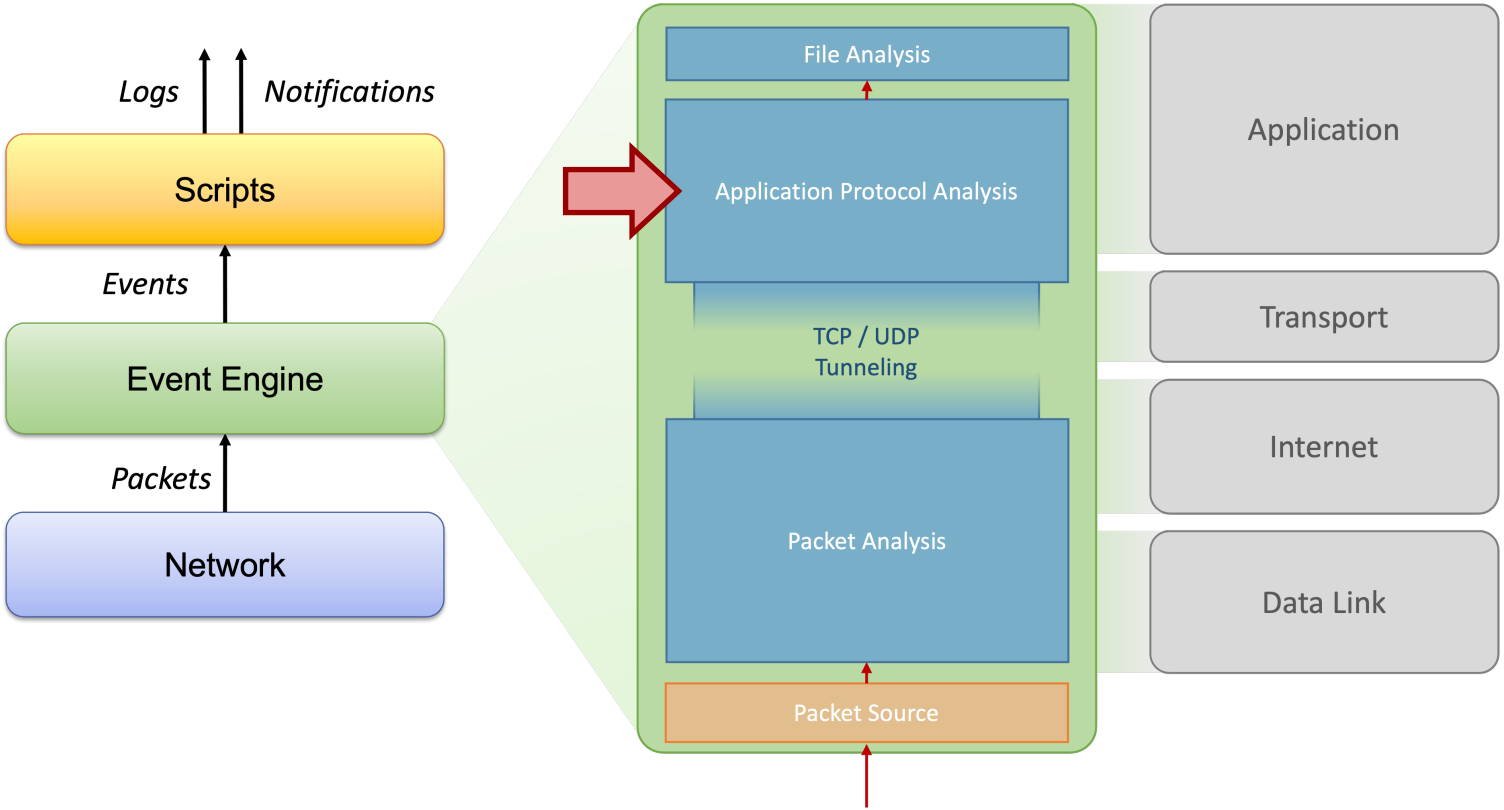
Orig UPPERCASE, Resp lowercase, compressed

