Zeek and Destroy with Python and Machine Learning

Workshop

Eva Szilagyi and David Szili

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Agenda for Today

- Introduction to Zeek
- Zeek Architecture
- Zeek Events and Logs
- Machine Learning on Zeek Logs
- Snakes! ☺ (Anaconda and Python)
- Zeek Scripting
- Zeek Broker and Machine Learning

Intermediate/Advanced Topics!

- We are going to discuss **intermediate/advanced topics!**
- The assumption is that you are *somewhat* familiar with **security monitoring and/or machine learning** concepts.
- Also; the **main focus is on Zeek**, and not on machine learning.
- **Finally; none of this is our own research! We are just connecting dots (see refs later)!**

Workshop Virtual Machine

- **ais_secmon_xubuntu-22.04.4-desktop-amd64**
- **VMware Workstation Pro, Player, or Fusion**
  - You can try VirtualBox or other virtualization too, but you are on your own… sorry!
- You will need:
  - **4 GB RAM**
  - **4 CPU cores/threads**
  - **50 GB disk space**
- Workshop VM (Ubuntu) user/pass: **user / Workshop1234%**
  - However, it should not require password for login and sudo.

About Eva

- Managing partner at **Alzette Information Security** (@AlzetteInfoSec)
- Network penetration testing, **web application penetration testing, source code review, security monitoring**
- **BSides Luxembourg** organizer https://bsideslux.lu
- **Twitch: https://www.twitch.tv/alzetteinfosec**
- **YouTube: https://www.youtube.com/@alzetteinfosec**
- **Twitter (X):** @EvaSzilagyiSec
- **E-mail: eva.szilagyi@alzetteinfosec.com**

About David

- Managing partner at **Alzette Information Security** (@AlzetteInfoSec)
- Network penetration testing, security architectures, **security monitoring, threat hunting, incident response**, **digital forensics**
- **Instructor at SANS** Institute: FOR572, FOR509
- **SANS Lead author:** SANS DFIR NetWars
- **BSides Luxembourg** Organizer: https://bsideslux.lu
- **Twitch: https://www.twitch.tv/alzetteinfosec**
- **YouTube: https://www.youtube.com/@alzetteinfosec**
- **Twitter (X):** @DavidSzili
- **E-mail: david.szili@alzetteinfosec.com**

Introduction to Zeek

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

About Zeek

What is Zeek?

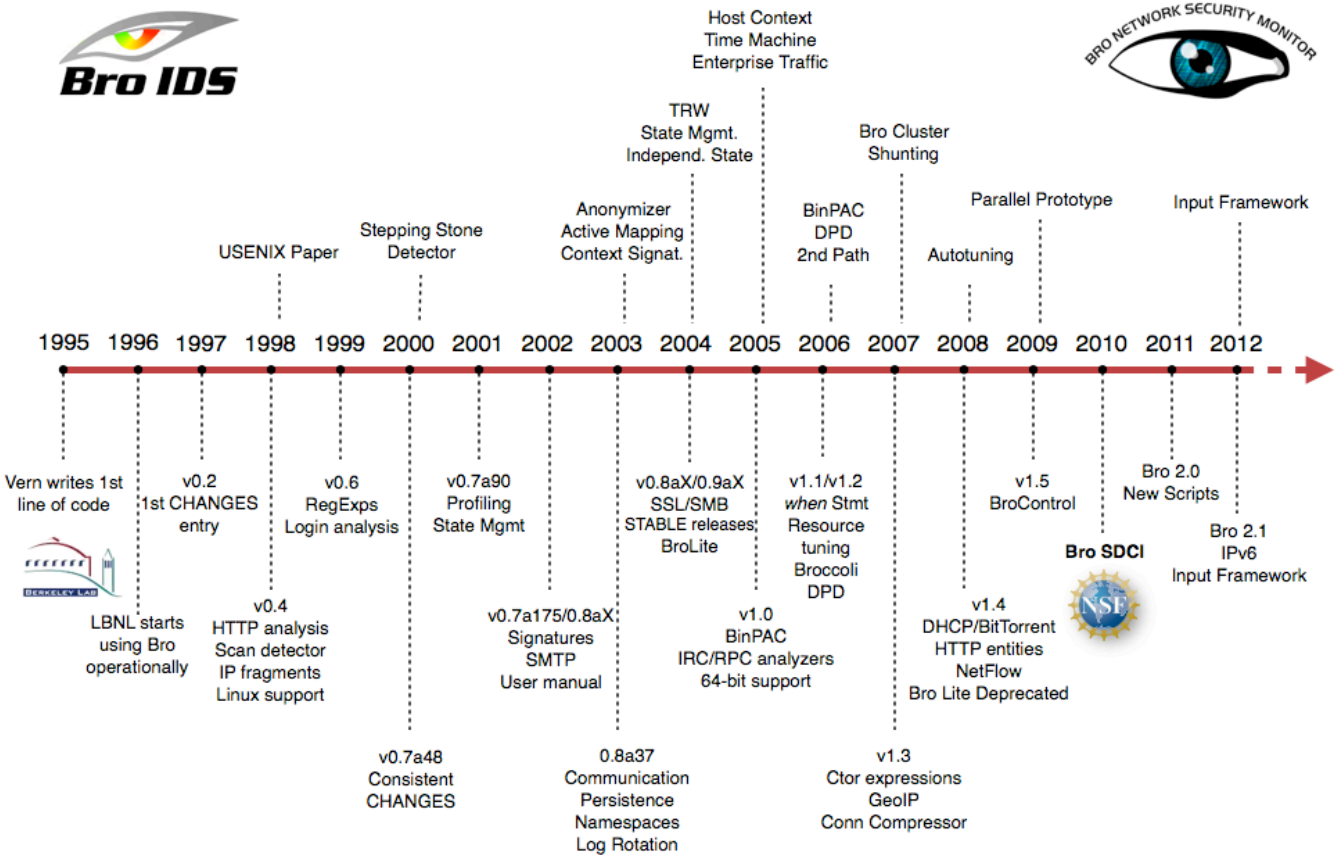
- Passive, **open-source** network traffic analyzer
- **Event/data-driven** NIDS/NSM
- Fully customizable and extensible **platform for traffic analysis**
- **Can run on commodity hardware** (up to 10GbE or even 100GbE links)
- Commercial offerings: **Corelight**

Why Zeek?

- **Network Intrusion Detection Systems (NIDS)**
  - Alert data only
- **Network Security Monitoring (NSM)**
  - Alert data
  - Flow (or Session) data
  - Transaction data
  - Packet data (PCAP)
  - Statistical data
  - Correlated data

Zeek's History

- **1995** – Initial version by Vern Paxson
- **1996** – Berkeley Lab deployment
- **2003** – National Science Foundation (NSF) began supporting Bro R&D
- **2010** – National Center for Supercomputing Applications (NCSA) joined the team as a core partner
- **2013** – NSF renewed its support
- **2014** – try.bro.org (now try.zeek.org)
- **2016** – Zeek package manager
- **2018** – Changing the name of the software from Bro to Zeek.
- **2020** – Spicy Parser Generator



