Meet "Spicy"

You can write a protocol parser now!

Robin Sommer

Co-Founder

Corelight, Inc.

robin@corelight.com

corelight

zeek

About me


PADERBORN
UNIVERSITY


SAARLAND
UNIVERSITY


TECHNISCHE
UNIVERSITÄT
MÜNCHEN


ICSI
INTERNATIONAL
COMPUTER SCIENCE
INSTITUTE


BERKELEY LAB
Lawrence Berkeley National Laboratory


corelight

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 endpoints
> 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

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 packets (e.g., SYN & RST)
G	Content **G**ap
Q	Multi-flag packet (SYN & FIN or SYN + RST)
T	Re**t**ransmitted packet
W	Packet with zero **w**indow advertisement
^	Flipped connection

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
client_port	port	Client port at time of server handing out IP
server_port	port	Server port at time of server handing out IP
mac	string	Client's hardware address
host_name	string	Name given by client in Hostname option 12
client_fqdn	string	FQDN given by client in Client FQDN option 81
domain	string	Domain given by server in option 15
requested_addr	addr	IP address requested by client
assigned_addr	addr	IP address assigned by server
lease_time	interval	IP address lease interval
client_message	string	Message with DHCP_DECLINE so client can tell server why address was rejected
server_message	string	Message with DHCP_NAK to let client know why request was rejected
msg_types	vector	DHCP message types seen by transaction
duration	interval	Duration of DHCP session
client_chaddr	string	Hardware address reported by the client
msg_orig	vector	Address originated from msg_types field
client_software	string	Software reported by client in vendor_class
server_software	string	Software reported by server in vendor_class
circuit_id	string	DHCP relay agents that terminate circuits
agent_remote_id	string	Globally unique ID added by relay agents to identify remote host end of circuit
subscriber_id	string	Value independent of physical network connection that provides customer DHCP configuration regardless of physical location

dns.log | DNS query/response details

FIELD	TYPE	DESCRIPTION
ts	time	Earliest timestamp of DNS protocol message
uid & id		Underlying connection info > **See conn.log**
proto	enum	Transport layer protocol of connection
trans_id	count	16-bit identifier assigned by program that generated DNS query
rtt	interval	Round trip time for query and response
query	string	Domain name subject of DNS query
qclass	count	QCLASS value specifying query class
qclass_name	string	Descriptive name query class
qtype	count	QTYPE value specifying query type
qtype_name	string	Descriptive name for query type
rcode	count	Response code value in DNS response

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

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

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

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.*

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

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

ssl.log | SSL handshakes

FIELD	TYPE	DESCRIPTION
ts	time	Time when SSL connection first detected
uid & id		Underlying connection info > **See conn.log**
version	string	SSL/TLS version server chose
cipher	string	SSL/TLS cipher suite server chose
curve	string	Elliptic curve server chose when using ECDH/ECDHE
server_name	string	Value of Server Name Indicator SSL/TLS extension
resumed	bool	Flag that indicates session was resumed
last_alert	string	Last alert seen during connection
next_protocol	string	Next protocol server chose using application layer next protocol extension, if present
established	bool	Flags if SSL session successfully established
ssl_history	string	SSL history showing which types of packets were received in which order. Client-side letters are capitalized, server-side lowercase.

ssl_history

^	direction flipped	U	certificate_status
H	**h**ello_request	A	supplemental_**d**ata
C	**C**lient_hello	Z	unassigned_handshake_type
S	**S**erver_hello	I	change_c**i**pher_spec
V	hello_**V**erify_request	B	heart**B**eat
T	NewSessionTicket	D	application_**d**ata
X	certificate**X**	E	end_of_**e**arly_data
K	server_**K**ey_exchange	O	encrypted_extensi**o**ns
R	certificate_**R**equest	P	key_**p**date
N	server_hello_do**N**e	M	**M**essage_hash
Y	certificate_verif**Y**	J	hello_retry_request
G	client_key_exchan**G**e	L	a**L**ert
F	**F**inished	Q	unknown_content_type
W	certificate_url		

cert_chain_fps	vector	All fingerprints for the certificates offered by the server
client_cert_chain_fps	vector	All fingerprints for the certificates offered by the client
subject	string	Subject of X.509 cert offered by server
issuer	string	Subject of signer of X.509 server cert
client_subject	string	Subject of X.509 cert offered by client
client_issuer	string	Subject of signer of client cert
sni_matches_cert	bool	Set to true if the hostname sent in the SNI matches the certificate, false if it does not. Unset if the client did not send an SNI.
request_client_certificate_authorities	vector	List of client certificate CAs accepted by the server
server_version	count	Numeric version of the server in the server hello
client_version	count	Numeric version of the client in the client hello
client_ciphers	vector	Ciphers that were offered by the client for the connection
ssl_client_exts	vector	SSL client extensions
ssl_server_exts	vector	SSL server extensions
ticket_lifetime_hint	count	Suggested ticket lifetime sent in the session ticket handshake by the server
dh_param_size	count	The diffie helman parameter size, when using DH
point_formats	vector	Supported elliptic curve point formats
client_curves	vector	The curves supported by the client
orig_alpn	vector	Application layer protocol negotiation extension sent by the client
client_supported_versions	vector	TLS 1.3 supported versions
server_supported_version	count	TLS 1.3 supported version
psk_key_exchange_modes	vector	TLS 1.3 Pre-shared key exchange modes
client_key_share_groups	vector	Key share groups from client hello
server_key_share_group	count	Selected key share group from server hello
client_comp_methods	vector	Client supported compression methods
sigalgs	vector	Client supported signature algorithms
hashalgs	vector	Client supported hash algorithms
validation_status	string	Certificate validation result for this connection
ocsp_status	string	OCSP validation result for this connection
valid_ct_logs	count	Number of different logs for which valid SCTs encountered in connection
valid_ct_operators	count	Number of different log operators for which valid SCTs encountered in connection

dce_rpc.log | Details on DCE/RPC messages

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > **See conn.log**
rtt	interval	Round trip time from request to response
named_pipe	string	Remote pipe name
endpoint	string	Endpoint name looked up from uuid
operation	string	Operation seen in call

ntlm.log | NT LAN Manager (NTLM)

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > **See conn.log**
username	string	Username given by client
hostname	string	Hostname given by client
domainname	string	Domainname given by client
server_nb_computer_name	string	NetBIOS name given by server in a CHALLENGE
server_dns_computer_name	string	DNS name given by server in a CHALLENGE
server_tree_name	string	Tree name given by server in a CHALLENGE
success	bool	Indicates whether or not authentication was successful

rdp.log | Remote Desktop Protocol (RDP)

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > **See conn.log**
cookie	string	Cookie value used by client machine
result	string	Status result for connection
security_protocol	string	Security protocol chosen by server
client_channels	vector	Channels requested by the client
keyboard_layout	string	Keyboard layout (language) of client machine
client_build	string	RDP client version used by client machine
client_name	string	Name of client machine
client_dig_product_id	string	Product ID of client machine
desktop_width	count	Desktop width of client machine
desktop_height	count	Desktop height of client machine
requested_color_depth	string	Color depth requested by client in high_color_depth field
cert_type	string	If connection is encrypted with native RDP encryption, type of cert being used
cert_count	count	Number of certs seen
cert_permanent	bool	Indicates if provided certificate or certificate chain is permanent or temporary
encryption_level	string	Encryption level of connection
encryption_method	string	Encryption method of connection
ssl	bool	Flag connection if seen over SSL

smb_files.log | Details on SMB files

FIELD	TYPE	DESCRIPTION
ts	time	Time when file was first discovered
uid & id		Underlying connection info > **See conn.log**
fuid	string	Unique ID of file
action	enum	Action this log record represents
path	string	Path pulled from tree that file was transferred to or from
name	string	Filename if one was seen
size	count	Total size of file
prev_name	string	If rename action was seen, this will be file's previous name
times	record SMB::MAC-Times	Last time file was modified

smb_mapping.log | SMB mappings

FIELD	TYPE	DESCRIPTION
ts	time	Time when tree was mapped
uid & id		Underlying connection info > **See conn.log**
path	string	Name of tree path
service	string	Type of resource of tree (disk share, printer share, named pipe, etc)
native_file_system	string	File system of tree
share_type	string	If this is SMB2, share type will be included

rcode	count	Response code value in DNS response
rcode_name	string	Descriptive name of response code value
AA	bool	Authoritative Answer bit: responding name server is authority for domain name
TC	bool	Truncation bit: message was truncated
RD	bool	Recursion Desired bit: client wants recursive service for query
RA	bool	Recursion Available bit: name server supports recursive queries
Z	count	Reserved field, usually zero in queries and responses
answers	vector	Set of resource descriptions in query answer
TTLs	vector	Caching intervals of RRs in answers field
rejected	bool	DNS query was rejected by server
auth	table	Authoritative responses for query
addl	table	Additional responses for query

dpd.log | Dynamic protocol detection failures

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when protocol analysis failed
uid & id		Underlying connection info > **See conn.log**
proto	enum	Transport protocol for violation
analyzer	string	Analyzer that generated violation
failure_reason	string	Textual reason for analysis failure
packet_segment	string	Payload chunk that most likely resulted in protocol violation

files.log | File analysis results

FIELD	TYPE	DESCRIPTION
ts	time	Time when file first seen
fuid	string	Identifier associated with single file
uid & id		Underlying connection info > **See conn.log**
source	string	Identification of file data source
depth	count	Value to represent depth of file in relation to source
analyzers	table	Set of analysis types done during file analysis
mime_type	string	Mime type, as determined by Zeek's signatures
filename	string	Filename, if available from file source
duration	interval	Duration file was analyzed for
local_orig	bool	Indicates if data originated from local network
is_orig	bool	If file sent by connection originator or responder
seen_bytes	count	Number of bytes provided to file analysis engine
total_bytes	count	Total number of bytes that should comprise full file
missing_bytes	count	Number of bytes in file stream missed
overflow_bytes	count	Number of bytes in file stream not delivered to stream file analyzers
timedout	bool	If file analysis timed out at least once
parent_fuid	string	Container file ID was extracted from
md5	string	MD5 digest of file contents
sha1	string	SHA1 digest of file contents
sha256	string	SHA256 digest of file contents
extracted	string	Local filename of extracted file
extracted_cutoff	bool	Set to true if file being extracted was cut off so whole file was not logged
extracted_size	count	Number of bytes extracted to disk
entropy	double	Information density of file contents

ftp.log | FTP request/reply details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when command sent
uid & id		Underlying connection info > **See conn.log**
user	string	Username for current FTP session
password	string	Password for current FTP session
command	string	Command given by client
arg	string	Argument for command, if given
mime_type	string	Sniffed mime type of file
file_size	count	Size of file
reply_code	count	Reply code from server in response to command
reply_msg	string	Reply message from server in response to command
data_channel	record FTP::Expected Data Channel	Expected FTP data channel

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

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
error_msg	string	Error message
from	time	Ticket valid from
till	time	Ticket valid until
cipher	string	Ticket encryption type
forwardable	bool	Forwardable ticket requested
renewable	bool	Renewable ticket requested
client_cert_subject	string	Subject of client certificate, if any
client_cert_fuid	string	File unique ID of client cert, if any
server_cert_subject	string	Subject of server certificate, if any
server_cert_fuid	string	File unique ID of server cert, if any
auth_ticket	string	Ticket hash authorizing request/transaction
new_ticket	string	Ticket hash returned by KDC

mysql.log | MySQL

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > **See conn.log**
cmd	string	Command that was issued
arg	string	Argument issued to command
success	bool	Server replied command succeeded
rows	count	Number of affected rows, if any
response	string	Server message, if any

pe.log | Portable executable

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
id	string	File id of this portable executable file
machine	string	Target machine file was compiled for
compile_ts	time	Time file was created
os	string	Required operating system
subsystem	string	Subsystem required to run this file
is_exe	bool	Is file an executable, or just an object file?
is_64bit	bool	Is file a 64-bit executable?
uses_aslr	bool	Does file support Address Space Layout Randomization?
uses_dep	bool	Does file support Data Execution Prevention?
uses_code_integrity	bool	Does file enforce code integrity checks?
uses_seh	bool	Does file use structured exception handing?
has_import_table	bool	Does file have import table?
has_export_table	bool	Does file have export table?
has_cert_table	bool	Does file have attribute certificate table?
has_debug_data	bool	Does file have debug table?
section_names	vector of string	Names of sections, in order