S	A **S**YN without the ACK bit set
H	A SYN-ACK ("**h**andshake")
A	A pure **A**CK
D	Packet with payload ("**d**ata")
F	Packet with **F**IN bit set
R	Packet with **R**ST bit set
C	Packet with a bad **c**hecksum
I	**I**nconsistent packet (Both SYN & RST)
Q	Multi-flag packet (SYN & FIN or SYN + RST)
T	Re**t**ransmitted packet
W	Packet with zero **w**indow advertisement
^	Flipped connection

data_channel	record	Expected FTP data channel
	FTP::Expected Data Channel	
fuid	string	File unique ID

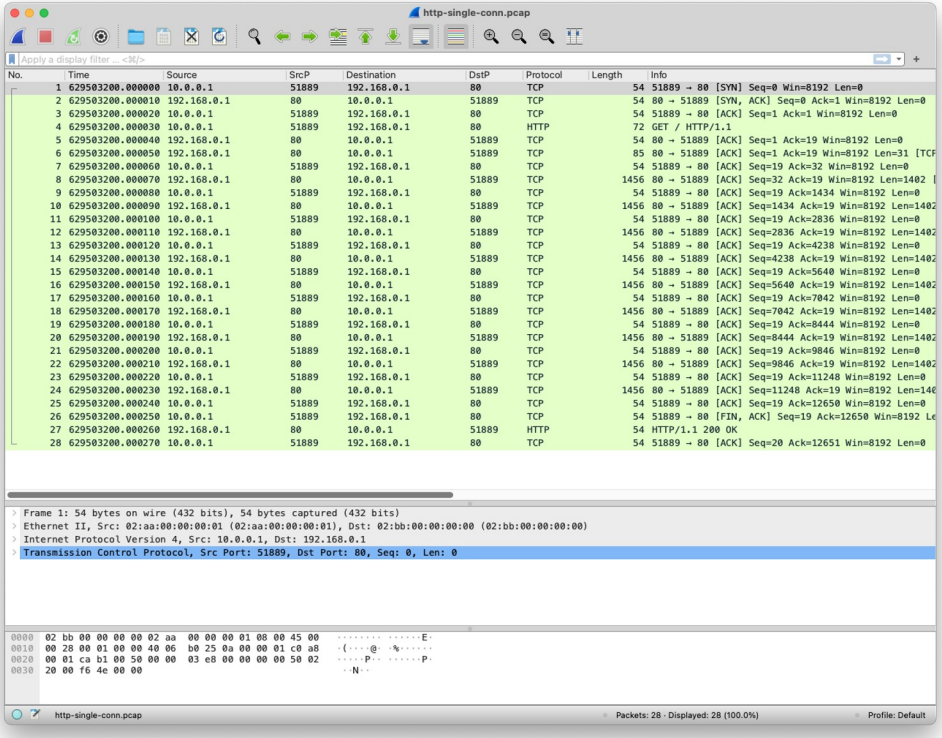
corelight OPEN SOURCE

Zeek Architecture – Overview



Protocol in conn.log

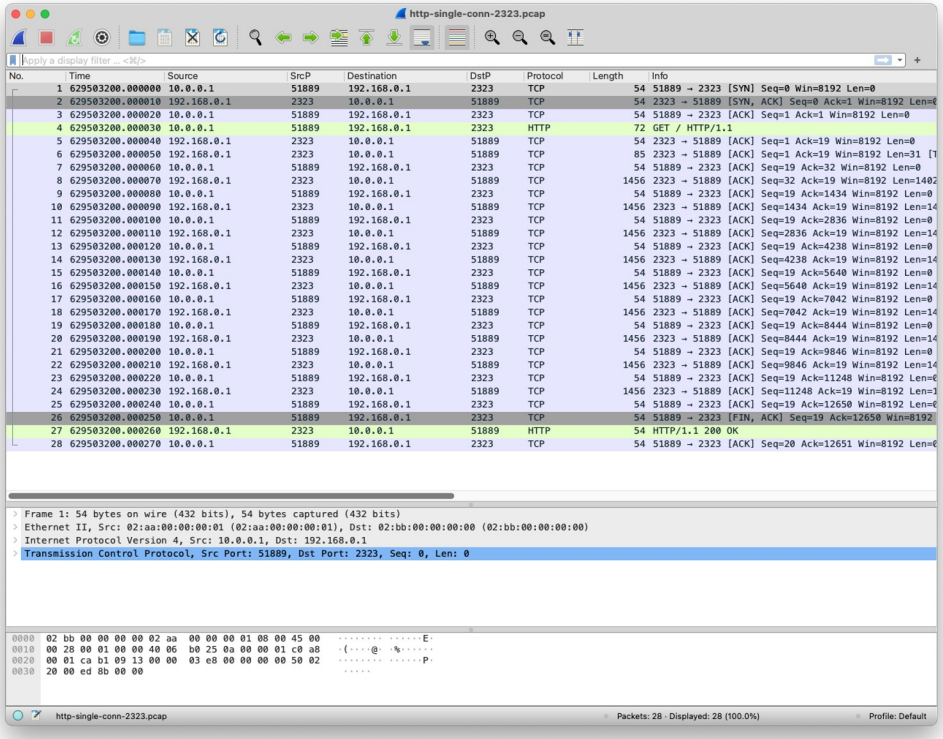

```
{
  "ts": 629503200,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "10.0.0.1",
  "id.orig_p": 51889,
  "id.resp_h": "192.168.0.1",
  "id.resp_p": 80,
  "proto": "tcp",
  "service": "http",
  "duration": 0.00025999954605102539,
  "orig_bytes": 18,
  "resp_bytes": 12649,
  "conn_state": "SF",
  "local_orig": true,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADadFf",
  "orig_pkts": 15,
  "orig_ip_bytes": 618,
  "resp_pkts": 13,
  "resp_ip_bytes": 13169
}
```

corelight
OPEN SOURCE

Protocol in conn.log – What changed?


```
{
  "ts": 629503200,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "10.0.0.1",
  "id.orig_p": 51889,
  "id.resp_h": "192.168.0.1",
  "id.resp_p": 2323,
  "proto": "tcp",
  "service": "http",
  "duration": 0.00025999954605102539,
  "orig_bytes": 18,
  "resp_bytes": 12649,
  "conn_state": "SF",
  "local_orig": true,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADadFf",
  "orig_pkts": 15,
  "orig_ip_bytes": 618,
  "resp_pkts": 13,