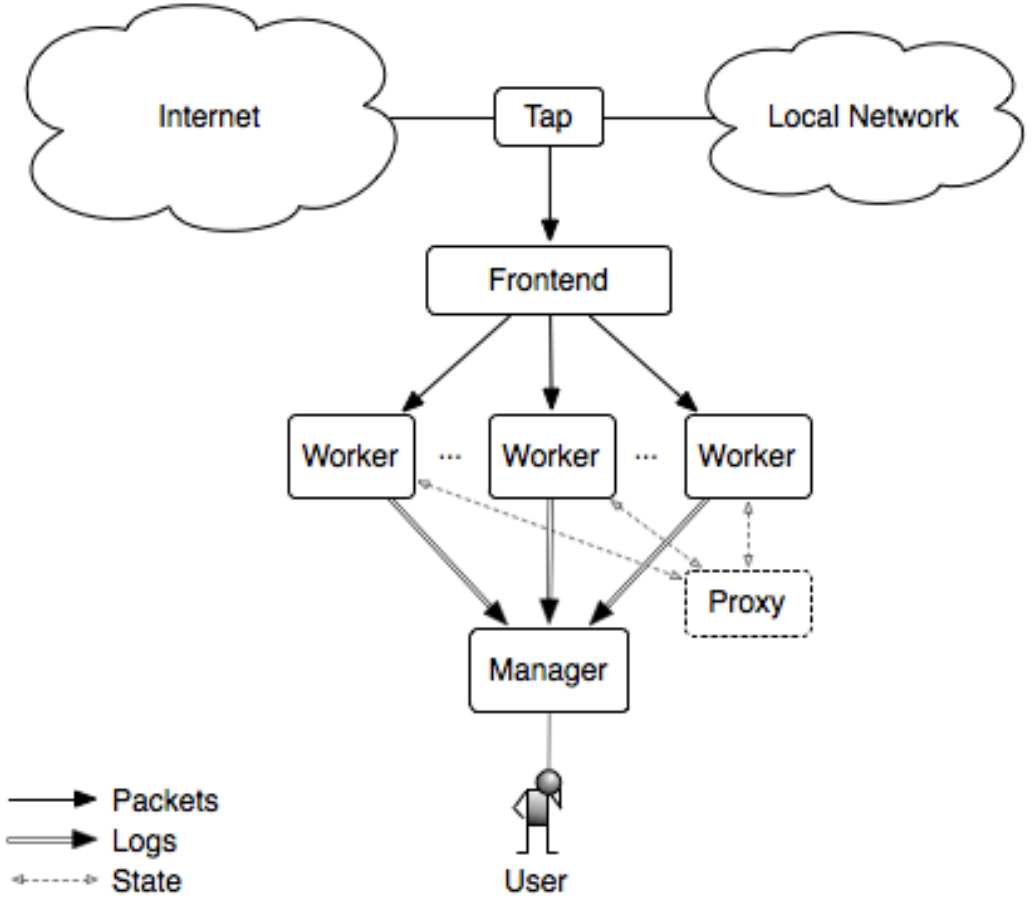
Zeek Architecture

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

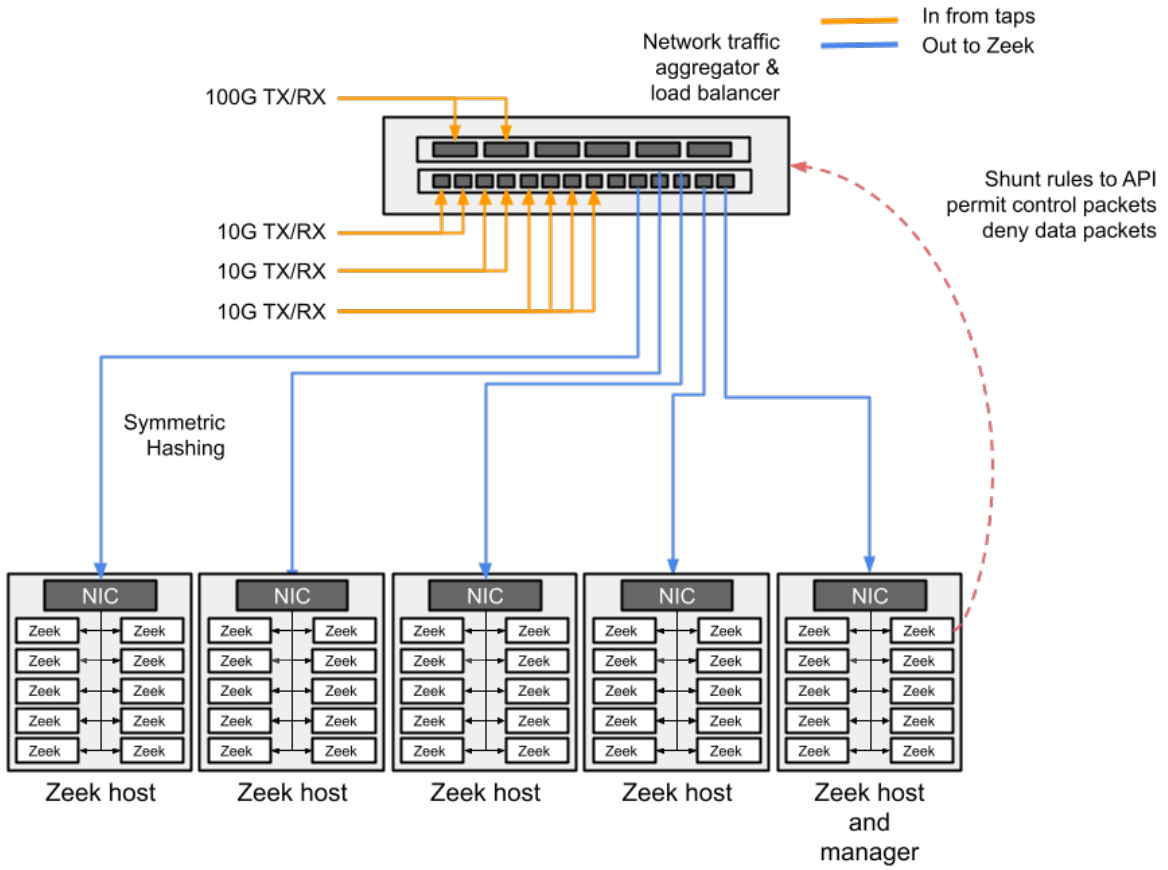
Zeek's Cluster Architecture (1)

- Standalone vs. cluster mode
- **Network Frontend**:
  - hardware flow balancers
  - on-host flow balancing (PF_RING)
- **Manager**: central log collector
- **Worker**: traffic inspection, stream reassembly, protocol analysis
- **Proxy**: synchronizing Zeek state
- **Logger** (optional): receives log messages from nodes

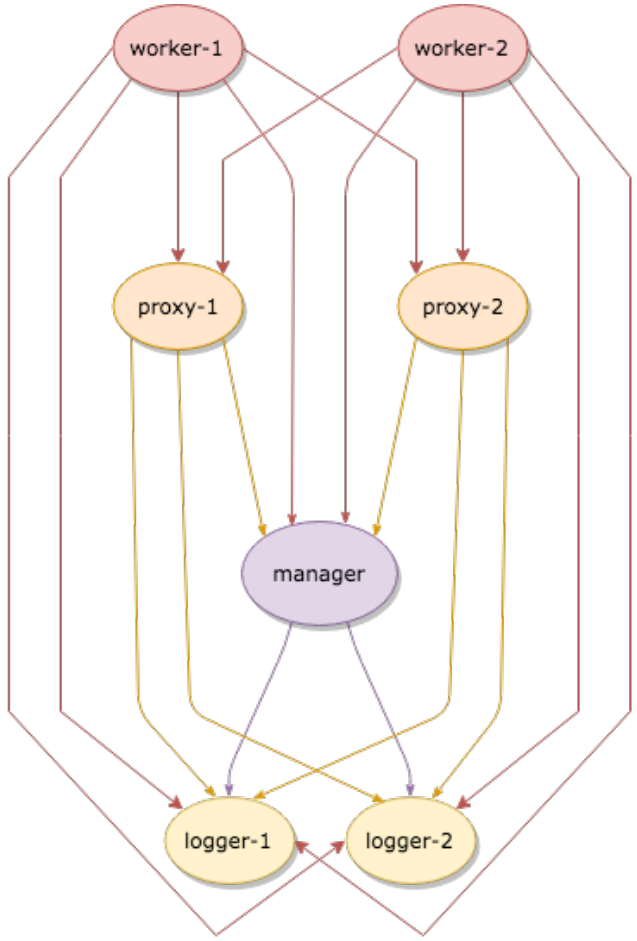


Source: https://docs.zeek.org/en/master/_images/deployment.png

Zeek's Cluster Architecture (2)



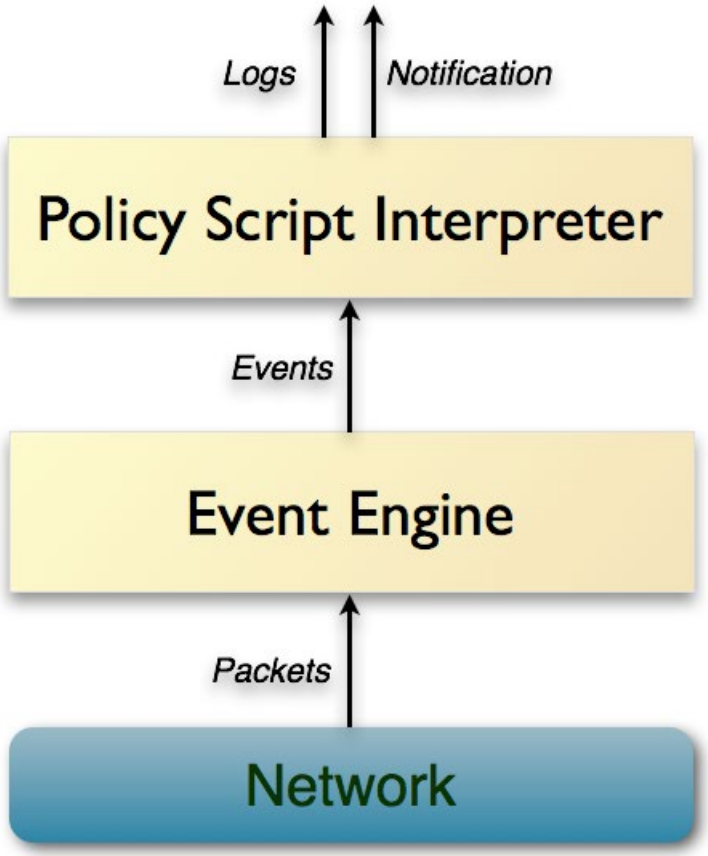
Source: https://docs.zeek.org/en/master/_images/cluster-diagram.png