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 GetResponse/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

socks.log | SOCKS proxy requests

FIELD	TYPE	DESCRIPTION
ts	time	Time when proxy connection detected
uid & id		Underlying connection info > **See conn.log**
version	count	Protocol version of SOCKS
user	string	Username used to request a login to proxy
password	string	Password used to request a login to proxy
status	string	Server status for attempt at using proxy
request	record SOCKS::Address	Client requested SOCKS address
request_p	port	Client requested port
bound	record SOCKS::Address	Server bound address
bound_p	port	Server bound port

software.log | Software observed on network

FIELD	TYPE	DESCRIPTION
ts	time	Time at which software was detected
host	addr	IP address detected running the software
host_p	port	Port on which software is running
software_type	enum	Type of software detected (e.g., HTTP::SERVER)
name	string	Name of software (e.g., Apache)
version	record Software::Version	Software version
unparsed_version	string	Full, unparsed version string found
url	string	Root URL where software was discovered

ssh.log | SSH handshakes

FIELD	TYPE	DESCRIPTION
ts	time	Time when SSH connection began
uid & id		Underlying connection info > **See conn.log**
version	count	SSH major version (1 or 2)
auth_success	bool	Authentication result (T=success, F=failure, unset=unknown)
auth_attempts	count	Number of authentication attempts observed
direction	enum	Direction of connection
client	string	Client's version string
server	string	Server's version string
cipher_alg	string	Encryption algorithm in use
mac_alg	string	Signing (MAC) algorithm in use
compression_alg	string	Compression algorithm in use
kex_alg	string	Key exchange algorithm in use
host_key_alg	string	Server host key's algorithm
host_key	string	Server's key fingerprint
remote_location	record geo_location	Add geographic data related to remote host of connection

corelight

https://go.corelight.com/zeek-log-poster-digital

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 endpoints
> 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

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 packets (e.g., SYN & RST)
G	Content **G**ap
Q	Multi-flag packet (SYN & FIN or SYN + RST)
T	Re**t**ransmitted packet
W	Packet with zero **w**indow advertisement
^	Flipped connection

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
client_port	port	Client port at time of server handing out IP
server_port	port	Server port at time of server handing out IP
mac	string	Client's hardware address
host_name	string	Name given by client in Hostname option 12
client_fqdn	string	FQDN given by client in Client FQDN option 81
domain	string	Domain given by server in option 15
requested_addr	addr	IP address requested by client
assigned_addr	addr	IP address assigned by server
lease_time	interval	IP address lease interval
client_message	string	Message with DHCP_DECLINE so client can tell server why address was rejected
server_message	string	Message with DHCP_NAK to let client know why request was rejected
msg_types	vector	DHCP message types seen by transaction
duration	interval	Duration of DHCP session
client_chaddr	string	Hardware address reported by the client
msg_orig	vector	Address originated from msg_types field
client_software	string	Software reported by client in vendor_class
server_software	string	Software reported by server in vendor_class
circuit_id	string	DHCP relay agents that terminate circuits
agent_remote_id	string	Globally unique ID added by relay agents to identify remote host end of circuit
subscriber_id	string	Value independent of physical network connection that provides customer DHCP configuration regardless of physical location

dns.log | DNS query/response details

FIELD	TYPE	DESCRIPTION
ts	time	Earliest timestamp of DNS protocol message
uid & id		Underlying connection info > **See conn.log**
proto	enum	Transport layer protocol of connection
trans_id	count	16-bit identifier assigned by program that generated DNS query
rtt	interval	Round trip time for query and response
query	string	Domain name subject of DNS query
qclass	count	QCLASS value specifying query class
qclass_name	string	Descriptive name query class
qtype	count	QTYPE value specifying query type
qtype_name	string	Descriptive name for query type
rcode	count	Response code value in DNS response
rcode_name	string	Descriptive name of response code value
AA	bool	Authoritative Answer bit: responding name server is authority for domain name
TC	bool	Truncation bit: message was truncated
RD	bool	Recursion Desired bit: client wants recursive service for query
RA	bool	Recursion Available bit: name server supports recursive queries
Z	count	Reserved field, usually zero in queries and responses
answers	vector	Set of resource descriptions in query answer
TTLs	vector	Caching intervals of RRs in answers field
rejected	bool	DNS query was rejected by server
auth	table	Authoritative responses for query
addl	table	Additional responses for query

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

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

radius.log | RADIUS authentic

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when ev
uid & id		Underlying connection
username	string	Username, if present
mac	string	MAC address, if presen
framed_addr	addr	Address given to netwo if present
tunnel_client	string	Address (IPv4, IPv6, or f end of tunnel, if presen
connect_info	string	Connect info, if present
reply_msg	string	Reply message from se
result	string	Successful or failed aut
ttl	interval	Duration between first Access-Accept message

sip.log | SIP analysis

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when reque
uid & id		Underlying connection
trans_depth	count	Pipelined depth into re transaction
method	string	Verb used in SIP reques
uri	string	URI used in request
date	string	Contents of Date: head
request_from	string	Contents of request Fro
request_to	string	Contents of To: header
response_from	string	Contents of response F
response_to	string	Contents of response T
reply_to	string	Contents of Reply-To: h
call_id	string	Contents of Call-ID: hea
seq	string	Contents of CSeq: head
subject	string	Contents of Subject: he
request_path	vector	Client message transm from headers
response_path	vector	Server message transm extracted from headers
user_agent	string	Contents of User-Agent
status_code	count	Status code returned by
status_msg	string	Status message returne
warning	string	Contents of Warning: h
request_body_len	count	Contents of Content-Le client
response_body_len	count	Contents of Content-Le server
content_type	string	Contents of Content-Ty server

[1] *The tag= value usually appended to the sender is stripped*

smtp.log | SMTP transactions

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when mes
uid & id		Underlying connection
trans_depth	count	Transaction depth if the
helo	string	Contents of Helo head
mailfrom	string	Email addresses found
rcptto	table	Email addresses found
date	string	Contents of Date head
from	string	Contents of From head
to	table	Contents of To header
cc	table	Contents of CC header
reply_to	string	Contents of ReplyTo he
msg_id	string	Contents of MsgID hea
in_reply_to	string	Contents of In-Reply-To
subject	string	Contents of Subject he
x_originating_ip	addr	Contents of X-Originat
first_received	string	Contents of first Recei
second_received	string	Contents of second Re
last_reply	string	Last message server se
path	vector	Message transmission
user_agent	string	Value of User-Agent he
tls	bool	Indicates connection s
fuids	vector	File unique IDs attache
is_webmail	bool	If message sent via we

snmp.log | SNMP messages

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp of first pack
uid & id		Underlying connection
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

ssl.log | SSL handshake

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when SSL connection detected
uid & id		Underlying connection info > **See conn.log**
version	string	SSL version server offered
cipher	string	SSL cipher suite server chose
server_name	string	Value of Server Name Indication SSL extension
session_id	string	Session ID offered by client for session resumption
subject	string	Subject of the X.509 cert offered by server
issuer	string	Subject of signer of server cert
client_subject	string	Subject of the X.509 cert offered by client
client_issuer	string	Subject of signer of client cert
cert_hash	string	MD5 hash of server cert, client cert, or both
validation_status	string	Certificate validation result for this handshake
ocsp_status	string	Result of OCSP validation
valid_ct_logs	count	Number of valid logs used to validate SCTs encountered in connection
valid_ct_operators	count	Number of different log operators for which valid SCTs encountered in connection

dce_rpc.log | Details on DCE/RPC messages

FIELD	TYPE	DESCRIPTION
rtt	interval	Round trip time from request to response
named_pipe	string	Remote pipe name
endpoint	string	Endpoint name looked up
operation	string	Operation seen in call

dpd.log | Dynamic protocol detection failures

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when protocol analysis failed
uid & id		Underlying connection info > **See conn.log**
proto	enum	Transport protocol for violation
analyzer	string	Analyzer that generated violation
failure_reason	string	Textual reason for analysis failure
packet_segment	string	Payload chunk that most likely resulted in protocol violation

files.log | File analysis results

FIELD	TYPE	DESCRIPTION
ts	time	Time when file first seen
fuid	string	Identifier associated with single file
uid & id		Underlying connection info > **See conn.log**
source	string	Identification of file data source
depth	count	Value to represent depth of file in relation to source
analyzers	table	Set of analysis types done during file analysis
mime_type	string	Mime type, as determined by Zeek's signatures
filename	string	Filename, if available from file source
duration	interval	Duration file was analyzed for
local_orig	bool	Indicates if data originated from local network
is_orig	bool	If file sent by connection originator or responder
seen_bytes	count	Number of bytes provided to file analysis engine
total_bytes	count	Total number of bytes that should comprise full file
missing_bytes	count	Number of bytes in file stream missed
overflow_bytes	count	Number of bytes in file stream not delivered to stream file analyzers
timedout	bool	If file analysis timed out at least once
parent_fuid	string	Container file ID was extracted from
md5	string	MD5 digest of file contents
sha1	string	SHA1 digest of file contents
sha256	string	SHA256 digest of file contents
extracted	string	Local filename of extracted file
extracted_cutoff	bool	Set to true if file being extracted was cut off so whole file was not logged
extracted_size	count	Number of bytes extracted to disk
entropy	double	Information density of file contents

ftp.log | FTP request/reply details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when command sent
uid & id		Underlying connection info > **See conn.log**
user	string	Username for current FTP session
password	string	Password for current FTP session
command	string	Command given by client
arg	string	Argument for command, if given
mime_type	string	Sniffed mime type of file
file_size	count	Size of file
reply_code	count	Reply code from server in response to command
reply_msg	string	Reply message from server in response to command
data_channel	record FTP::Expected Data Channel	Expected FTP data channel

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
error_msg	string	Error message
from	time	Ticket valid from
till	time	Ticket valid until
cipher	string	Ticket encryption type
forwardable	bool	Forwardable ticket requested
renewable	bool	Renewable ticket requested
client_cert_subject	string	Subject of client certificate, if any
client_cert_fuid	string	File unique ID of client cert, if any
server_cert_subject	string	Subject of server certificate, if any
server_cert_fuid	string	File unique ID of server cert, if any
auth_ticket	string	Ticket hash authorizing request/transaction
new_ticket	string	Ticket hash returned by KDC

mysql.log | MySQL

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
uid & id		Underlying connection info > **See conn.log**
cmd	string	Command that was issued
arg	string	Argument issued to command
success	bool	Server replied command succeeded
rows	count	Number of affected rows, if any
response	string	Server message, if any

pe.log | Portable executable

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when event happened
id	string	File id of this portable executable file
machine	string	Target machine file was compiled for
compile_ts	time	Time file was created
os	string	Required operating system
subsystem	string	Subsystem required to run this file
is_exe	bool	Is file an executable, or just an object file?
is_64bit	bool	Is file a 64-bit executable?
uses_aslr	bool	Does file support Address Space Layout Randomization?
uses_dep	bool	Does file support Data Execution Prevention?
uses_code_integrity	bool	Does file enforce code integrity checks?
uses_seh	bool	Does file use structured exception handing?
has_import_table	bool	Does file have import table?
has_export_table	bool	Does file have export table?
has_cert_table	bool	Does file have attribute certificate table?
has_debug_data	bool	Does file have debug table?
section_names	vector of string	Names of sections, in order

snmp.log | SNMP messages

FIELD	TYPE	DESCRIPTION
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

socks.log | SOCKS proxy requests

FIELD	TYPE	DESCRIPTION
ts	time	Time when proxy connection detected
uid & id		Underlying connection info > **See conn.log**
version	count	Protocol version of SOCKS
user	string	Username used to request a login to proxy
password	string	Password used to request a login to proxy
status	string	Server status for attempt at using proxy
request	record SOCKS::Address	Client requested SOCKS address
request_p	port	Client requested port
bound	record SOCKS::Address	Server bound address
bound_p	port	Server bound port