  "resp_ip_bytes": 13169
}
```

corelight
OPEN SOURCE

Application Protocol Identification

Application Protocol Identification


```
{
  "ts": 629503200,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "10.0.0.1",
  "id.orig_p": 51889,
  "id.resp_h": "192.168.0.1",
  "id.resp_p": 2323,
  "proto": "tcp",
  "service": "http",
  "duration": 0.0002599954605102539,
  "orig_bytes": 18,
  "resp_bytes": 12649,
  "conn_state": "SF",
  "local_orig": true,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADadFf",
  "orig_pkts": 15,
  "orig_ip_bytes": 618,
  "resp_pkts": 13,
  "resp_ip_bytes": 13169,
  "ip_proto": 6
}
```


- Dynamic Protocol Detection (DPD)
  → Trial and Error approach
  - Just run every analyzer?
    🥵 too resource intensive!

- Attach analyzers based on heuristics
  - **Port**-based
    Well-known/registered ports (e.g., 80 for HTTP)
  - **Content**-based
    Look for expected patterns (e.g., HTTP Methods)

📗 Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection

DPD: Signature Matching

- Challenge: Signature sensitivity
  - Pattern accuracy → specific vs. lax
  - Require matches in originator and responder direction? (`requires-reverse-signature`)

```
signature dpd_http_client {
  ip-proto == tcp
  payload /^[[:space:]]*(OPTIONS|GET|HEAD|POST|PUT
  tcp-state originator
  enable "http"
}

signature dpd_http_server {
  ip-proto == tcp
  payload /^HTTP\/[0-9]/
  tcp-state responder
  enable "http"
}
```

scripts/base/protocols/http/dpd.sig

```
GET /index.html
```

```
HTTP/1.0 200 OK
Hello World!
```

```
CUSTOMMETHOD /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
Hello customized World!
```