Source: https://docs.zeek.org/en/master/_images/cluster-layout.png

Zeek's Internal Architecture

- **Event Engine**: runs protocol analyzers, generates network events
- **Policy Script Interpreter**: performs action/writes output



Source: https://docs.zeek.org/en/master/_images/architecture.png

Zeek Events

- **Zeek's Event Engine:**
  - Reduces the incoming packet stream into a series of higher-level events
  - Places events into an ordered "event queue"
- **Events:**
  - State change (new_connection, signature_match)
  - Protocol specific (http_response, dns_request)
  - Data availability (http_entity_data, **file_sniff**)
  - Etc.

Zeek Frameworks (1)

Framework	Description
Broker Communication Framework	**Exchange information with other Zeek processes**
Cluster Framework	The basic premise of Zeek clusterization
Configuration Framework	Allows updating script options dynamically at runtime
File Analysis Framework	**Generalized presentation of file-related information**
Input Framework	*Allows users to import data into Zeek*
Intelligence Framework	*Consume data and make it available for matching*
Logging Framework	**Fine-grained control of what and how is logged**
Management Framework	Provides a Zeek-based, service-oriented architecture
NetControl Framework	Flexible, unified interface for active response

Source: https://docs.zeek.org/en/master/frameworks/index.html

Zeek Frameworks (2)

Framework	Description
NetControl Framework	Flexible, unified interface for active response
Notice Framework	*Detect potentially interesting situations and take action*
Packet Analysis	*Handles parsing of packet headers at layers*
Summary Statistics Framework	Measuring aspects of network traffic
Signature Framework	Signature language for low-level pattern matching
Telemetry Framework	Can be used to record metrics
TLS Decryption	Limited support for decrypting TLS connections

Source: https://docs.zeek.org/en/master/frameworks/index.html

Zeek Directory Hierarchy

Directory	Content
$(PREFIX)/zeek/bin/	**Executables**: binpac, hiltic, spicyc, zeek, zeekctl, zeek-cut, zkg, etc.
$(PREFIX)/zeek/etc/	**Configuration**: networks.cfg, node.cfg, zeekctl.cfg, /zkg/config
$(PREFIX)/zeek/logs/	**Logs**: current (symlink to $(PREFIX)/zeek/spool/zeek), <date-hashed>
$(PREFIX)/zeek/spool/	Current logs, error logs: /tmp
$(PREFIX)/zeek/share/zeek/	/base: initialization (init-bare.zeek, init-default.zeek, etc.) **/site: extensions and local.zeek** /policy: tuning, protocol policies /zeekctl: scripts for zeekctl /zeekygen: documentation example
$(PREFIX)/zeek/lib/zeek/	**Plugins**: zeekctl, **broker**, zkg, etc.

Where $(PREFIX) will typically be either /usr/local/ (compiled) or /opt/ (binary package install).

Zeek Logs

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Zeek Logs (Just a Few Examples)

Log File	Description
conn.log	TCP/UDP/ICMP connections
dhcp.log	DHCP leases
dns.log	DNS activity
ftp.log	FTP activity
http.log	HTTP requests and replies
rdp.log	RDP
smb_cmd.log	SMB commands
smb_files.log	SMB files
ssh.log	SSH connections

Log File	Description
ssl.log	SSL/TLS handshake info
files.log	File analysis results
x509.log	X.509 certificate info
intel.log	Intelligence data matches
notice.log	Zeek notices
signatures.log	Signature matches
known_hosts.log	Hosts seen (TCP handshakes)
software.log	Software seen on the network
weird.log	Unexpected network activity

Complete list: https://docs.zeek.org/en/master/script-reference/log-files.html
Details: https://docs.zeek.org/en/master/logs/index.html

Using zeek-cut

- The default Zeek log format is **TSV**
- The **zeek-cut** utility can be used to build terminal commands
- The tool parses the header in each file, allowing you to refer specific columns

```
$ cat conn.log | zeek-cut id.orig_h id.orig_p id.resp_h id.resp_p
192.168.1.102       68          192.168.1.1         67
192.168.1.103       137         192.168.1.255       137
192.168.1.102       137         192.168.1.255       137
192.168.1.103       138         192.168.1.255       138
192.168.1.102       138         192.168.1.255       138
192.168.1.104       137         192.168.1.255       137
192.168.1.104       138         192.168.1.255       138
192.168.1.103       68          192.168.1.1         67
192.168.1.102       138         192.168.1.255       138
192.168.1.104       68          192.168.1.1         67
192.168.1.102       1170        192.168.1.1         53
192.168.1.104       1174        192.168.1.1         53
192.168.1.1         5353        224.0.0.251         5353
fe80::219:e3ff:fee7:5d23        5353        ff02::fb    5353
192.168.1.103       137         192.168.1.255       137
```

UNIX Epoch vs. ISO8601

- zeek-cut has the flag **-d** to convert the epoch time values in the log files to a human-readable format:

```
$ cat http.log | zeek-cut -d ts uid host
2009-11-18T10:14:13+0100      CmBOWT297WuJIENdwl     download.windowsupdate.com
```

- Converting the timestamp from a log file to UTC can be accomplished with the **-u** option:

```
$ cat http.log | zeek-cut -u ts uid host
2009-11-18T09:14:13+0000      CmBOWT297WuJIENdwl     download.windowsupdate.com
```

- The format can be altered using the **-D and -U flags**, and strftime syntax:

```
$ cat http.log | zeek-cut -D %d-%m-%YT%H:%M:%S%z ts uid host
18-11-2009T10:14:13+0100      CmBOWT297WuJIENdwl     download.windowsupdate.com
```

UIDs, and FUIDs (and Community-ID)

- **Unique IDentifier (UID)**: correlating a session across multiple Zeek log files

```
$ cat conn.log | zeek-cut uid id.resp_h resp_bytes | sort -nrk3 | head -5
CSjNSg2PjautayFDCk    199.7.51.190   314640
CHHYy23JnTWsOPjoee    69.147.86.184  244265
Ce17F52e1L5egkZi07    151.207.243.129        174678
CSjqes3Hu7SxsWq4x5    198.189.255.75 95603
CsTFmw4tqZOMNjwc4b    198.189.255.75 95598
```

- Included in any log file entry associated with that connection

```
$ cat http.log | zeek-cut uid id.resp_h method status_code host | grep CSjNSg2PjautayFDCk
CSjNSg2PjautayFDCk    199.7.51.190   GET    200    SVRSecure-crl.verisign.com
```

- Files also have a unique **File Unique IDentifier (FUID)** to correlate across multiple Zeek logs related to files
- **Community-ID** is an optional ID to correlate across multiple applications

JSON, jq, zcutter, and a Cheat-Sheet

- In production Zeek deployments, **JSON** format is common
  - Make these changes to your default local.zeek:

```
#@load tuning/defaults
@load tuning/json-logs
```

- But some tools need TSV, and you often find yourself converting from TSV to JSON and vice versa, or "massaging" TSV/JSON
  - Active Countermeasures team's **zcutter** can help:
    - https://github.com/activecm/zcutter
  - Tools like **jq** can help with JSON:
    - https://github.com/jqlang/jq
  - But JSON and jq has a… "steep learning curve" :D so check out this **JSON and jq Quick Start Guide**:
    - https://www.sans.org/posters/json-and-jq-quick-start-guide/

Zeek Logs + ML "Process" ☺

Zeek Logs + ML "Process" ☺

Zeek Logs Hands-On

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Machine Learning on Zeek Logs

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Tools to Parse and Analyze Zeek Logs