software.log | Software observed on network

FIELD	TYPE	DESCRIPTION
ts	time	Time at which software was detected
host	addr	IP address detected running the software
host_p	port	Port on which software is running
software_type	enum	Type of software detected (e.g., HTTP::SERVER)
name	string	Name of software (e.g., Apache)
version	record Software::Version	Software version
unparsed_version	string	Full, unparsed version string found
url	string	Root URL where software was discovered

ssh.log | SSH handshakes

FIELD	TYPE	DESCRIPTION
ts	time	Time when SSH connection began
uid & id		Underlying connection info > **See conn.log**
version	count	SSH major version (1 or 2)
auth_success	bool	Authentication result (T=success, F=failure, unset=unknown)
auth_attempts	count	Number of authentication attempts observed
direction	enum	Direction of connection
client	string	Client's version string
server	string	Server's version string
cipher_alg	string	Encryption algorithm in use
mac_alg	string	Signing (MAC) algorithm in use
compression_alg	string	Compression algorithm in use
kex_alg	string	Key exchange algorithm in use
host_key_alg	string	Server host key's algorithm
host_key	string	Server's key fingerprint
remote_location	record geo_location	Add geographic data related to remote host of connection

http.log | HTTP request/reply details

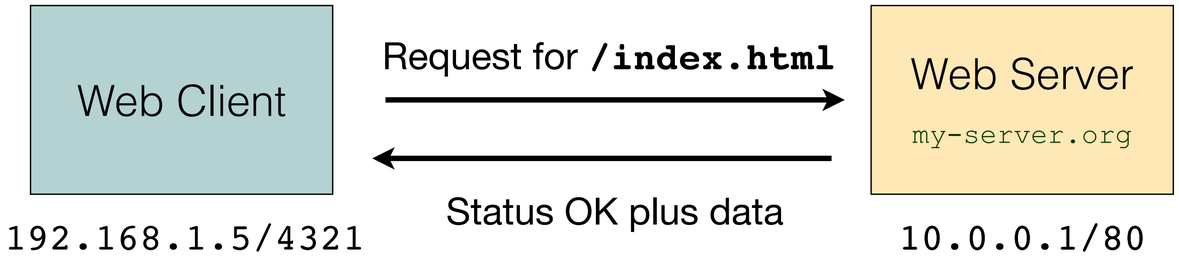
FIELD	TYPE	DESCRIPTION
ts	time	Timestamp for when request happened
uid & id		Underlying connection info > **See conn.log**
trans_depth	count	Pipelined depth into connection
method	string	Verb used in HTTP request (GET, POST, etc.)
host	string	Value of HOST header
uri	string	URI used in request
referrer	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

corelight

https://go.corelight.com/zeek-log-poster-digital

Behind the scenes: Processing HTTP

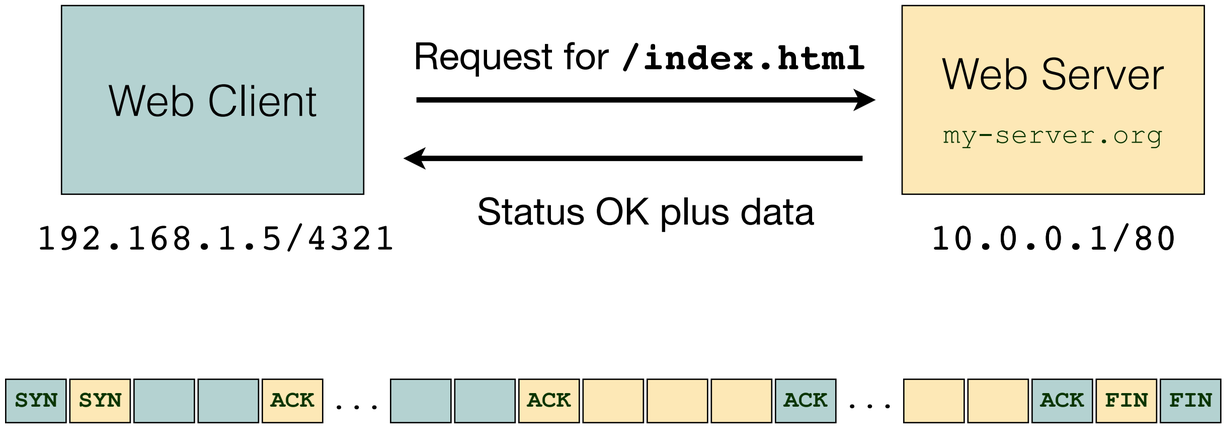
Behind the scenes: Processing HTTP



http.log | HTTP request/reply details

Field	Value
id.orig_h	
id.orig_p	
id.resp_h	
id.resp_p	
method	
host	
uri	
status_code	
status_msg	
…	

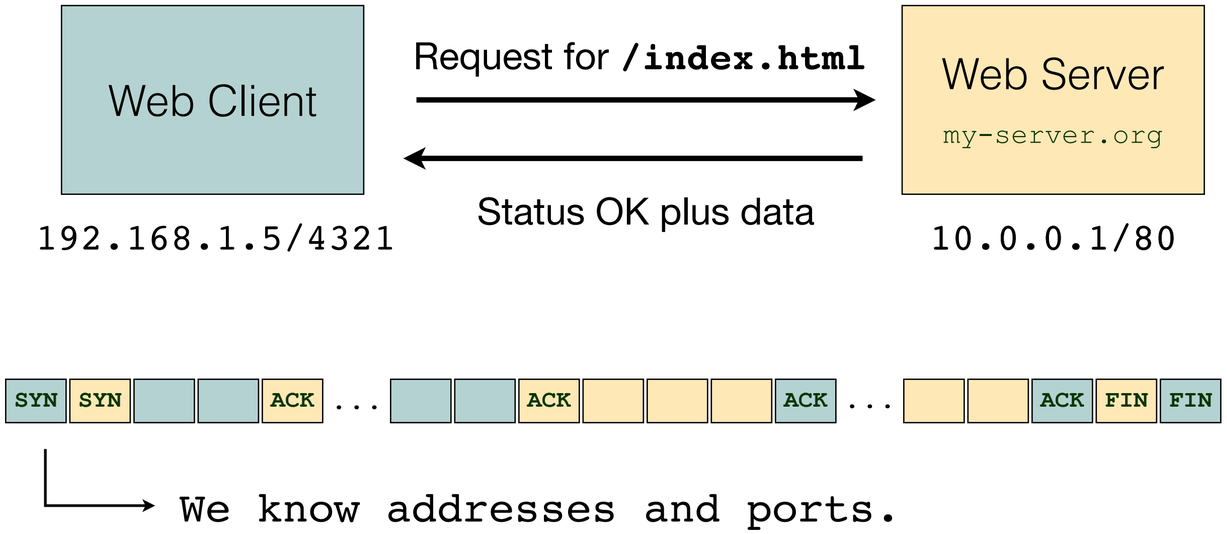
Behind the scenes: Processing HTTP



http.log | HTTP request/reply details

Field	Value
id.orig_h	
id.orig_p	
id.resp_h	
id.resp_p	
method	
host	
uri	
status_code	
status_msg	
…	

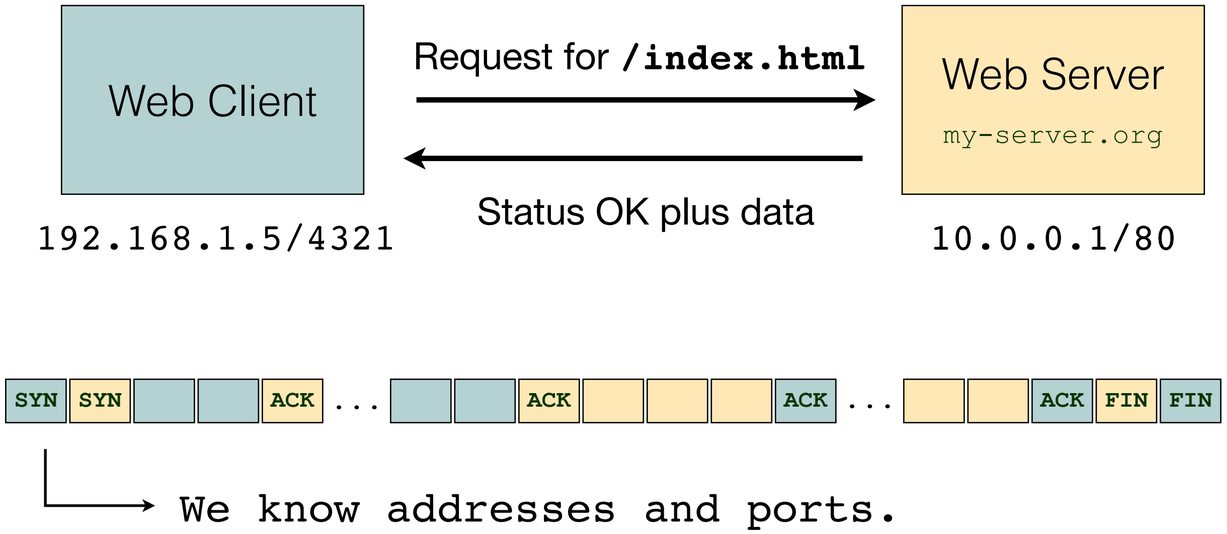
Behind the scenes: Processing HTTP



http.log | HTTP request/reply details

Field	Value
id.orig_h	
id.orig_p	
id.resp_h	
id.resp_p	
method	
host	
uri	
status_code	
status_msg	
…	

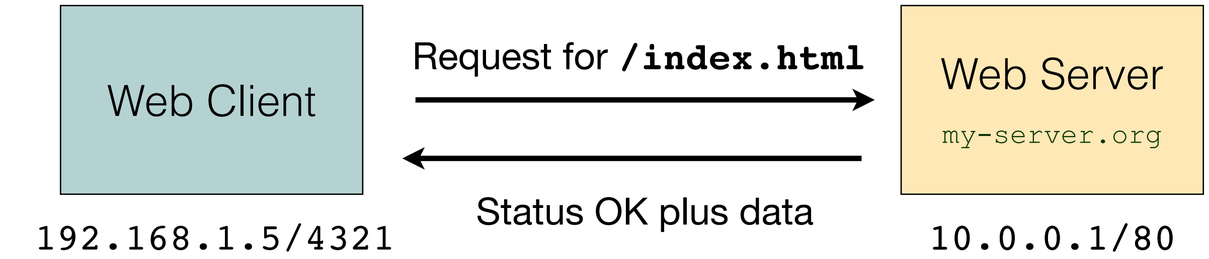
Behind the scenes: Processing HTTP

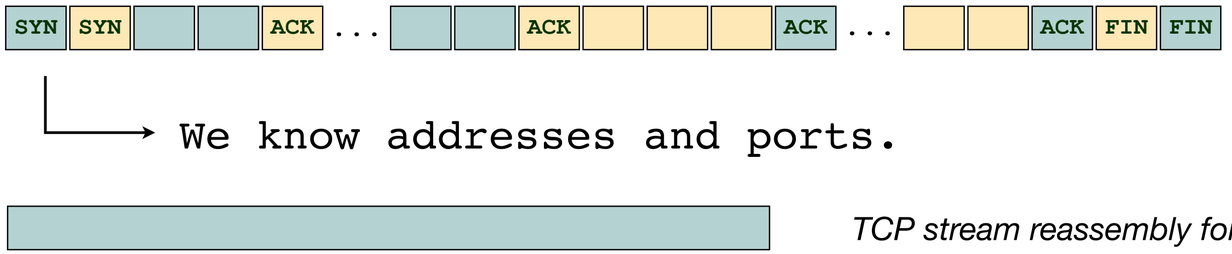


http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	
host	
uri	
status_code	
status_msg	
…	

Behind the scenes: Processing HTTP



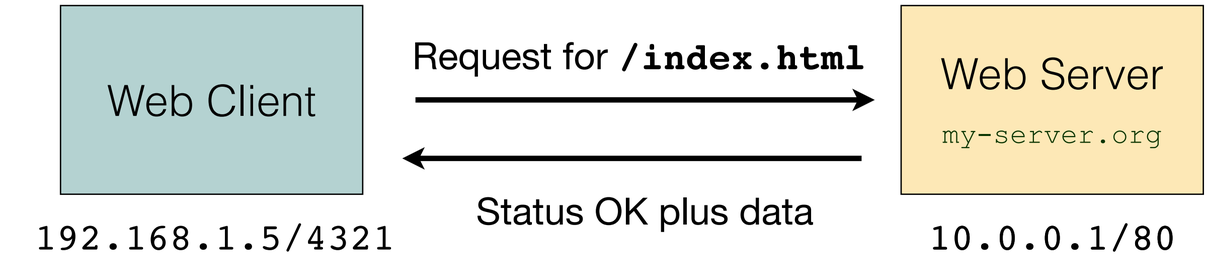


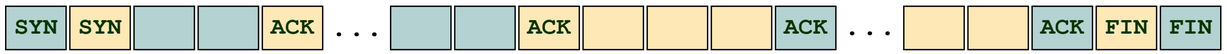
TCP stream reassembly for originator

http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	
host	
uri	
status_code	
status_msg	
…	

Behind the scenes: Processing HTTP





We know addresses and ports.