DPD: Signature Matching

- Challenge: Signature sensitivity
  - Pattern accuracy → lax vs. specific
  - Require matches in originator and responder direction? (`requires-reverse-signature`)
- Content buffered to replay on signature match
  - `dpd_buffer_size = 1024`
  - `dpd_max_packets = 100`
  - DPD stops if buffer is exhausted

```
GET /index.html
```

```
HTTP/1.0 200 OK
Hello World!
```

```
CUSTOMMETHOD /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
Hello customized World!
```

```
POST /index.html HTTP/1.1
Content-Type: application/octet-stream
Content-Length: 64000

Data...
```

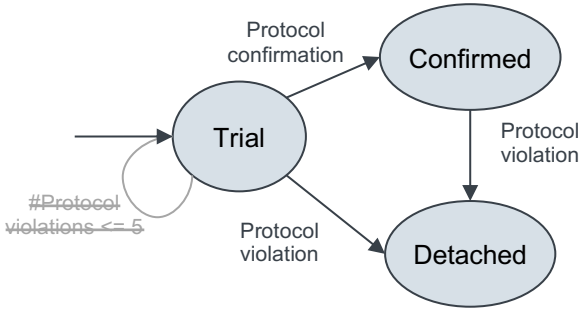
```
HTTP/1.1 200 OK
Finally, hello World!
```

DPD: Trial and Error


```
{
  "ts": 629503200,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "10.0.0.1",
  "id.orig_p": 51889,
  "id.resp_h": "192.168.0.1",
  "id.resp_p": 2323,
  "proto": "tcp",
  "service": "http",
  "duration": 0.0002599954605102539,
  "orig_bytes": 18,
  "resp_bytes": 12649,
  "conn_state": "SF",
  "local_orig": true,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADadFf",
  "orig_pkts": 15,
  "orig_ip_bytes": 618,
  "resp_pkts": 13,
  "resp_ip_bytes": 13169,
  "ip_proto": 6
}
```


- Dynamic Protocol Detection (DPD)
  → Trial and Error approach
  - Run all potential analyzers in parallel
  - Analyzers run in trial mode until
    - **Confirmation** → `service` field set in `conn.log`
    - **Violation** → Analyzer is detached ~~if threshold crossed~~
  - Confirmed analyzers are detached upon violation
    - ~~Protocol removed from `service` in `conn.log`~~
    - Protocol is kept in `service` in `conn.log`

Changed in Zeek 7.2



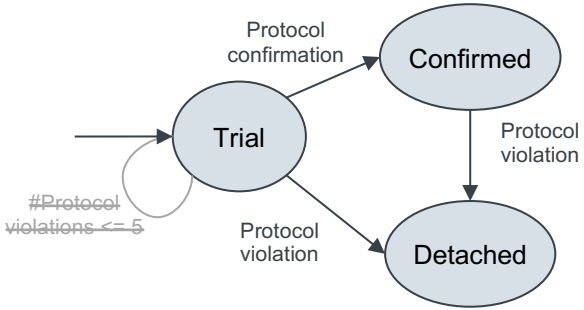
corelight
OPEN SOURCE

DPD: Trial and Error


```
{
  "ts": 629503200,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "10.0.0.1",
  "id.orig_p": 51889,
  "id.resp_h": "192.168.0.1",
  "id.resp_p": 2323,
  "proto": "tcp",
  "service": "http",
  "duration": 0.0002599954605102539,
  "orig_bytes": 18,
  "resp_bytes": 12649,
  "conn_state": "SF",
  "local_orig": true,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADadFf",
  "orig_pkts": 15,
  "orig_ip_bytes": 618,
  "resp_pkts": 13,
  "resp_ip_bytes": 13169,
  "ip_proto": 6
}
```