- From the Stratosphere Laboratory team:
  - **StratosphereLinuxIPS (SLIPS)**: https://github.com/stratosphereips/StratosphereLinuxIPS
  - **zeek_anomaly_detector**: https://github.com/stratosphereips/zeek_anomaly_detector
- From the Active Countermeasures team:
  - **Real Intelligence Threat Analytics (RITA)**: https://github.com/activecm/rita/
  - AC-Hunter™ Community Edition: https://www.activecountermeasures.com/ac-hunter-community-edition/

Zeek Logs + ML "Process" ☺

Machine Learning on Zeek Logs Hands-On (1)

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

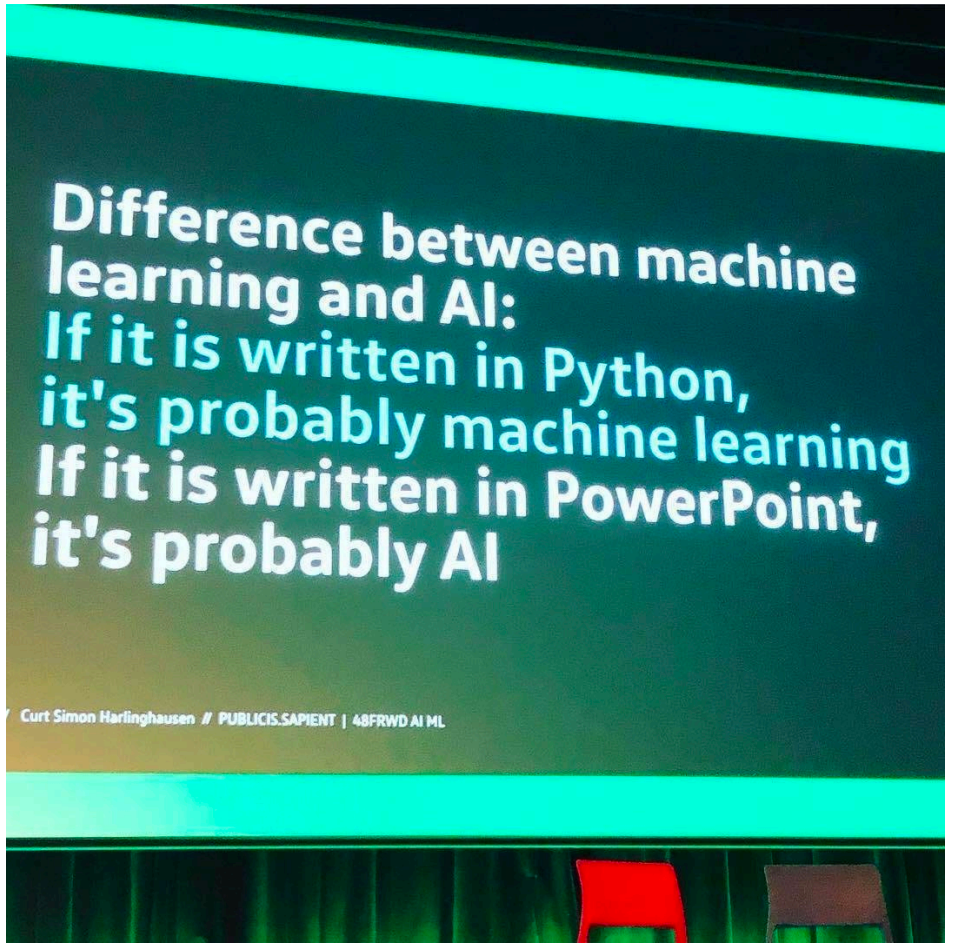
Snakes! ☺ (Anaconda and Python)

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Anaconda, Jupyter Lab, Python

- We are going to use the following tools:
  - **Anaconda**
  - **Jupyter Lab**
  - **Python** (no, not R ☺)
    - **pandas**
    - **numpy**
    - **mathplotlib**
    - **scikit-learn**
    - **tensorflow**

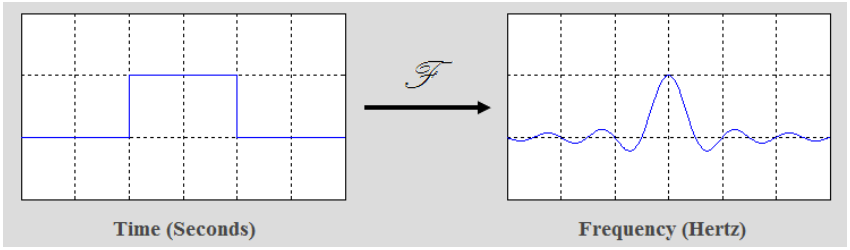


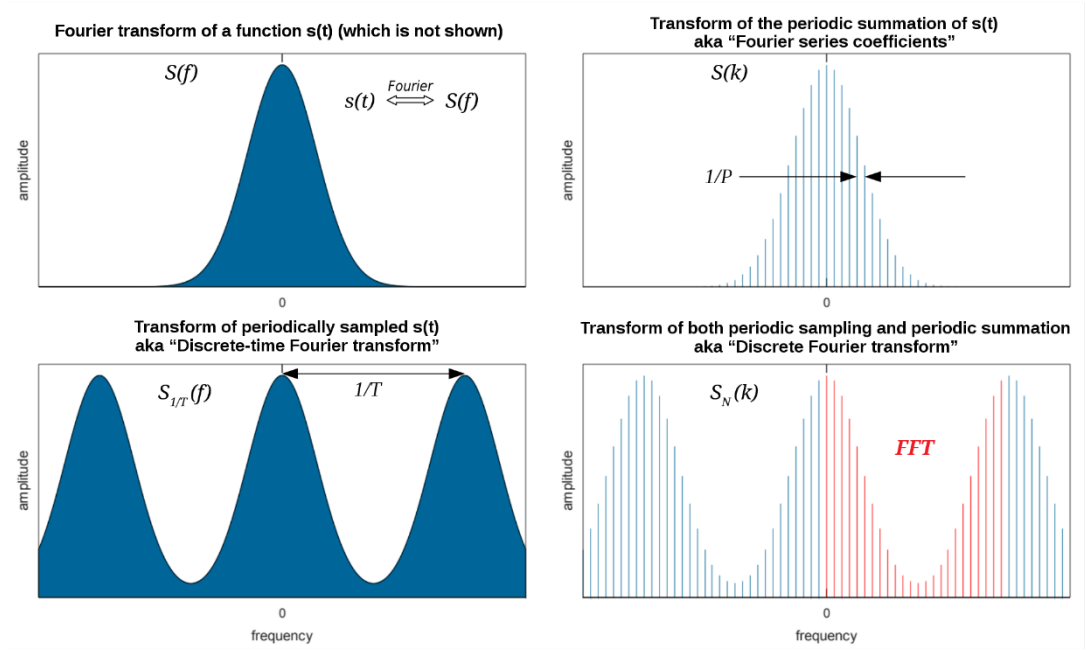
Source: https://twitter.com/aboutsecurity/status/1094277269751762946