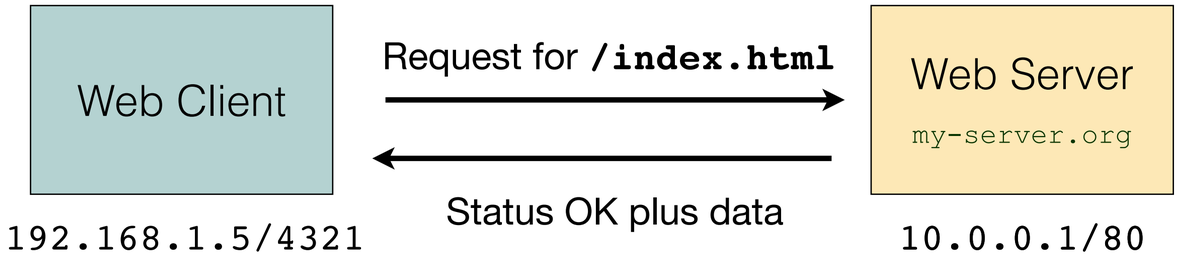
```
GET /index.html HTTP/1.1 …  Host: my-server.org …
```

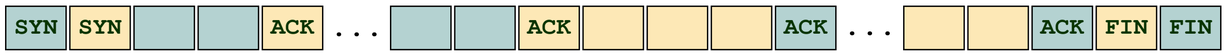
TCP stream reassembly for originator

http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	
host	
uri	
status_code	
status_msg	
…	

Behind the scenes: Processing HTTP





We know addresses and ports.

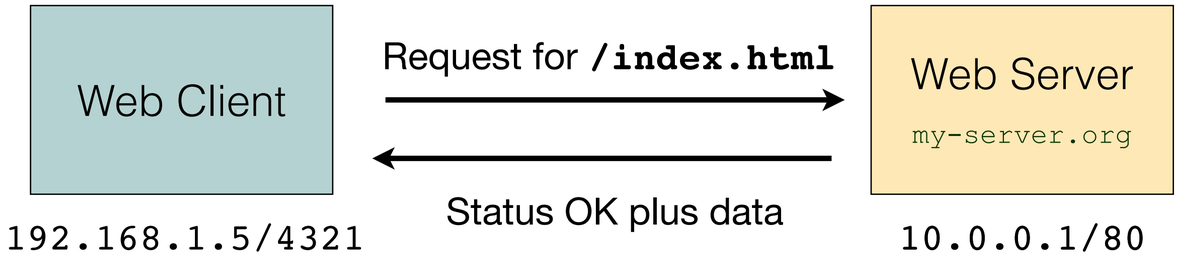
TCP stream reassembly for originator

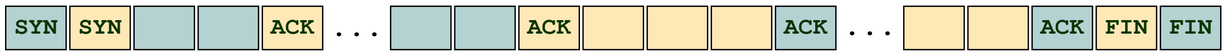
We know what's being requested.

http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	
host	
uri	
status_code	
status_msg	
…	

Behind the scenes: Processing HTTP





We know addresses and ports.

GET /index.html HTTP/1.1 … Host: my-server.org …

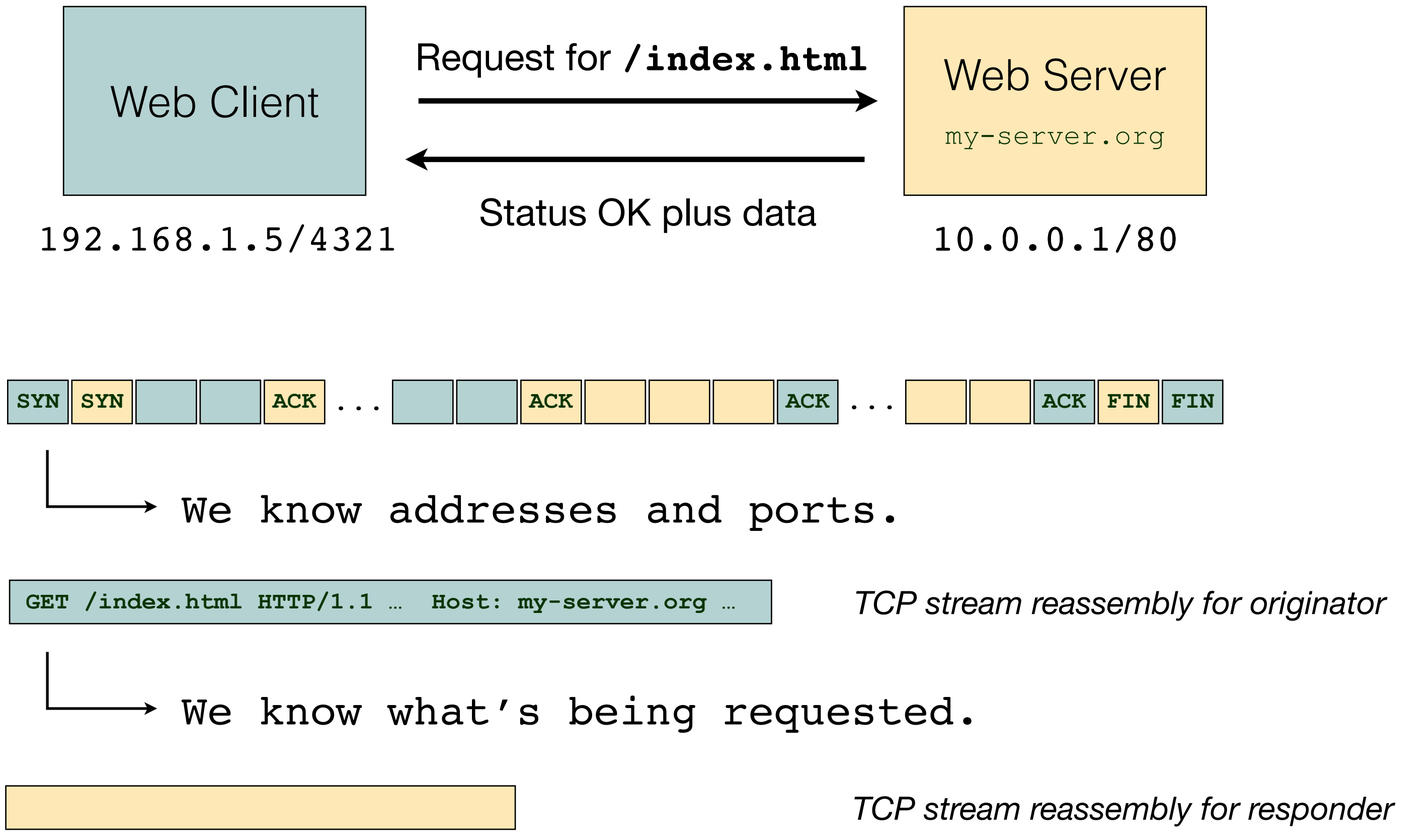
TCP stream reassembly for originator

We know what's being requested.

http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	GET
host	my-server.org
uri	/index.html
status_code	
status_msg	
…	

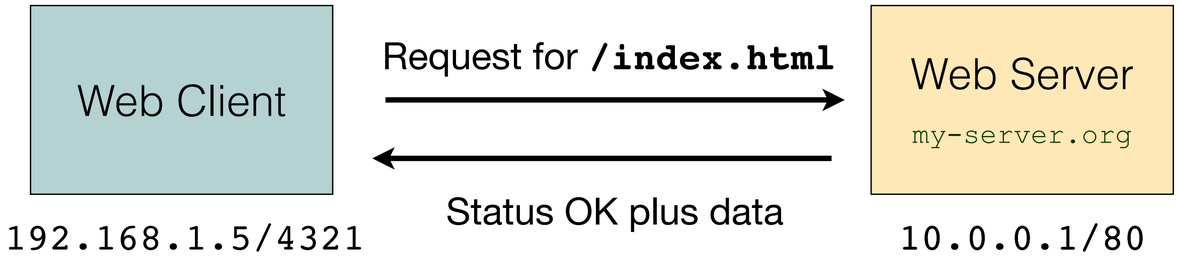
Behind the scenes: Processing HTTP

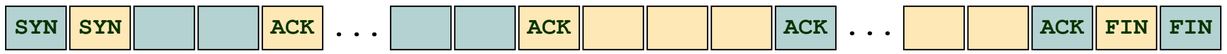


http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	GET
host	my-server.org
uri	/index.html
status_code	
status_msg	
…	

Behind the scenes: Processing HTTP





We know addresses and ports.

`GET /index.html HTTP/1.1 … Host: my-server.org …`

TCP stream reassembly for originator

We know what's being requested.

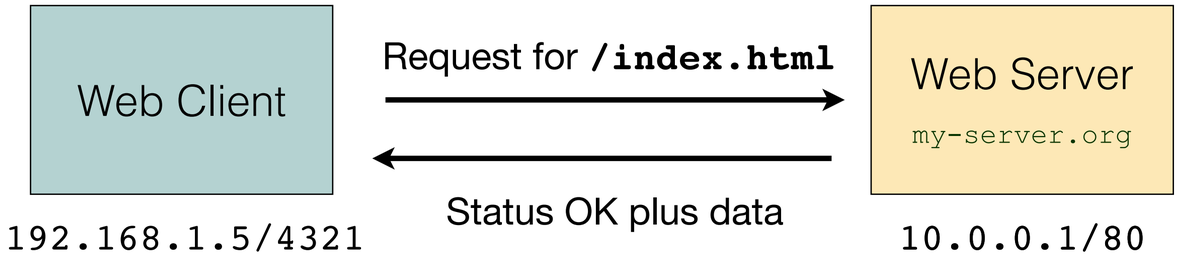
`200 OK … <HTML> …`

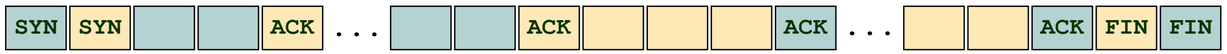
TCP stream reassembly for responder

http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	GET
host	my-server.org
uri	/index.html
status_code	
status_msg	
…	

Behind the scenes: Processing HTTP





We know addresses and ports.

GET /index.html HTTP/1.1 … Host: my-server.org …

TCP stream reassembly for originator

We know what's being requested.