- ⚠ Protocol-specific log generation independent of confirmation
- Configurable cut-offs
  - ~~DPD::max_violations: table[Analyzer::Tag] of count = table() &default = 5~~
  - `DPD::ignore_violations: set[Analyzer::Tag] = set()`
  - `DPD::ignore_violations_after = 10 * 1024`

Changed in Zeek 7.2

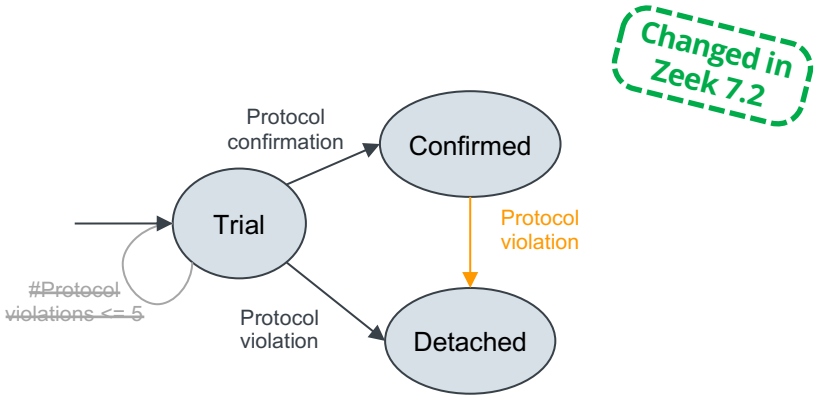


corelight
OPEN SOURCE

DPD: Trial and Error

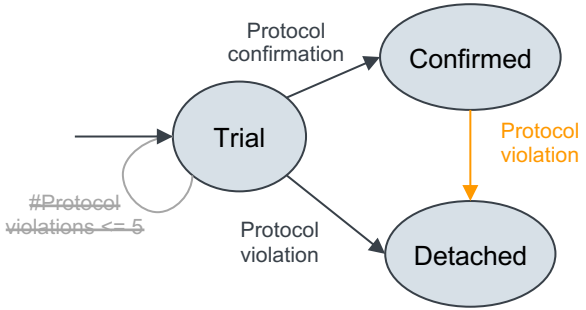

```
{
  "ts": 629503200,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "10.0.0.1",
  "id.orig_p": 51889,
  "id.resp_h": "192.168.0.1",
  "id.resp_p": 2323,
  "proto": "tcp",
  "service": "http",
  "duration": 0.0002599954605102539,
  "orig_bytes": 18,
  "resp_bytes": 12649,
  "conn_state": "SF",
  "local_orig": true,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADadFf",
  "orig_pkts": 15,
  "orig_ip_bytes": 618,
  "resp_pkts": 13,
  "resp_ip_bytes": 13169,
  "ip_proto": 6
}
```


- How to tell an analyzer *confirmed* and then got *detached*?
  - **Zeek 5.1 - 7.1:** Protocol won't show up in `service` field
  - **Zeek 7.2:** Protocol remains in `service` field



- Only generated for violations of previously confirmed analyzers
  - ⚠️ Analyzer got detached → no further analysis

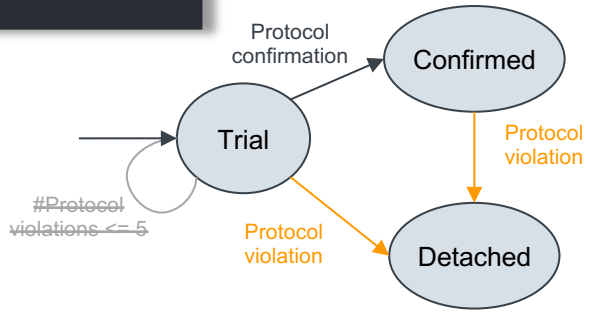

```
{
  "ts": 1669822677.033301,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "192.168.12.5",
  "id.orig_p": 51792,
  "id.resp_h": "192.0.78.212",
  "id.resp_p": 80,
  "proto": "tcp",
  "analyzer": "HTTP",
  "failure_reason": "not a http request line"
}
```