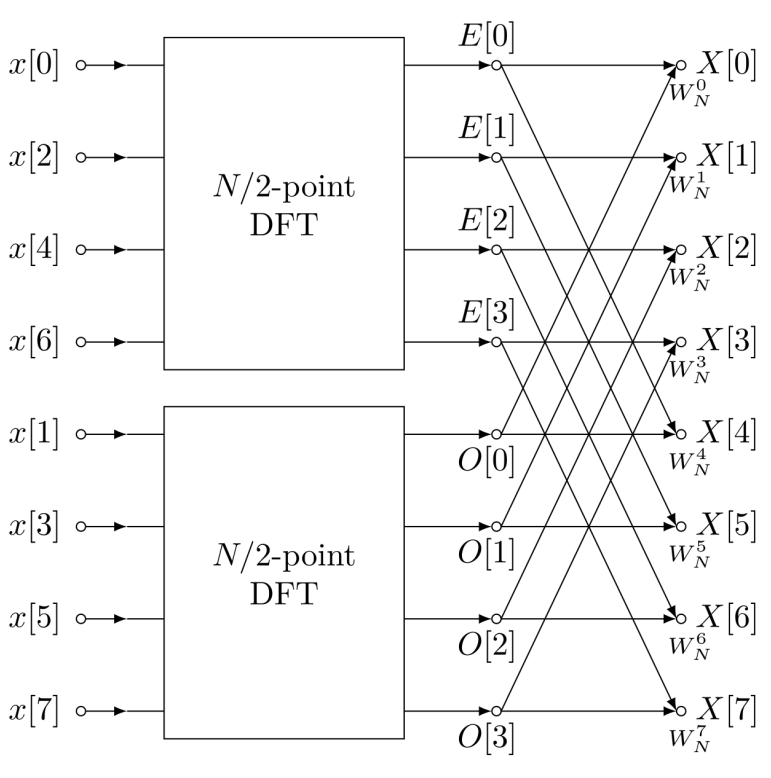
Disclaimer / Credits

- **None of this is our own research! We are just connecting dots!**
- Largely based on **David Hoelzer's** SANS "**SEC595**: Applied Data Science and AI/Machine Learning for Cybersecurity Professionals" **Day 2 labs**:
  - https://www.sans.org/cyber-security-courses/applied-data-science-machine-learning/
- Go check out David Hoelzer's YouTube channel:
  - https://www.youtube.com/@DHAtEnclaveForensics/videos
- Also, check out David Hoelzer's other presentations on the SANS Cyber Defense YouTube channel:
  - https://www.youtube.com/@SANSCyberDefense/search?query=Hoelzer
- We also used **Nik Alleyne's blog post and GitHub repository**:
  - https://www.securitynik.com/2023/10/beginning-fourier-transform-detecting.html
  - https://showmethepackets.com/index.php/2023/10/09/beginning-fourier-transform-detecting-beaconing-in-our-networks/
  - https://github.com/SecurityNik/Data-Science-and-ML/blob/main/Beginning%20Fourrier%20Transform%20for%20Beacon%20Detection%20-%20Blog.ipynb

(Discrete) Fourier Transform and (R)FFT





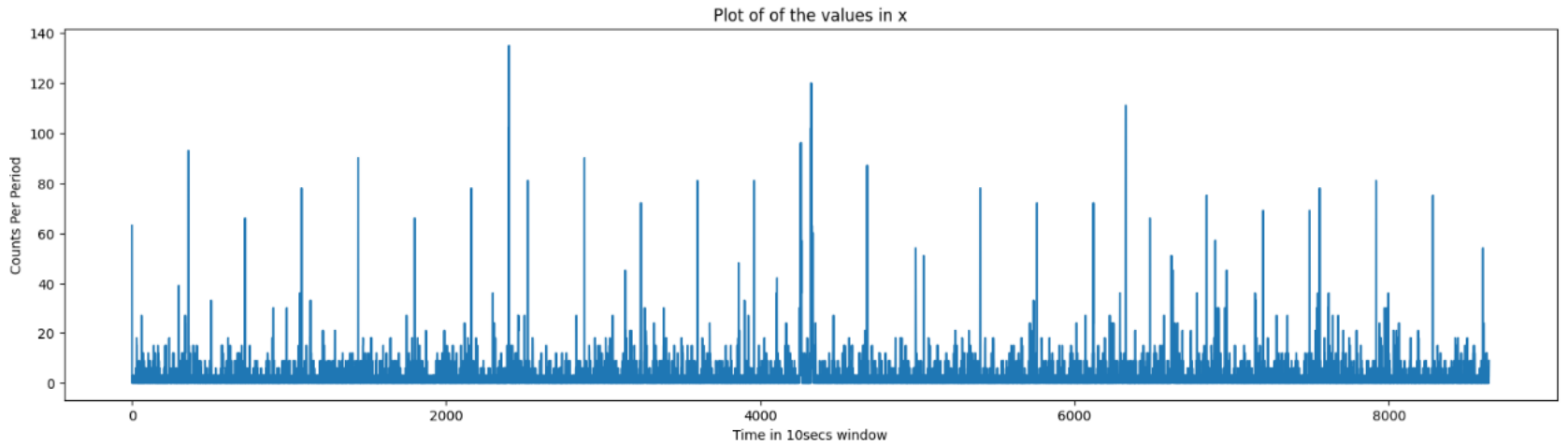


Sources: https://www.thefouriertransform.com/
https://en.wikipedia.org/wiki/Discrete_Fourier_transform and https://en.wikipedia.org/wiki/Fast_Fourier_transform

Filter files by name
/
Name Last Modified
01_Zeek_Logs_Hands-On 5 days ago
02_Machine_Learning_on_Zeek_Logs_Hands-On_... 5 days ago
03_Machine_Learning_on_Zeek_Logs_Hands-On_... 5 days ago
04_Zeek_Scripting_Hands-On 5 days ago
05_Zeek_Broker_and_Python_Hands-On_(1) 5 days ago
06_Training_ML_Models_DEMO 5 days ago
07_Zeek_Broker_and_Python_Hands-On_(2) 5 days ago
START_HERE.md 5 days ago

Launcher | 03_Machine_Learning_on_Z | Markdown | Python 3 (ipykernel)

[24]:
```
# Plot the values in x
plt.figure(figsize=(20,5))
plt.title('Plot of of the values in x')
plt.plot(x)
plt.xlabel(xlabel='Time in 10secs window')
plt.ylabel(ylabel='Counts Per Period')
plt.show()
```



Calcualte the FFT

[25]:
```
# Get the Fourier Transform of the signal
# Notice that we are using RFFT, beacase we are talking about packets, so the imaginary component does not make much sense
fft = np.fft.rfft(x)
fft, len(fft)
```

[25]:
```
(array([22611.          +0.j        ,  -974.8764059  +500.87661188j,
         230.48994964 +916.53831905j, ...,   602.5534833  +676.3881848j ,
         289.29628961 +837.45030111j,  197.33450509+3243.00721865j]),
```

Zeek Logs + ML "Process" ☺

Machine Learning on Zeek Logs Hands-On (2)

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Zeek + Broker + ML "Process" ☺

Zeek Scripting

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Zeek Scripting Overview

- **Event-driven**
- **Domain-specific**
- **Turing-complete**
- **Based on ML** (LISP-like)
- Basically, all Zeek output is generated by Zeek scripts!

Source: https://docs.zeek.org/en/master/script-reference/index.html

Types (1)

Name	Description
bool	Boolean (T = true, F = false)
count, **int**, **double**	Numeric types (count = unsigned int)
time, interval	Temporal types (e.g. 3.5mins)
string	String
pattern	Regular expression (flex lexical analyzer, e.g. /foo|bar/)
port, **addr**, subnet	Network types (e.g. 80/tcp, 192.168.0.1, 10.0.0.0/8)

Source: https://docs.zeek.org/en/master/script-reference/types.html

Types (2)

Name	Description
enum	Enumeration (user-defined type)
table, set, vector, **record**	Container types (table = hash, record = structure)
function, **event**, hook	Executable types
file	File type (only for writing)
opaque	Opaque type (for some built-in functions)
any	Any type (for functions or containers)

Source: https://docs.zeek.org/en/master/script-reference/types.html

Operators (1)

- Relational operators

Name	Syntax
Equality	a == b
Inequality	a != b
Less than	a < b
Less than or equal	a <= b
Greater than	a > b
Greater than or equal	a >= b

- Logical operators

Name	Syntax
Logical AND	a && b
Logical OR	a || b
Logical NOT	!a

Source: https://docs.zeek.org/en/master/script-reference/operators.html

Operators (2)

- Arithmetic operators

Name	Syntax
Addition	a + b
Subtraction	a - b
Multiplication	a * b
Division	a / b
Modulo	a % b

Name	Syntax
Unary plus	+a
Unary minus	-a
Pre-increment	++a
Pre-decrement	--a
Absolute value	|a|

Source: https://docs.zeek.org/en/master/script-reference/operators.html

Operators (3)

- Bitwise operators

Name	Syntax
Bitwise AND	a & b
Bitwise OR	a | b
Bitwise XOR	a ^ b
Bitwise complement	~a

- Set operators

Name	Syntax
Set intersection	s1 & s2
Set union	s1 | s2
Set difference	s1 - s2

Source: https://docs.zeek.org/en/master/script-reference/operators.html

Operators (4)

- Assignment operators

Name	Syntax
Assignment	a = b
Addition assignment	a += b
Subtraction assignment	a -= b

- Record field operators

Name	Syntax
Field access	a$b
Field value existence	a?$b

Source: https://docs.zeek.org/en/master/script-reference/operators.html

Operators (5)

- Pattern operators

Name	Syntax
Exact matching	p == s
Embedded matching	p in s
Conjunction	p1 & p2
Disjunction	p1 | p2

- Type casting

Name	Syntax
Type cast	v as t
Check if a cast is supported	v is t

Source: https://docs.zeek.org/en/master/script-reference/operators.html

Operators (6)