200 OK … <HTML> …

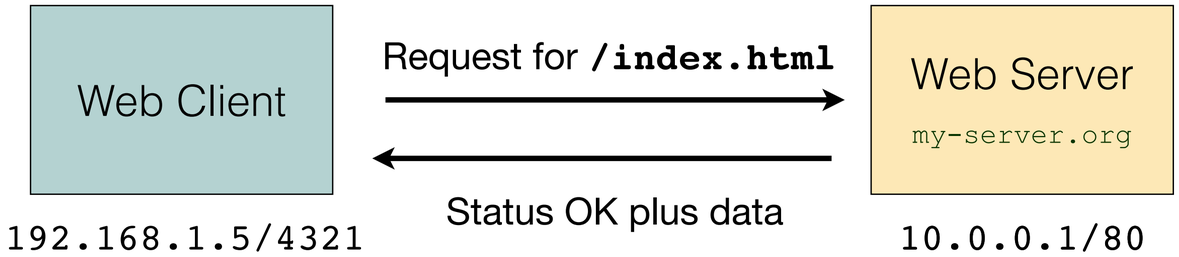
TCP stream reassembly for responder

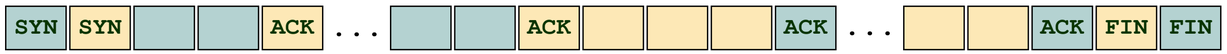
We know server has accepted, and we got the page content.

http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	GET
host	my-server.org
uri	/index.html
status_code	
status_msg	
…	

Behind the scenes: Processing HTTP





We know addresses and ports.

`GET /index.html HTTP/1.1 … Host: my-server.org …`

TCP stream reassembly for originator

We know what's being requested.

`200 OK … <HTML> …`

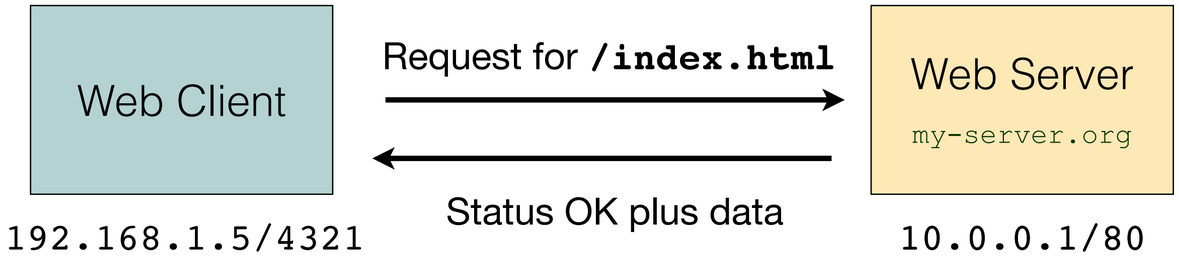
TCP stream reassembly for responder

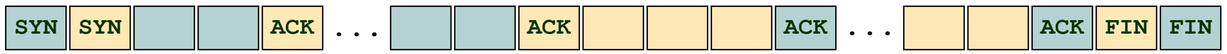
We know server has accepted, and we got the page content.

http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	GET
host	my-server.org
uri	/index.html
status_code	200
status_msg	OK
…	

Behind the scenes: Processing HTTP





We know addresses and ports.

GET /index.html HTTP/1.1 … Host: my-server.org …

TCP stream reassembly for originator

We know what's being requested.

200 OK … <HTML> …

TCP stream reassembly for responder

We know server has accepted, and we got the page content.

We know the connection is done.

http.log | HTTP request/reply details

Field	Value
id.orig_h	192.168.1.5
id.orig_p	10.0.0.1
id.resp_h	80/tcp
id.resp_p	4321/tcp
method	GET
host	my-server.org
uri	/index.html
status_code	200
status_msg	OK
…	

Zeek's C++ Code for HTTP Request Line

```
GET /index.html HTTP/1.1
```

Zeek's C++ Code for HTTP Request Line

```
int HTTP_Analyzer::HTTP_RequestLine(const char* line,
                                    const char* end_of_line) {
    const char* rest = nullptr;
    const char* end_of_method = get_HTTP_token(line, end_of_line);

    if ( end_of_method == line ) {
        // something went wrong with get_HTTP_token
        // perform a weak test to see if the string "HTTP/"
        // is found at the end of the RequestLine
        if ( end_of_line - 9 >= line &&
             strncasecmp(end_of_line - 9, " HTTP/", 6) == 0 )
            goto bad_http_request_with_version;

        goto error;
    }

    rest = util::skip_whitespace(end_of_method, end_of_line);

    if ( rest == end_of_method )
        goto error;

    if ( ! ParseRequest(rest, end_of_line) ) {
        reporter->AnalyzerError(this, "HTTP ParseRequest failed");
        return -1;
    }

    // If we determined HTTP/0.9, assert that minimally we have
    // an URI and a 3 character method. If that doesn't hold,
    //probably not HTTP or very strange.
    if ( request_version == HTTP_VersionNumber{0, 9} ) {
        bool maybe_get_method = (end_of_method - line) >= 3;
        bool has_uri = request_URI && request_URI->Len() > 0;

        if ( ! maybe_get_method || ! has_uri )
            goto error;
    }

    request_method = StringVal(end_of_method - line, line);
[…]
}
```

```
bool HTTP_Analyzer::ParseRequest(const char* line, const char* end_of_line){
    const char* end_of_uri, version_start, version_end;

    for ( end_of_uri = line; end_of_uri < end_of_line; ++end_of_uri ) {
        if ( ! is_reserved_URI_char(*end_of_uri)
            && ! is_unreserved_URI_char(*end_of_uri) && *end_of_uri != '%' )
            break;
    }

    […check for missing URI…]

    for ( version_start = end_of_uri; version_start < end_of_line; ++version_start ) {
        end_of_uri = version_start;
        version_start = util::skip_whitespace(version_start, end_of_line);
        if ( PrefixMatch(version_start, end_of_line, "HTTP/") )
            break;
    }

    if ( version_start >= end_of_line )
       SetVersion(&request_version, {0, 9}); // If no version is found
    else {
        if ( version_start + 8 <= end_of_line ) {
            version_start += 5; // "HTTP/"
            SetVersion(&request_version,
                       HTTP_Version(end_of_line - version_start, version_start));
            […check for crud after version…]
        } else { […report bad HTTP version…] }
    }

    request_URI = make_intrusive<StringVal>(end_of_uri - line, line);
[…]}

HTTP_VersionNumber HTTP_Analyzer::HTTP_Version(int len, const char* data) {
    if ( len >= 3 && data[0] >= '0' && data[0] <= '9' &&
                     data[1] == '.' && data[2] >= '0' && data[2] <= '9' ) {
        uint8_t major = data[0] - '0';
        uint8_t minor = data[2] - '0';
        return {major, minor};
    }
    else { […report bad HTTP version…] }
}
```

```
GET /index.html HTTP/1.1
```

Every single parser is a challenge to write

Every single parser is a challenge to write

Efficient

100.000s of connections

Incremental, stateful

Low latency, little memory

Every single parser is a challenge to write

Efficient

100.000s of connections

Incremental, stateful

Low latency, little memory

Complete

Protocols can be complex

Protocols lack specs

Hard to get traffic

Every single parser is a challenge to write

Efficient

100.000s of connections

Incremental, stateful

Low latency, little memory

Complete

Protocols can be complex

Protocols lack specs

Hard to get traffic

Robust & Safe

Cannot trust input data

Lots of “crud” in traffic

"Weird" Traffic

"Weird" Traffic

All "weirds" from 1hr of traffic at Berkeley Lab

DNS_Conn_count_too_large	line_terminated_with_single_CR	DNS_truncated_ans_too_short	base64_illegal_encoding
baroque_SYN	bad_UDP_checksum	FIN_advanced_last_seq	ayiya_empty_packet
TCP_christmas	bad_HTTP_request	missing_HTTP_entity	unknown_HTTP_method
active_connection_reuse	bad_ICMP_checksum	fragment_with_DF	unknown_gre_version_in_tunnel
data_before_established	bad_SYN_ack	geneve_invalid_version	truncated_GRE
possible_split_routing	Teredo_bubble_with_payload	HTTP_bad_chunk_size	truncated_ethernet_frame_in_tunnel
SYN_seq_jump	NUL_in_line	HTTP_response_before_request	SSL_unclear_connection_direction
connection_originator_SYN_ack	excessive_data_without_further_acks	HTTP_excessive_pipelining	geneve_invalid_version_in_tunnel
bad_TCP_checksum	DNS_label_len_gt_pkt	inflate_failed	DNSSEC_RRSIG_unknown_ZoneSignAlgo
truncated_header	DNS_truncated_RR_rdlength_lt_len	SSL_excessive_alerts_in_record	DNSSEC_NSEC_bitmapLen0
DNS_truncated_len_lt_hdr_len	DNS_truncated_quest_too_short	DNS_RR_bad_length	unknown_gre_flags
DNS_zero_rdlength	excessively_small_fragment	asn_binary_to_int64_shift_too_large	repeated_SYN_with_ack
TCP_seq_underflow_or_misorder	dnp3_corrupt_header_checksum	DNS_RR_length_mismatch	irc_too_many_invalid
window_recision	SYN_after_reset	unknown_gre_version	irc_line_too_short
max_find_all_string_length_exceeded	smtp_mail_transaction_invalid	ssl_early_application_data	irc_line_size_exceeded
SYN_with_data	DNS_RR_unknown_type	dnp3_header_lacks_magic	irc_invalid_reply_number
unknown_SIP_method	empty_http_request	HTTP_version_mismatch	irc_invalid_command
TCP_ack_underflow_or_misorder	RST_storm	truncated_IP_in_tunnel	FTP_invalid_command
data_after_reset	SYN_after_close	line_terminated_without_CRLF	DNSSEC_RRSIG_PrivateOID_ZoneSignAlgo
SYN_inside_connection	DNS_label_too_long	DNS_NAME_too_long	DNSSEC_DNSKEY_unknown_ZoneSignAlgo
above_hole_data_without_any_acks	DNS_label_forward_compress_offset	unknown_ip_version_in_tunnel	DNSSEC_DNSKEY_Invalid_Flag
premature_connection_reuse	QUIC_many_protocols	egre_protocol_type	DNS_label_len_gt_name_len
inappropriate_FIN	QUIC_max_history_length_reached	double_%_in_URI	

Robust & Safe?

Robust & Safe?

Zeek 6.0.4

- A **crash with ICMP** packets involving errant length checking was fixed.

Zeek 6.0.3

- A specially-crafted series of packets containing nested MIME entities can cause Zeek to spend **large amounts of time parsing** the entities.

Zeek 6.0.2

- A specially-crafted **SSL packet** could cause Zeek to leak memory and potentially **crash**.

- A specially-crafted **IEEE802.11 packet** could cause Zeek to overflow memory and potentially **crash**.

Zeek 6.0.1

- A specially-crafted **HTTP packet** can cause Zeek's filename extraction code to take a **long time to process** the data.

- A specially-crafted series of **FTP packets** made up of a CWD request followed by a large amount of ERPT requests may cause Zeek to spend **a long time logging** the commands.

- A specially-crafted **VLAN packet** can cause Zeek to overflow memory and potentially **crash**.

Four of six 6.0.x updates fixed analyzer issues

Some of these have been in the code forever

Parser vulnerabilities are endemic in this space

Parser vulnerabilities are endemic in this space

Year	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	File Inclusion	CSRF	XXE	SSRF	Open Redirect	Input Validation
2014	16	4	0	0	0	0	0	0	0	0	3
2015	5	3	0	0	0	0	0	0	0	0	14
2016	32	9	0	0	0	0	0	0	0	0	53
2017	5	3	0	0	0	0	0	0	0	0	15
2018	10	6	0	0	0	0	0	0	0	0	10
2019	2	3	0	0	0	0	0	0	0	0	2
2020	1	2	0	0	0	0	0	0	0	0	1
2021	3	0	0	0	0	0	0	0	0	0	0
2022	1	2	0	0	0	0	0	0	0	0	0
2023	3	8	0	0	0	0	0	0	0	0	0
2024	1	0	0	0	0	0	0	0	0	0	0
Total	79	40									98

https://www.cvedetails.com/product/8292/Wireshark-Wireshark.html?vendor_id=4861

Every single parser is a challenge to write

Efficient

100.000s of connections

Incremental, stateful

Low latency, little memory

Complete

Protocols can be complex

Protocols lack specs

Hard to get traffic

Robust & Safe

Cannot trust input data

Lots of “crud” in traffic

Every single parser is a challenge to write

Efficient

100.000s of connections

Incremental, stateful

Low latency, little memory

Complete

Protocols can be complex

Protocols lack specs

Hard to get traffic

Robust & Safe

Cannot trust input data

Lots of “crud” in traffic

Can we make make it easier to develop all these parsers?