- Generated for all protocol violations for all analyzers (incl. file analyzers)
  - Primarily intended for debugging of analyzers in development phase


```
{
  "ts": 629503200.00015,
  "cause": "violation",
  "analyzer_kind": "protocol",
  "analyzer_name": "SSH",
  "uid": "CJKFoj4bpHEhTeaRoj",
  "fuid":      ,
  "id.orig_h": "10.0.0.1",
  "id.orig_p": 51889,
  "id.resp_h": "192.168.0.1",
  "id.resp_p": 22,
  "failure_reason": "Binpac exception: binpac exception: invalid index for case: SSH_Key_Exchange: 3",
  "failure_data":
}
```

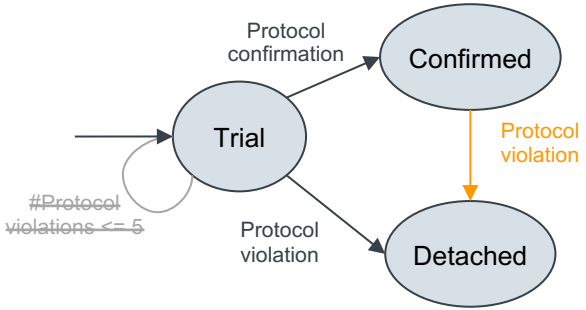

- Customizable
  - `Analyzer::Logging::include_confirmations = F`
  - `Analyzer::Logging::include_disabling = F`
  - `Analyzer::Logging::failure_data_max_size = 40`
  - `Analyzer::Logging::ignore_analyzers = set()`



DPD: Logs – conn.log


```
{
  "ts": 1669822676.914779,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "192.168.12.5",
  "id.orig_p": 51792,
  "id.resp_h": "192.0.78.212",
  "id.resp_p": 80,
  "proto": "tcp",
1. "service": "http,-http",
  "duration": 0.41950106620788574,
  "orig_bytes": 71,
  "resp_bytes": 699,
  "conn_state": "SF",
  "local_orig": true,
  "local_resp": false,
  "missed_bytes": 0,
  "history": "ShADadfF",
  "orig_pkts": 11,
  "orig_ip_bytes": 531,
  "resp_pkts": 9,
  "resp_ip_bytes": 1071,
  "ip_proto": 6,
2. "failed_service": [
    "http"
  ]
}
```


- New in Zeek 7.2: Keeping track of detached, previously confirmed analyzers in `conn.log`
  1. Add to **`service`** field using "-" prefix
     → `DPD::track_removed_services_in_connection = T`
  2. Add **`failed_service`** field
     → `@load protocols/conn/failed-service-logging`



HTTP traffic on port 22 (SSH)

Expectations?

DPD: Logs – conn.log 💡


```
{
  "ts": 1705192016.281275,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "127.0.0.1",
  "id.orig_p": 42906,
  "id.resp_h": "127.0.0.1",
  "id.resp_p": 8888,
  "proto": "tcp",
  "service": "http,websocket,ssh",
  "duration": 8.016665935516357,
  "orig_bytes": 4117,
  "resp_bytes": 2017,
  "conn_state": "RSTO",
  "local_orig": true,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADadR",
  "orig_pkts": 20,
  "orig_ip_bytes": 5165,
  "resp_pkts": 22,
  "resp_ip_bytes": 3169,
  "ip_proto": 6
}
```


- New in Zeek 7.2: The `service` field is ordered
  → reflects the order in which protocols got **confirmed**
  - Note: Confirmation order does not necessarily reflect nesting of protocol layers