- Other operators

Name	Syntax
Membership test	a in b
Non-membership test	a !in b
Table or vector element access	a[b]
Substring extraction	a[b:c]

Name	Syntax
Create a deep copy	copy(a)
Module namespace access	a::b
Conditional	a ? b : c

Source: https://docs.zeek.org/en/master/script-reference/operators.html

Attributes (the most important ones)

Name	Description
&redef	Redefine a global constant or extend a type
&priority	Specify the priority for event handler or hook
&log	Mark a record field as to be written to a log
&optional	Allow a record field value to be missing
&default	Specify a default value

Source: https://docs.zeek.org/en/master/script-reference/attributes.html

Declarations

Name	Description
module	Change the current module
export	Export identifiers from the current module
local	Declare a local variable
global	Declare a global variable
const	Declare a constant
option	Declare a configuration option
type	Declare a user-defined type
redef	Redefine a global value or extend user-defined type
function, **event**, hook	Declare a function, event handler, or hook

Source: https://docs.zeek.org/en/master/script-reference/statements.html

Statements

Name	Description
add, delete	Add or delete elements
print	Print to stdout or file
for, while, next, break	Loop over each element in a container object (for), or as long as a condition evaluates to true (while)
if, else if, else	Evaluate boolean expression and if true, execute a statement

Name	Description
switch, case, break, fallthrough	Evaluate expression and execute a statement with a matching value
when	Asynchronous execution
event, schedule	Invoke or schedule an event handler
return	Return from function, hook, or event handler

Source: https://docs.zeek.org/en/master/script-reference/statements.html

Directives and Namespaces

- Directives
  - Evaluated before script execution
  - Like pre-processor macros in C/C++
- A few examples:

Name	Scope
@load	Load script
@load-plugin	Load plugin
@load-sigs	Load signature
@DIR	Directory pathname
@FILENAME	Script filename

- Namespaces

Name	Scope
Local	Local block
Module global	Global in the module
Global	All Modules

Source: https://docs.zeek.org/en/master/script-reference/directives.html

And a Bunch of Other Things…

- Protocol Analyzers
- File Analyzers
- Hooks
- Zeek script debugging
- Zeek Plugins
- Go, check the documentation:
  - https://docs.zeek.org/en/master/script-reference/index.html

Zeek + Broker + ML "Process" ☺

Zeek Scripting Hands-On

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Zeek Broker and Machine Learning

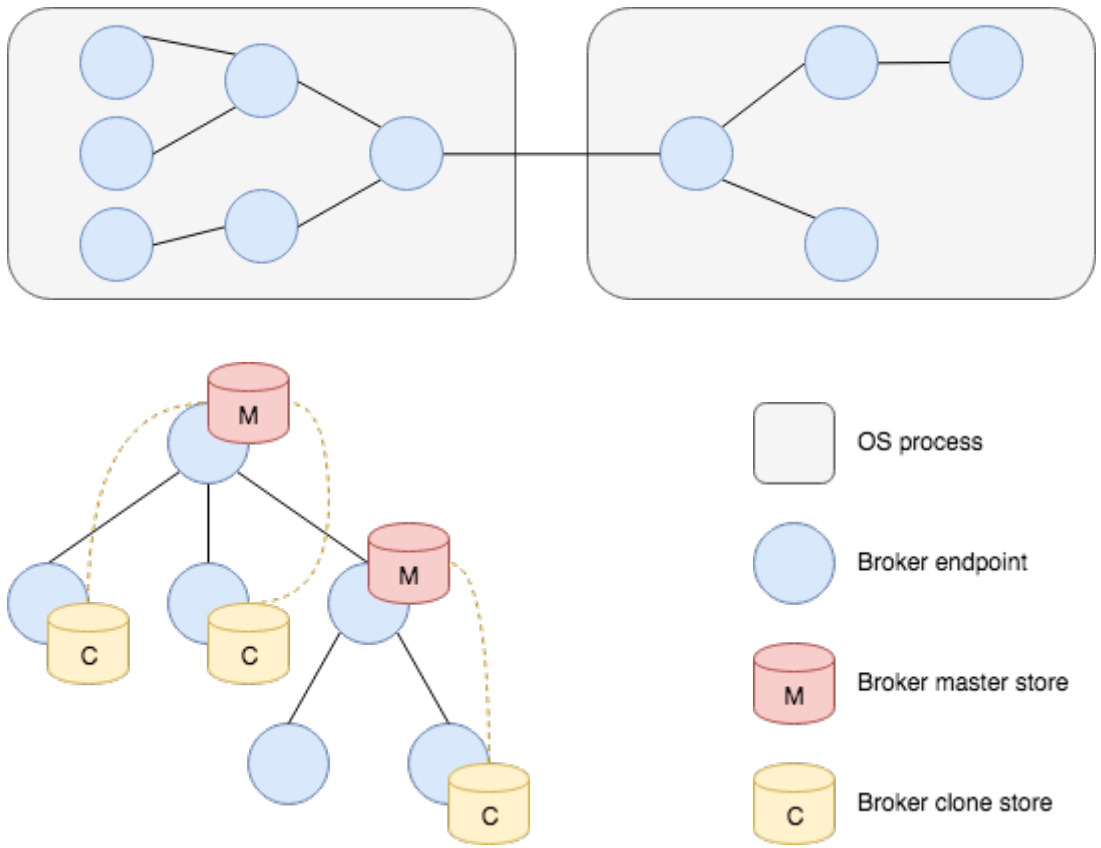
Hack.lu – 2024 October

Zeek Broker

- Broker is a library for type-rich **publish/subscribe communication**
- It is the successor of Broccoli
- **It enables arbitrary applications to communicate in Zeek's data model**
- Broker also offers **distributed key-value stores** to facilitate unified data management and persistence

Broker Overview

- **Endpoints**: data senders and receivers.
- **Peering**: to publish or receive messages an endpoint needs to peer with other endpoints.
- **Messages**: information to send
- **Topics**: filters for a publish or subscribe communication pattern.
- **Subscriptions**: peers only receive messages which match one of their subscriptions.
- **Data stores**: distributed key-value stores.



Broker's Python Bindings

- **Almost all functionality of Broker** is also accessible through Python bindings
- The Python API mostly mimics the C++ interface
- Installation:
  - Virtual Environment (compiled Broker from source)
  - **Binary package: $(PREFIX)/zeek/lib/zeek/python/**

```
import sys
sys.path.append('/opt/zeek/lib/zeek/python/')
```

- **Your Broker version must match your Zeek version!**

Zeek 7.0.0 and Broker

- Lots of work (changes) being done on Broker for the next Zeek major version
- **Prefer WebSockets over the Python bindings** (the later most probably will be discontinued)
- Keep and eye on:
  - **https://github.com/zeek/broker/blob/master/CHANGES**
  - https://github.com/zeek/zeek/blob/master/CHANGES

Disclaimer / Credits (1)

- **None of this is our own research! We are just connecting dots!**
- Largely based on **David Hoelzer's livestream recordings, presentations, and his SANS "SEC595**: Applied Data Science and AI/Machine Learning for Cybersecurity Professionals" **Day 3 and Day 5 labs**:
  - https://www.sans.org/cyber-security-courses/applied-data-science-machine-learning/
- Go check out David Hoelzer's YouTube channel:
  - https://www.youtube.com/@DHAtEnclaveForensics/videos
- We used code from these videos:
  - **Machine Learning with Zeek and Tensorflow Part 1: https://www.youtube.com/watch?v=5w25kEMLdQk**
  - **Machine Learning with Zeek and Tensorflow - Part 2: Processing the Data: https://www.youtube.com/watch?v=8qJFnX214yE**
  - **Threat Hunting with Data Science, Machine Learning, and Artificial Intelligence: https://www.youtube.com/watch?v=fdqFdnkf9I4**
- Also, check out David Hoelzer's other presentations on the SANS Cyber Defense YouTube channel:
  - https://www.youtube.com/@SANSCyberDefense/search?query=Hoelzer

Disclaimer / Credits (2)

- We used the code from the **Zeek Broker documentation**:
  - https://docs.zeek.org/projects/broker/en/master/python.html
- We also used a little **Dr. Keith Jones' "How To Easily Connect Zeek to Python"** YouTube video, blog, and GitHub repository:
  - https://www.youtube.com/watch?v=iIYi17VqFkY
  - https://drkeithjones.com/index.php/2023/03/11/how-to-connect-zeek-to-python/
  - https://github.com/keithjjones/zeek-python-broker-demo
- Go check out Dr. Keith Jones' YouTube channel:
  - https://www.youtube.com/@dr.keithjones