Lowering the bar for writing robust parsers

High-Level Abstractions

Safe Execution

Reusability

Towards Better Languages

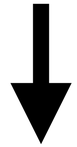
Projects have developed a variety of approaches to support writing parsers.

Towards Better Languages

Projects have developed a variety of approaches to support writing parsers.

Towards Better Languages

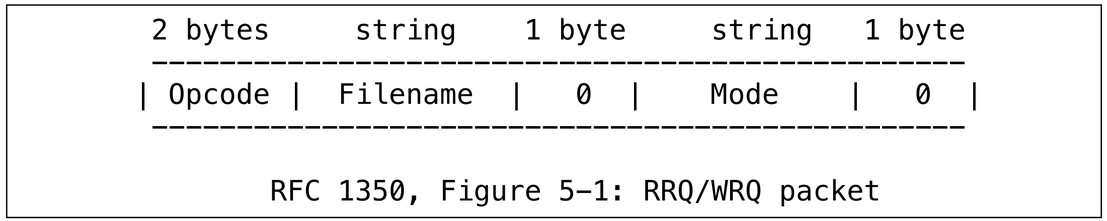
Projects have developed a variety of approaches to support writing parsers.



Safe? Yes. Abstractions? No.

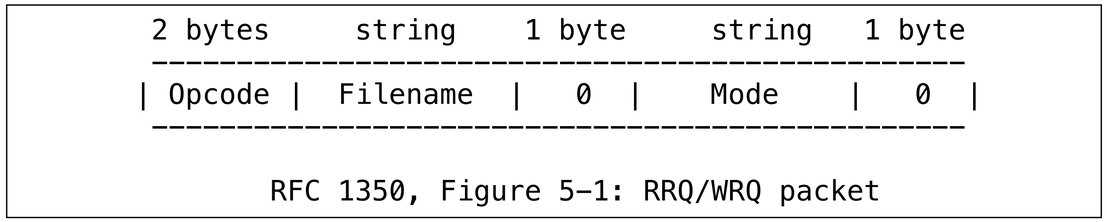
Example Rust In Suricata: TFTP

Example Rust In Suricata: TFTP



RFC 1350, Figure 5-1: RRQ/WRQ packet

Example Rust In Suricata: TFTP



RFC 1350, Figure 5-1: RRQ/WRQ packet

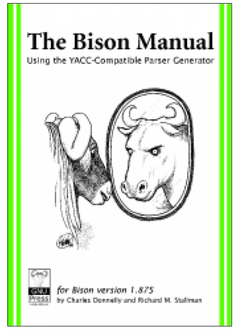
```
fn tftp_request(slice: &[u8]) -> IResult<&[u8], TFTPTransaction> {
    let (i, _) = tag([0])(slice)?;
    let (i, opcode) = be_u8(i)?;
    let (i, filename) = getstr(i)?;
    let (i, _) = tag([0])(i)?;
    let (i, mode) = getstr(i)?;
    Ok((i,
        TFTPTransaction::new(opcode, String::from(filename), String::from(mode))
       )
      )
}

fn parse_tftp_request(input: &[u8]) -> Option<TFTPTransaction> {
    match tftp_request(input) {
        Ok((_, tx)) => {
            if !tx.is_mode_ok() {
                return None;
            }
            if !tx.is_opcode_ok() {
                return None;
            }
            return Some(tx);
        }
        Err(_) => { return None; }
    }
}
```

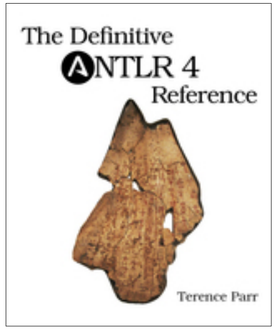
https://github.com/OISF/suricata/blob/master/rust/src/tftp/tftp.rs

Meanwhile, in another domain …

Meanwhile, in another domain …

Powerful tools for generating parsers from declarative specifications

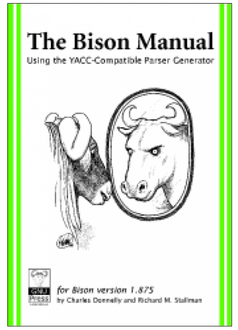


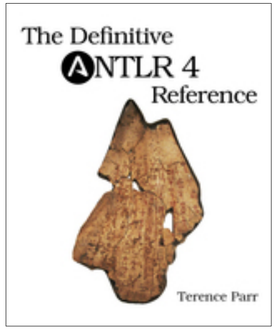


Tree-sitter

Meanwhile, in another domain …

Powerful tools for generating parsers from declarative specifications



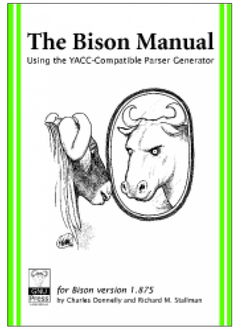


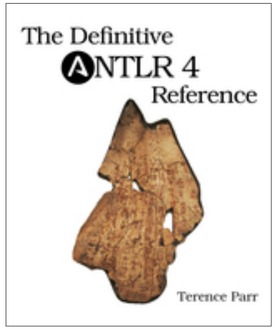
Tree-sitter

```
exp: NUM               { $$ = $1;         }
   | exp '+' exp       { $$ = $1 + $2;    }
   | exp '-' exp       { $$ = $1 - $2;    }
   | exp '*' exp       { $$ = $1 * $2;    }
   | exp '/' exp       { $$ = $1 / $2;    }
```

Meanwhile, in another domain …

Powerful tools for generating parsers from declarative specifications





Tree-sitter

These parsers aren’t suitable for protocols, unfortunately.

No concurrent, incremental processing

No domain-specific idioms

binpac: A yacc for Writing Application Protocol Parsers

Ruoming Pang*
Google, Inc.

Vern Paxson
International Computer Science Institute

Robin Sommer
International Computer Science Institute

Larry Peterson
Princeton University

IMC, 2006

binpac: A yacc for Writing Application Protocol Parsers

Ruoming Pang
Google, Inc.

Vern Paxson
International Computer Science Institute

Robin Sommer
International Computer Science Institute

Larry Peterson
Princeton University

IMC, 2006

TLS v3 Client Hello

```
type ClientHello(rec: HandshakeRecord) = record {
    client_version: uint16;
    gmt_unix_time : uint32;
    random_bytes  : bytestring &length = 28;
    session_len   : uint8;
    session_id    : uint8[session_len];
    dtls_cookie   : case client_version of {
        DTLSv10, DTLSv12 -> cookie : ClientHelloCookie(rec);
        default          -> nothing: bytestring &length=0;
    };
[…]}
```

Source: Zeek's TLS analyzer)

binpac

class binpac::
ConnectionAnalyzer

(C++)

Zeek

binpac: A yacc for Writing Application Protocol Parsers

Ruoming Pang
Google, Inc.

Vern Paxson
International Computer Science Institute

Robin Sommer
International Computer Science Institute

Larry Peterson
Princeton University

IMC, 2006

This solves our task only partially

Remains limited to syntax, cannot express logic (→ back to C++)

Only supports application-layer protocols

Lacks support for higher-level idioms

Zeek-only, no portability

New Tool: The Spicy Parser Generator

New Tool: The Spicy Parser Generator

Complete scripting language for parsing untrusted input

Protocols, file formats, text, binary — anything, really

New Tool: The Spicy Parser Generator

Complete scripting language for parsing untrusted input

Protocols, file formats, text, binary — anything, really

Declarative, with imperative hooks

Syntax and semantics inside a single model

New Tool: The Spicy Parser Generator

Complete scripting language for parsing untrusted input

Protocols, file formats, text, binary — anything, really

Declarative, with imperative hooks

Syntax and semantics inside a single model

Built for the domain

Types, byte order, layering, reassembly, error recovery, performance

New Tool: The Spicy Parser Generator

Complete scripting language for parsing untrusted input

Protocols, file formats, text, binary — anything, really

Declarative, with imperative hooks

Syntax and semantics inside a single model

Built for the domain

Types, byte order, layering, reassembly, error recovery, performance

API for embedding into host applications

Generates C++ parsing code for any host application to execute

Spicy Code for an HTTP Request Line

```
GET /index.html HTTP/1.1
```

Spicy Code for an HTTP Request Line

```
module HTTP;

const Token      = /[^ \t\r\n]+/;
const WhiteSpace = /[ \t]+/;
const NewLine    = /\r?\n/;

public type RequestLine = unit {
    method:  Token;
    :        WhiteSpace;
    uri:     Token;
    :        WhiteSpace;
    version: Version;
    :        NewLine;

    on %done { print self; }
};

type Version = unit {
    :       b"HTTP/";
    number: /[0-9]+\.[0-9]+/;
};
```

http-request.spicy

```
GET /index.html HTTP/1.1
```

Spicy Code for an HTTP Request Line

```
module HTTP;

const Token      = /[^ \t\r\n]+/;
const WhiteSpace = /[ \t]+/;
const NewLine    = /\r?\n/;

public type RequestLine = unit {
    method:  Token;
    :        WhiteSpace;
    uri:     Token;
    :        WhiteSpace;
    version: Version;
    :        NewLine;

    on %done { print self; }
};

type Version = unit {
    :       b"HTTP/";
    number: /[0-9]+\.[0-9]+/;
};
```

Logic!

http-request.spicy

```
GET /index.html HTTP/1.1
```

Spicy Code for an HTTP Request Line

```
module HTTP;

const Token      = /[^ \t\r\n]+/;
const WhiteSpace = /[ \t]+/;
const NewLine    = /\r?\n/;

public type RequestLine = unit {
    method:  Token;
    :        WhiteSpace;
    uri:     Token;
    :        WhiteSpace;
    version: Version;
    :        NewLine;

    on %done { print self; }
};

type Version = unit {
    :       b"HTTP/";
    number: /[0-9]+\.[0-9]+/;
};
```

http-request.spicy

```
# echo "GET /index.html HTTP/1.0" | spicy-driver http-request.spicy
```

Spicy Code for an HTTP Request Line

```
module HTTP;

const Token      = /[^ \t\r\n]+/;
const WhiteSpace = /[ \t]+/;
const NewLine    = /\r?\n/;

public type RequestLine = unit {
    method:  Token;
    :        WhiteSpace;
    uri:     Token;
    :        WhiteSpace;
    version: Version;
    :        NewLine;

    on %done { print self; }
};

type Version = unit {
    :       b"HTTP/";
    number: /[0-9]+\.[0-9]+/;
};
```

http-request.spicy

```
# echo "GET /index.html HTTP/1.0" | spicy-driver http-request.spicy
[$method=b"GET", $uri=b"/index.html", $version=[$number=b"1.0"]]
```

Spicy Code for an HTTP Request Line

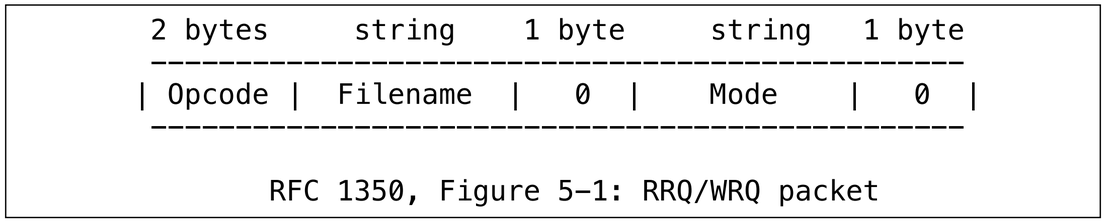
```
module HTTP;

const Token      = /[^ \t\r\n]+/;
const WhiteSpace = /[ \t]+/;
const NewLine    = /\r?\n/;

public type RequestLine = unit {
    method:  Token;
    :        WhiteSpace;
    uri:     Token;
    :        WhiteSpace;
    version: Version;
    :        NewLine;

    on %done { print self; }
};

type Version = unit {
    :       b"HTTP/";
    number: /[0-9]+\.[0-9]+/;
};
```

http-request.spicy

No Zeek!