corelight
OPEN SOURCE

DPD: Speculative Service 💡

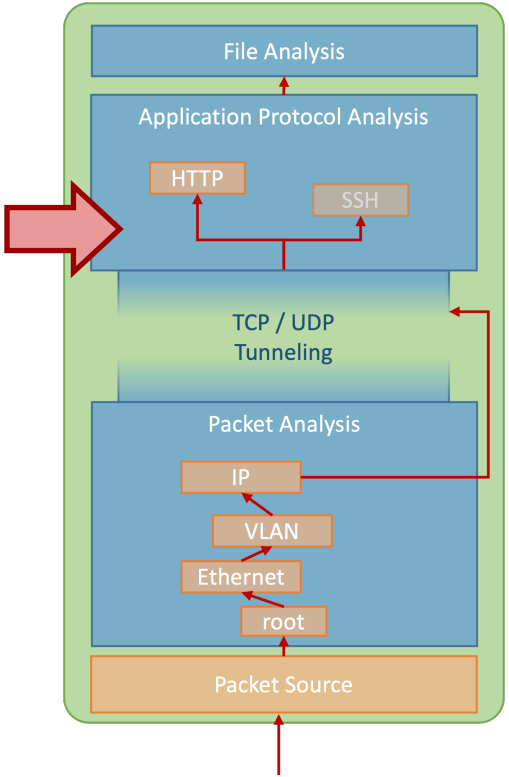
- Continuous matching to determine "speculative service"
  - Buffering is stopped once DPD-buffer is filled, and data gets deleted
  - `@load protocols/conn/speculative-service`
    → adds column to conn.log
    - `dpd_match_only_beginning = F`
      ⚠️ applies to all signatures
    - `dpd_late_match_stop = T`
  - No replay & no analyzer attached (resynchronization required)
  - Independent of previous successful DPD


```
{
  "ts": 629503200,
  "uid": "CJKFoj4bpHEhTeaRoj",
  "id.orig_h": "10.0.0.1",
  "id.orig_p": 51889,
  "id.resp_h": "192.168.0.1",
  "id.resp_p": 2323,
  "proto": "tcp",
  "service":      ,
  "duration": 0.0010809898376464844,
  "orig_bytes": 2137,
  "resp_bytes": 68729,
  "conn_state": "SF",
  "local_orig": true,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADadFf",
  "orig_pkts": 56,
  "orig_ip_bytes": 4377,
  "resp_pkts": 54,
  "resp_ip_bytes": 70889,
  "speculative_service": "http"
}
```

Application Protocol Analysis → Dynamic Protocol Detection

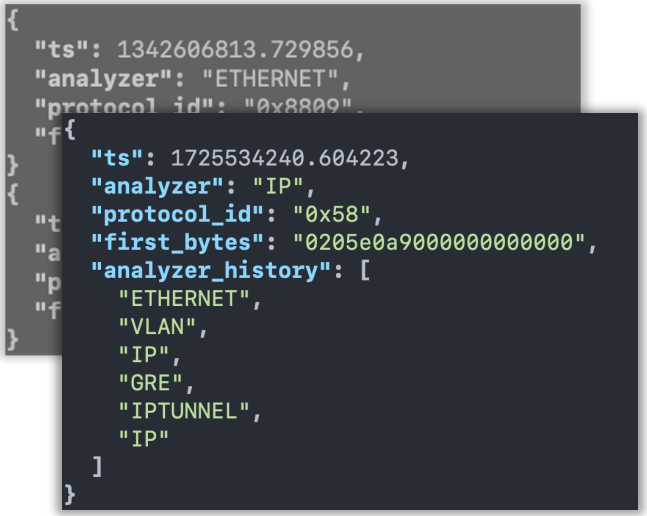
- Trial and error approach using Ports and Signatures
- Trade-offs can be tuned (buffer capacity, violation limit)
- `dpd.log` and `analyzer.log` inform about failed attempts

→ Zeek implements clever approaches to increase visibility
→ Zeek tells you what it “did not understand”



Packet Analysis: unknown_protocols.log – Analyzer History 💡

- Logs protocol IDs packet analyzers cannot map
  → `@load policy/misc/unknown-protocols`



- `UnknownProtocol::sampling_threshold = 3`
  → enter sampling mode if (analyzer, ID) pair was seen 3 times
- `UnknownProtocol::sampling_rate = 100,000`
  → in sampling mode log every 100,000 occurrence of an (analyzer, ID) pair
- `UnknownProtocol::sampling_duration = 1 hr`
  → expire sampling after 1 hr without observation of an (analyzer, ID) pair

← New in Zeek 7.1: Analyzer History