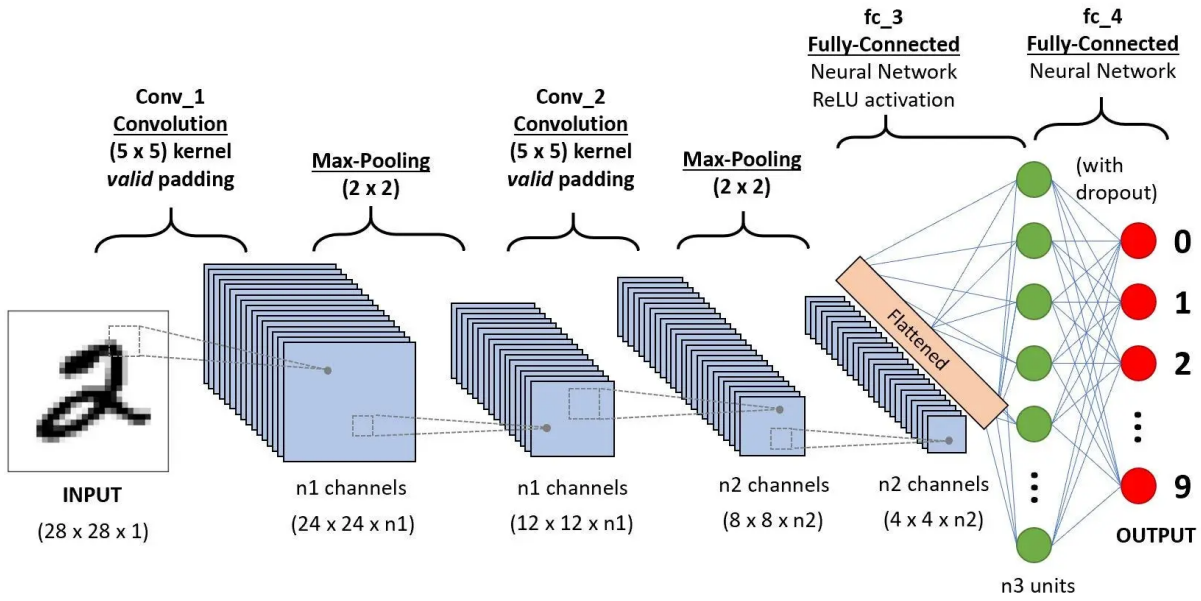
Zeek + Broker + ML "Process" ☺

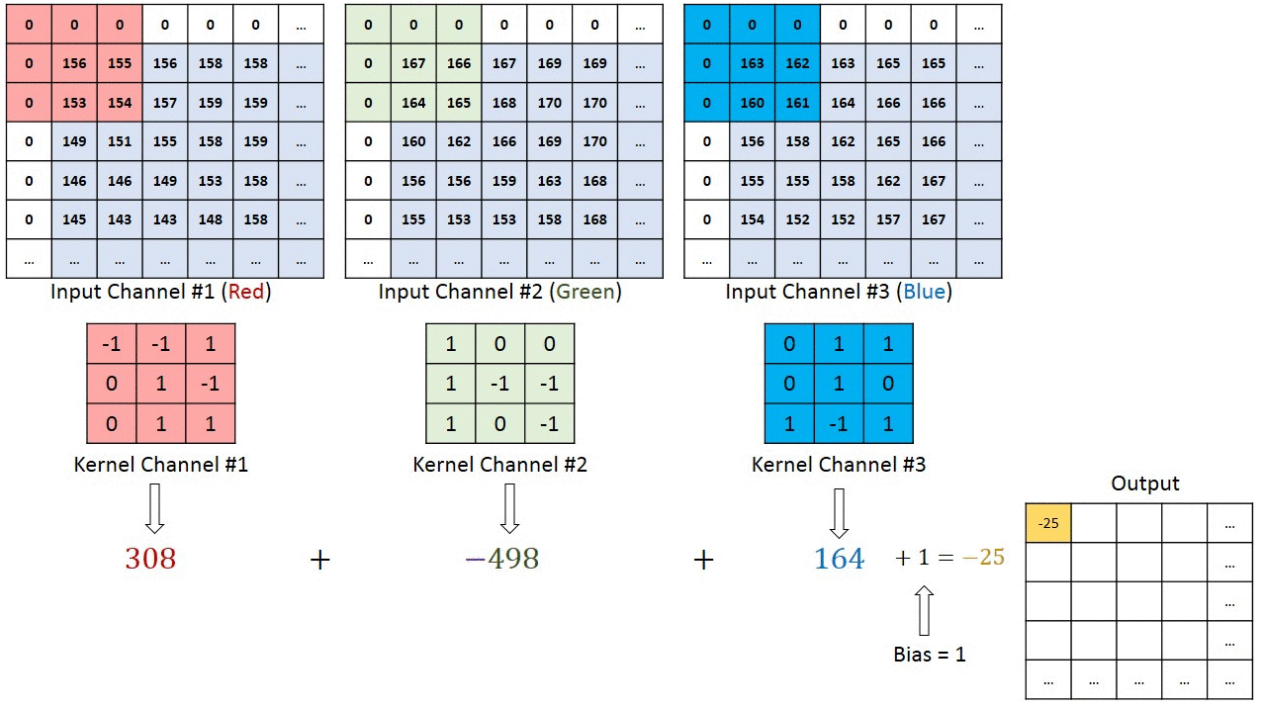
Zeek Broker and Python Hands-On (1)

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Convolutional Neural Networks