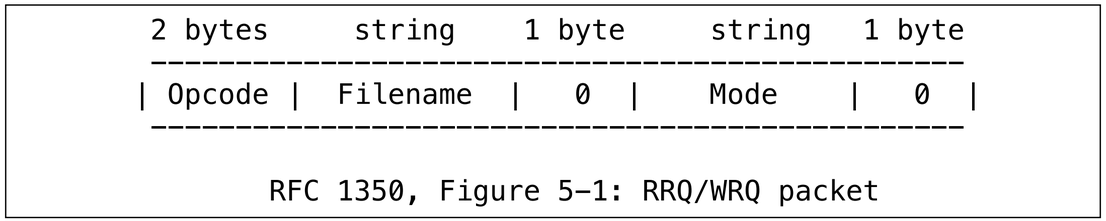
```
# echo "GET /index.html HTTP/1.0" | spicy-driver http-request.spicy
[$method=b"GET", $uri=b"/index.html", $version=[$number=b"1.0"]]
```

TFTP



RFC 1350, Figure 5-1: RRQ/WRQ packet

TFTP

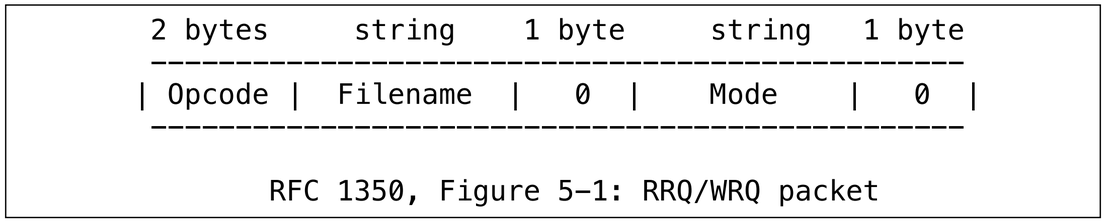


RFC 1350, Figure 5-1: RRQ/WRQ packet

```
module TFTP;

public type ReadRequest = unit {
    opcode:   uint16(0x01);
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};
```

tftp.spicy

TFTP



RFC 1350, Figure 5-1: RRQ/WRQ packet

```
module TFTP;

public type ReadRequest = unit {
    opcode:   uint16(0x01);
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};
```

tftp.spicy

```
# cat rrq.dat | spicy-dump --json tftp.spicy
{"opcode":1, "filename":"rfc1350.txt", "mode":"octet"}
```

A complete TFTP Parser per RFC 1350

A complete TFTP Parser per RFC 1350

```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
        Opcode::RRQ   -> rrq:   Request(True);
        Opcode::WRQ   -> wrq:   Request(False);
        Opcode::DATA  -> data:  Data;
        Opcode::ACK   -> ack:   Ack;
        Opcode::ERROR -> error: Error;
        };
};

type Opcode = enum { RRQ = 1, WRQ = 2, DATA = 3,
                     ACK = 4, ERROR = 5 };

type Request = unit(is_read: bool) {
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};

type Data = unit {
    num:  uint16;
    data: bytes &eod;
};

type Ack = unit {
    num: uint16;
};

type Error = unit {
    code: uint16;
    msg:  bytes &until=b"\x00";
};
```

tftp.spicy

35 LOC

A complete TFTP Parser per RFC 1350

```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
        Opcode::RRQ   -> rrq:   Request(True);
        Opcode::WRQ   -> wrq:   Request(False);
        Opcode::DATA  -> data:  Data;
        Opcode::ACK   -> ack:   Ack;
        Opcode::ERROR -> error: Error;
        };
};

type Opcode = enum { RRQ = 1, WRQ = 2, DATA = 3,
                     ACK = 4, ERROR = 5 };

type Request = unit(is_read: bool) {
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};

type Data = unit {
    num:  uint16;
    data: bytes &eod;
};

type Ack = unit {
    num: uint16;
};

type Error = unit {
    code: uint16;
    msg:  bytes &until=b"\x00";
};
```

tftp.spicy

35 LOC

```
protocol analyzer spicy::TFTP over UDP:
    parse with TFTP::Packet,
    port 69/udp;

on TFTP::Request if ( is_read )
    -> event tftp::read_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Request if ( ! is_read )
    -> event tftp::write_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Data  -> event tftp::data($conn, $is_orig, self.num, self.data);
on TFTP::Ack   -> event tftp::ack($conn, $is_orig, self.num);
on TFTP::Error -> event tftp::error($conn, $is_orig, self.code, self.msg);
```

tftp.evt

A complete TFTP Parser per RFC 1350

```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
        Opcode::RRQ   -> rrq:   Request(True);
        Opcode::WRQ   -> wrq:   Request(False);
        Opcode::DATA  -> data:  Data;
        Opcode::ACK   -> ack:   Ack;
        Opcode::ERROR -> error: Error;
        };
};

type Opcode = enum { RRQ = 1, WRQ = 2, DATA = 3,
                     ACK = 4, ERROR = 5 };

type Request = unit(is_read: bool) {
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};

type Data = unit {
    num:  uint16;
    data: bytes &eod;
};

type Ack = unit {
    num: uint16;
};

type Error = unit {
    code: uint16;
    msg:  bytes &until=b"\x00";
};
```

tftp.spicy

35 LOC

```
protocol analyzer spicy::TFTP over UDP:
    parse with TFTP::Packet,
    port 69/udp;

on TFTP::Request if ( is_read )
    -> event tftp::read_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Request if ( ! is_read )
    -> event tftp::write_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Data  -> event tftp::data($conn, $is_orig, self.num, self.data);
on TFTP::Ack   -> event tftp::ack($conn, $is_orig, self.num);
on TFTP::Error -> event tftp::error($conn, $is_orig, self.code, self.msg);
```

tftp.evt

```
event tftp::read_request(c: connection, fname: string, mode: string) {
    print "[In Zeek] TFTP read request", c$id, fname, mode;
}
```

tftp.zeek

A complete TFTP Parser per RFC 1350

```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
        Opcode::RRQ   -> rrq:   Request(True);
        Opcode::WRQ   -> wrq:   Request(False);
        Opcode::DATA  -> data:  Data;
        Opcode::ACK   -> ack:   Ack;
        Opcode::ERROR -> error: Error;
        };
};

type Opcode = enum { RRQ = 1, WRQ = 2, DATA = 3,
                     ACK = 4, ERROR = 5 };

type Request = unit(is_read: bool) {
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};

type Data = unit {
    num:  uint16;
    data: bytes &eod;
};

type Ack = unit {
    num: uint16;
};

type Error = unit {
    code: uint16;
    msg:  bytes &until=b"\x00";
};
```

tftp.spicy

35 LOC

```
protocol analyzer spicy::TFTP over UDP:
    parse with TFTP::Packet,
    port 69/udp;

on TFTP::Request if ( is_read )
    -> event tftp::read_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Request if ( ! is_read )
    -> event tftp::write_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Data  -> event tftp::data($conn, $is_orig, self.num, self.data);
on TFTP::Ack   -> event tftp::ack($conn, $is_orig, self.num);
on TFTP::Error -> event tftp::error($conn, $is_orig, self.code, self.msg);
```

tftp.evt

```
event tftp::read_request(c: connection, fname: string, mode: string) {
    print "[In Zeek] TFTP read request", c$id, fname, mode;
}
```

tftp.zeek

```
# spicyz -o tftp.hlto tftp.spicy tftp.evt
```

A complete TFTP Parser per RFC 1350

```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
        Opcode::RRQ   -> rrq:   Request(True);
        Opcode::WRQ   -> wrq:   Request(False);
        Opcode::DATA  -> data:  Data;
        Opcode::ACK   -> ack:   Ack;
        Opcode::ERROR -> error: Error;
        };
};

type Opcode = enum { RRQ = 1, WRQ = 2, DATA = 3,
                     ACK = 4, ERROR = 5 };

type Request = unit(is_read: bool) {
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};

type Data = unit {
    num:  uint16;
    data: bytes &eod;
};

type Ack = unit {
    num: uint16;
};

type Error = unit {
    code: uint16;
    msg:  bytes &until=b"\x00";
};
```

tftp.spicy

35 LOC

```
protocol analyzer spicy::TFTP over UDP:
    parse with TFTP::Packet,
    port 69/udp;

on TFTP::Request if ( is_read )
    -> event tftp::read_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Request if ( ! is_read )
    -> event tftp::write_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Data  -> event tftp::data($conn, $is_orig, self.num, self.data);
on TFTP::Ack   -> event tftp::ack($conn, $is_orig, self.num);
on TFTP::Error -> event tftp::error($conn, $is_orig, self.code, self.msg);
```

tftp.evt

```
event tftp::read_request(c: connection, fname: string, mode: string) {
    print "[In Zeek] TFTP read request", c$id, fname, mode;
}
```

tftp.zeek

```
# spicyz -o tftp.hlto tftp.spicy tftp.evt

# zeek -r tftp_rrq.pcap tftp.hlto tftp.zeek
```

A complete TFTP Parser per RFC 1350

```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
        Opcode::RRQ   -> rrq:   Request(True);
        Opcode::WRQ   -> wrq:   Request(False);
        Opcode::DATA  -> data:  Data;
        Opcode::ACK   -> ack:   Ack;
        Opcode::ERROR -> error: Error;
        };
};

type Opcode = enum { RRQ = 1, WRQ = 2, DATA = 3,
                     ACK = 4, ERROR = 5 };

type Request = unit(is_read: bool) {
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};

type Data = unit {
    num:  uint16;
    data: bytes &eod;
};

type Ack = unit {
    num: uint16;
};

type Error = unit {
    code: uint16;
    msg:  bytes &until=b"\x00";
};
```

tftp.spicy

35 LOC

```
protocol analyzer spicy::TFTP over UDP:
    parse with TFTP::Packet,
    port 69/udp;

on TFTP::Request if ( is_read )
    -> event tftp::read_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Request if ( ! is_read )
    -> event tftp::write_request($conn, $is_orig, self.filename, self.mode);

on TFTP::Data  -> event tftp::data($conn, $is_orig, self.num, self.data);
on TFTP::Ack   -> event tftp::ack($conn, $is_orig, self.num);
on TFTP::Error -> event tftp::error($conn, $is_orig, self.code, self.msg);
```

tftp.evt

```
event tftp::read_request(c: connection, fname: string, mode: string) {
    print "[In Zeek] TFTP read request", c$id, fname, mode;
}
```

tftp.zeek

```
# spicyz -o tftp.hlto tftp.spicy tftp.evt

# zeek -r tftp_rrq.pcap tftp.hlto tftp.zeek

[In Zeek] TFTP read request, [orig_h=192.168.0.253,
orig_p=50618/udp, resp_h=192.168.0.10, resp_p=69/udp],
rfc1350.txt, octet
```

Spicy Features

Where’s Spicy at now?

Where’s Spicy at now?

Spicy has become the default for new analyzers

```
Finger
LDAP
QUIC
Syslog
TLS
WebSocket
```

Where’s Spicy at now?

Spicy has become the default for new analyzers

```
Finger
LDAP
QUIC
Syslog
TLS
WebSocket
```

Unblocked Zeek analyzer development!

Where's Spicy at now?

Spicy has become the default for new analyzers

```
Finger
LDAP
QUIC
Syslog
TLS
WebSocket
```

Spicy has become the default for new analyzers

Open source & proprietary

```
IPsec
OpenVPN
OSPF
Radius
Wireguard
```

Unblocked Zeek analyzer development!

Where's Spicy at now?

Spicy has become the default for new analyzers

```
Finger
LDAP
QUIC
Syslog
TLS
WebSocket
```

Unblocked Zeek analyzer development!

Spicy has become the default for new analyzers

Open source & proprietary

```
IPsec
OpenVPN
OSPF
Radius
Wireguard
```

Driver for many Spicy improvements!

Where's Spicy at now?

Spicy has become the default for new analyzers

```
Finger
LDAP
QUIC
Syslog
TLS
WebSocket
```

Unblocked Zeek analyzer development!

Spicy has become the default for new analyzers

Open source & proprietary

```
IPsec
OpenVPN
OSPF
Radius
Wireguard
```

Driver for many Spicy improvements!

Community seems to be switching as well

~20, maybe, Spicy analyzers

```
DHCPv4/v6
CCLink
IEC 104/60870/61850
HART-IP
Profinet
```

Where's Spicy at now?

Spicy has become the default for new analyzers

```
Finger
LDAP
QUIC
Syslog
TLS
WebSocket
```

Unblocked Zeek analyzer development!

Spicy has become the default for new analyzers

Open source & proprietary

```
IPsec
OpenVPN
OSPF
Radius
Wireguard
```

Driver for many Spicy improvements!

Community seems to be switching as well

~20, maybe, Spicy analyzers

```
DHCPv4/v6
CCLink
IEC 104/60870/61850
HART-IP
Profinet
```

Especially IoT and ICS

Where's Spicy at now?

zeek

Spicy has become the default for new analyzers

```
Finger
LDAP
QUIC
Syslog
TLS (soon)
WebSocket
```

Unblocked Zeek analyzer development!

Collection of ICS parsers

4/10 are in Spicy

GE SRTP

Genisys

Profinet I/O Context Manager

Synchrophasor Data Transfer for Power Systems (C37.118)

`https://github.com/cisagov/icsnpp`

Has been driver for many improvements!

GitHub

Community seems to be switching as well

), maybe, Spicy analyzers

```
DHCPv4/v6
CCLink
IEC 104/60870/61850
HART-IP
Profinet
```

Especially IoT and ICS

Where’s Spicy at now?

It works — it does lower the bar substantially

Where's Spicy at now?

It works — it does lower the bar substantially

People are writing analyzers who would have never before

Where's Spicy at now?

It works — it does lower the bar substantially

People are writing analyzers who would have never before

However, parsing a protocol is still not trivial

Requires programming skills at about the Python-level

Still needs learning a new language & idioms

Still needs protocol documentation & traffic(!)

Still needs Zeek-side logic

Where's Spicy at now?

It works — it does lower the bar substantially

People are writing analyzers who would have never before

However, parsing a protocol is still not trivial

But it's teachable now!

Where's Spicy at now?

It works — it does lower the bar substantially

People are writing analyzers who would have never before

However, parsing a protocol is still not trivial

But it's teachable now!

Emphasize onboarding Spicy developers

Good documentation, templates, best practices

Developer tooling (debugging support, editor integration, code formatting)

Training, self-directed and on site (via Corelight)

`github.com/zeek/spicy`

`docs.zeek.org/projects/spicy`

1. Installation
2. Getting Started
3. Frequently Asked Questions
4. Tutorial: A Real Analyzer
5. Programming in Spicy
6. Toolchain
7. Zeek Integration
8. Custom Host Applications

Looking for something to do?

Protocols are a plenty

`github.com/zeek/spicy`

`docs.zeek.org/projects/spicy`

1. Installation
2. Getting Started
3. Frequently Asked Questions
4. Tutorial: A Real Analyzer
5. Programming in Spicy
6. Toolchain
7. Zeek Integration
8. Custom Host Applications

Looking for something to do?

Protocols are a plenty

Home networks, anyone?

github.com/zeek/spicy

docs.zeek.org/projects/spicy

1. Installation
2. Getting Started
3. Frequently Asked Questions
4. Tutorial: A Real Analyzer
5. Programming in Spicy
6. Toolchain
7. Zeek Integration
8. Custom Host Applications

Looking for something to do?

Protocols are a plenty

Home networks, anyone?

File formats are unchartered territory

Wireshark, powered by Spicy?

Wireshark, powered by Spicy?

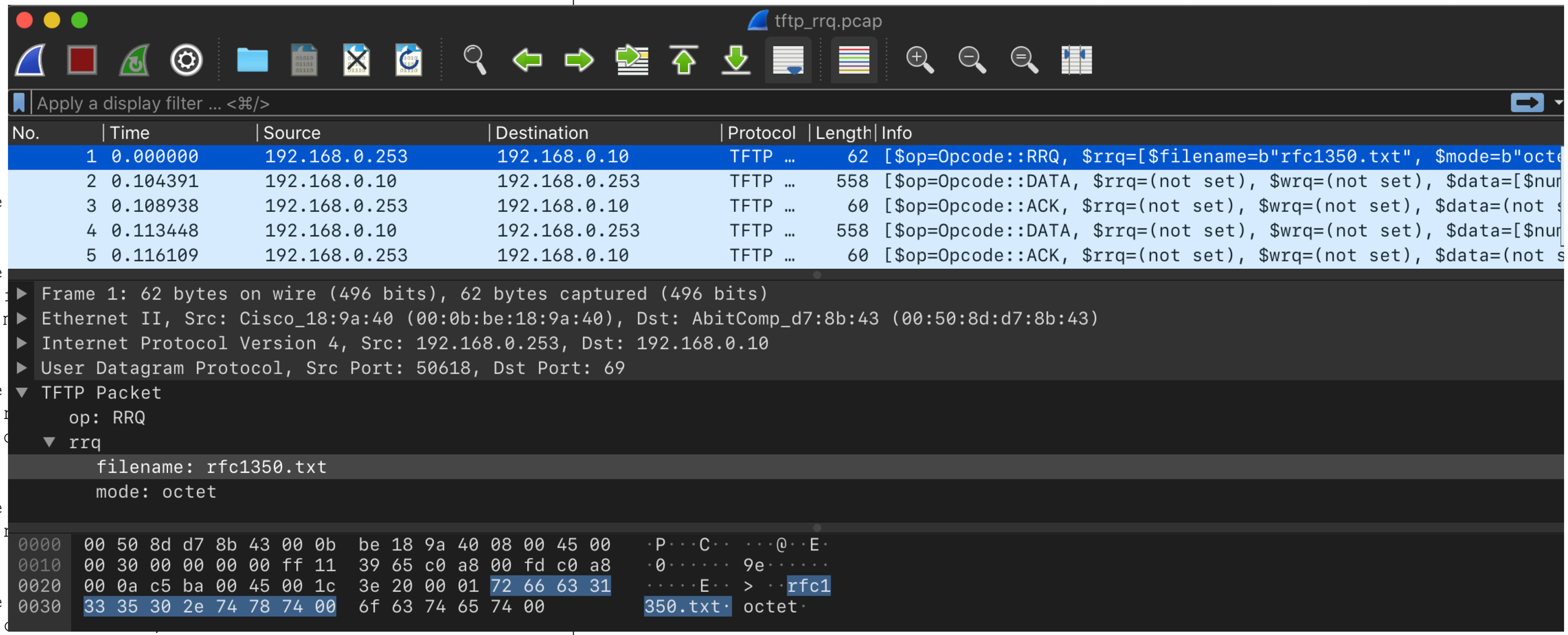
```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
        Opcode::RRQ   -> rrq:   Request(True);
        Opcode::WRQ   -> wrq:   Request(False);
        Opcode::DATA  -> data:  Data;
        Opcode::ACK   -> ack:   Acknowledgement;
        Opcode::ERROR -> error: Error;
        };
};

type Opcode = enum { RRQ = 1, WRQ = 2, DATA = 3,
                     ACK = 4, ERROR = 5 };

type Request = unit(is_read: bool) {
    filename: bytes &until=b"\x00";
    mode:     bytes &until=b"\x00";
};

type Data = unit {
    num:  uint16;
    data: bytes &eod;
};

type Acknowledgement = unit {
    num: uint16;
};

type Error = unit {
    code: uint16;
    msg:  bytes &until=b"\x00";
};
```

tftp.spicy

Wireshark, powered by Spicy?

```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
};

type
type
};
type
};
type
};
type
    msg:  bytes &until=b"\x00";
};
```

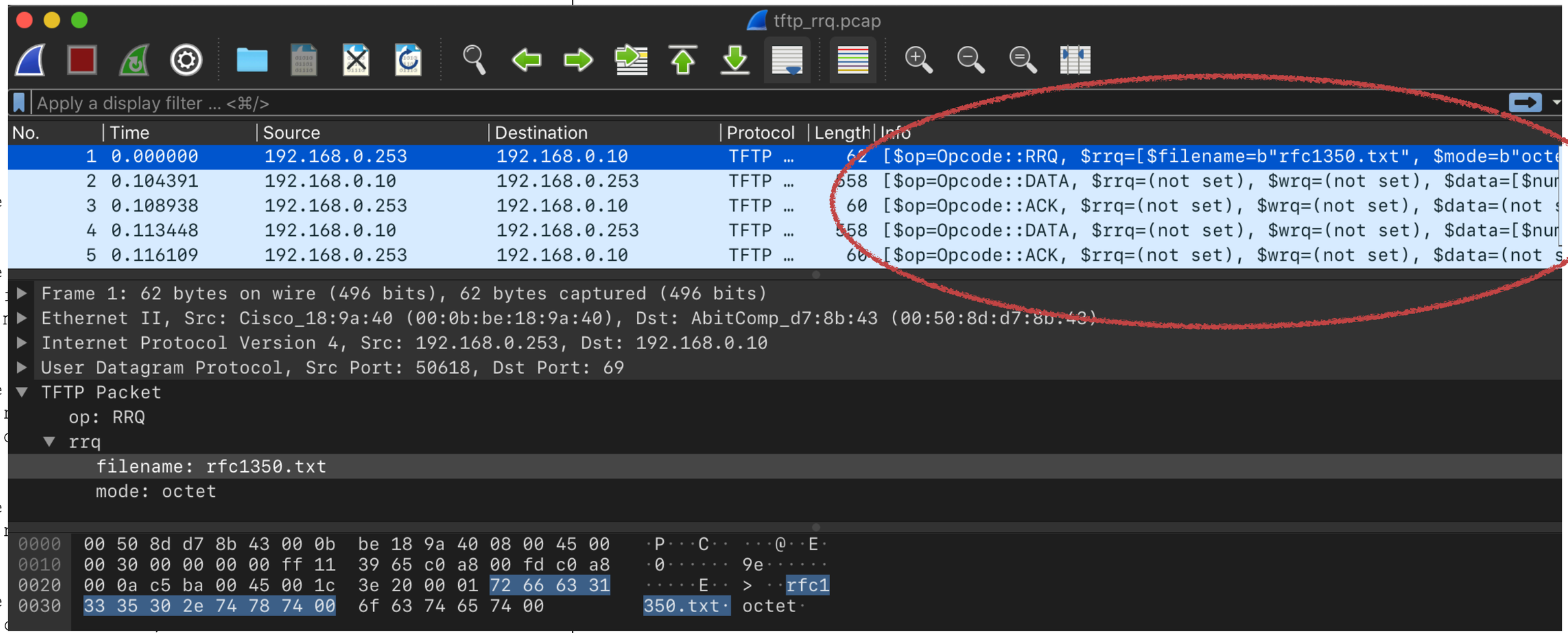
tftp.spicy



Wireshark, powered by Spicy?

```
module TFTP;

public type Packet = unit {
    op: uint16 &convert=Opcode($$);
    switch ( self.op ) {
};

type
type
};
type
};
type
};
type
    msg:  bytes &until=b"\x00";
};
```

tftp.spicy



Meet "Spicy"

You can write a protocol parser now!

Robin Sommer

robin@corelight.com

---

https://github.com/zeek/spicy

https://docs.zeek.org/projects/spicy

#spicy on the Zeek Slack

corelight

zeek