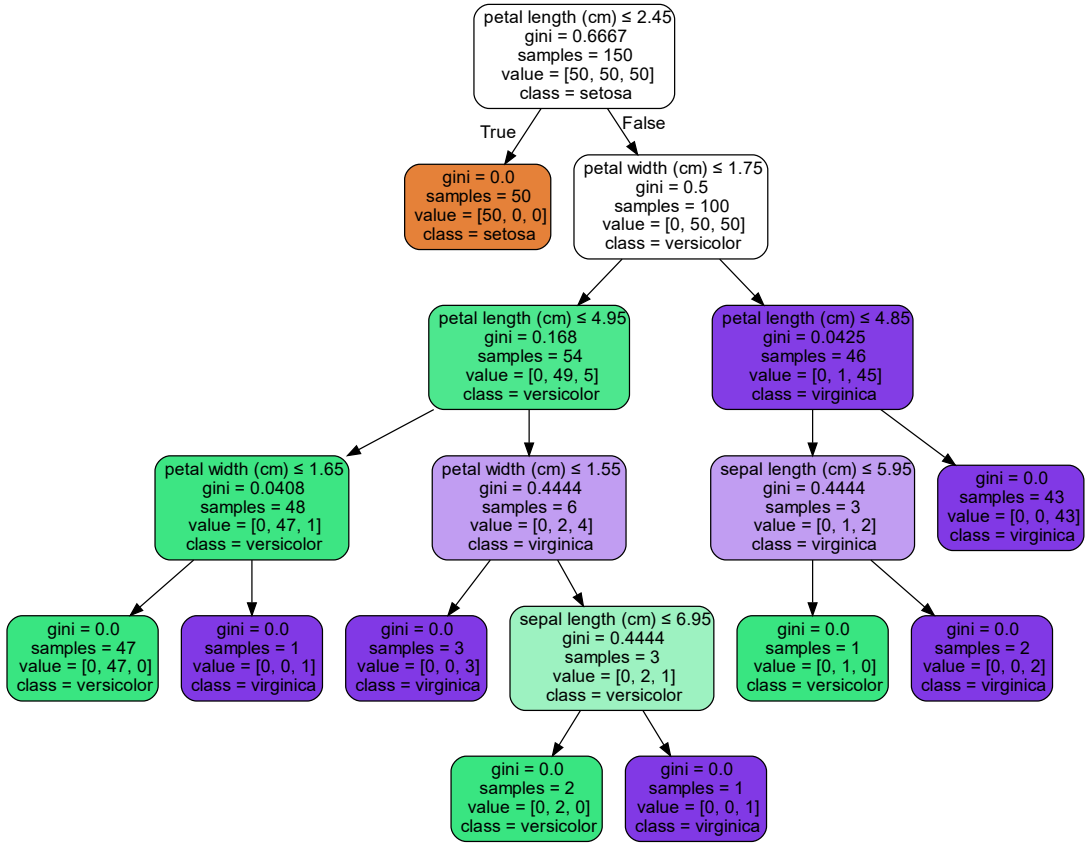




Source: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

Decision Tree Classifier

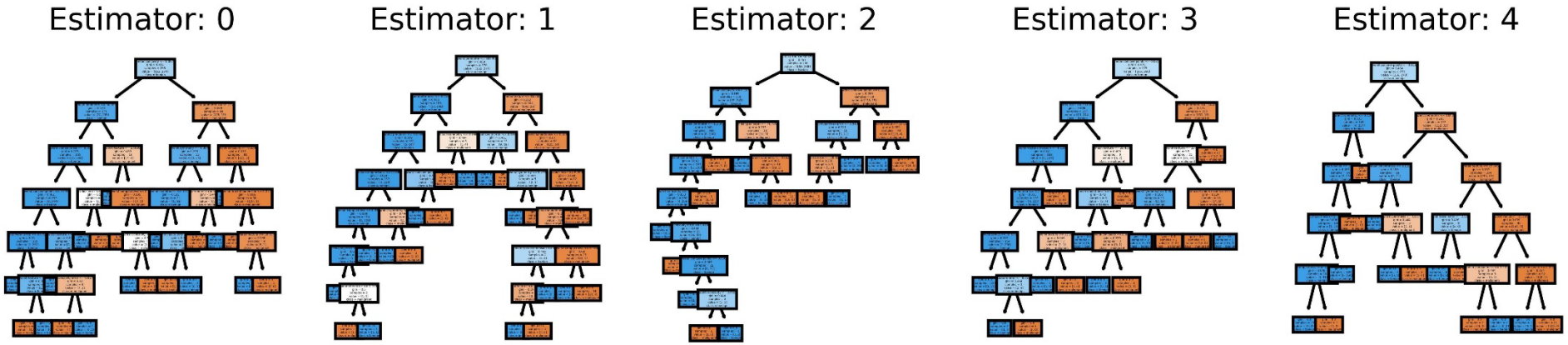
- **Non-parametric supervised learning** method
- **Gini coefficient**: represents the differences between areas
- The algorithm used to build the decision tree is attempting to **maximize the information gain at every decision node**



Source: https://scikit-learn.org/stable/modules/tree.html

Random Forest Classifier

- **Decision Trees have an issue with outliers in the data.**
- **Let's build a forest!** Each tree makes a determination, classifies that sample, and gets one vote. Whichever category has the greatest number of votes, is the final classification.



Source: https://stackoverflow.com/questions/40155128/plot-trees-for-a-random-forest-in-python-with-scikit-learn

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/

Name	Last Modified
data	7 days ago
models	5 days ago
Malware_Detection_Models.ipynb	5 days ago
malwarebytes.py	7 days ago

Malware_Detection_Models.i

Markdown

Python 3 (ipykernel)

dropout_2 (Dropout)	(None, 122, 122, 32)	0
flatten (Flatten)	(None, 476288)	0
dense (Dense)	(None, 1)	476,289

Total params: 1,457,165 (5.56 MB)
Trainable params: 485,721 (1.85 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 971,444 (3.71 MB)

[20]:
```
cnn_model_loaded.evaluate(x_test, y_test)
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.7930 - loss: 0.6321
```

[20]:
```
[0.36280587315559387, 0.8600000143051147]
```

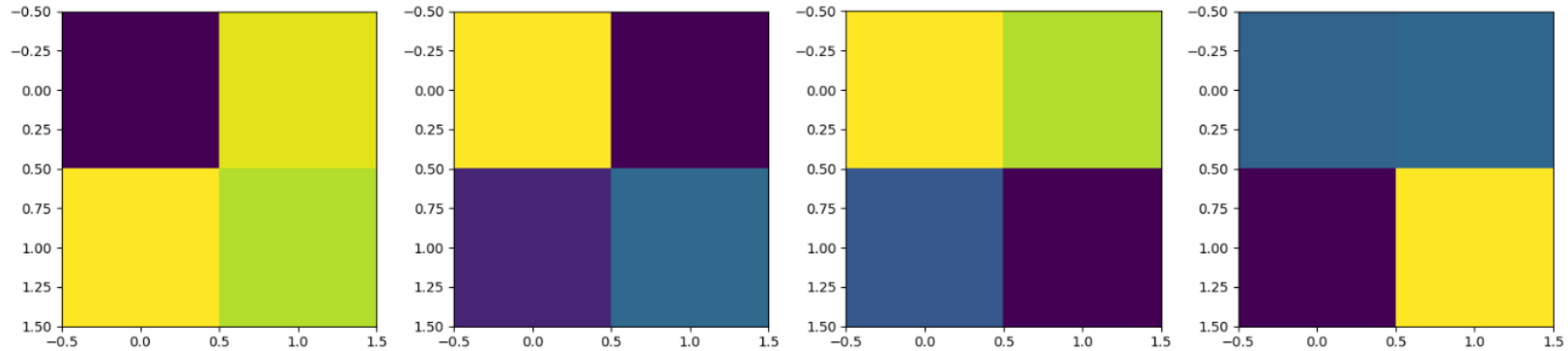
But What Does It Do?

[21]:
```
def plot_filters(layer, kernels, filters, kernel_size):
    figure = plt.figure(figsize=(16,8))
    subplots = []
    rows = filters // 4
    cols = 4
    for i in range(kernels[0].shape[3]):
        subplots.append(figure.add_subplot(rows, cols, i+1))
        #plt.imshow(kernels[0][:,:,:,i].reshape(kernel_size,kernel_size), cmap='gray')
        plt.imshow(kernels[0][:,:,:,i].reshape(kernel_size,kernel_size*kernels[0].shape[2]))
    plt.tight_layout()
    plt.show()
```

[22]:
```
conv_layer = cnn_model.get_layer('conv2d_layer_1')
kernels = conv_layer.get_weights()
filters=kernels[0].shape[-1]
kernel_size= kernels[0].shape[1]
kernels[0].shape
```

[22]:
```
(2, 2, 1, 8)
```

[23]:
```
plot_filters(conv_layer, kernels, filters, kernel_size)
```



Simple 0 1 Python 3 (ipykernel) | Idle

Mode: Command Ln 1, Col 1 Malware_Detection_Models.ipynb 1

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/

Name	Last Modified
data	7 days ago
models	5 days ago
Malware_Detection_Models.ipynb	5 days ago
malwarebytes.py	7 days ago

Malware_Detection_Models.i

Markdown

Python 3 (ipykernel)

```
Wall time: 3.82 s
[Parallel(n_jobs=16)]: Done 768 tasks      | elapsed:    0.1s
[Parallel(n_jobs=16)]: Done 1000 out of 1000 | elapsed:    0.2s finished
```

[77]:
```
0.91
```

[78]:
```
predictions = random_forest_classifier.predict(sample_check)
print(f"Ground Truth: [1, 0]\tPredicted: {predictions}")
```

```
[Parallel(n_jobs=16)]: Using backend ThreadingBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done  18 tasks      | elapsed:    0.0s
[Parallel(n_jobs=16)]: Done 168 tasks      | elapsed:    0.0s
[Parallel(n_jobs=16)]: Done 418 tasks      | elapsed:    0.0s
[Parallel(n_jobs=16)]: Done 768 tasks      | elapsed:    0.1s
Ground Truth: [1, 0]    Predicted: [1. 0.]
[Parallel(n_jobs=16)]: Done 1000 out of 1000 | elapsed:    0.1s finished
```

But What Does it Do?

[79]:
```
# https://stackoverflow.com/questions/40155128/plot-trees-for-a-random-forest-in-python-with-scikit-learn

fig, axes = plt.subplots(nrows = 1,ncols = 5,figsize = (10,2), dpi=900)
for index in range(0, 5):
    tree.plot_tree(random_forest_classifier.estimators_[index],
                   filled = True,
                   ax = axes[index]);
    axes[index].set_title('Estimator: ' + str(index), fontsize = 11)
```

Saving and Reloading the Model

[80]:
```
dump(random_forest_classifier, "./models/RandomForestClassifier1000MalwareDetectionModel.joblib")
```

[80]:
```
['./data/models/RandomForestClassifier1000MalwareDetectionModel.joblib']
```

[81]:
```
random_forest_classifier_loaded = load("./models/RandomForestClassifier1000MalwareDetectionModel.joblib")
```

[82]:
```
predictions = random_forest_classifier_loaded.predict(x_test)
accuracy(y_test, predictions)
```

Simple 0 1 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 Malware_Detection_Models.ipynb 1

Zeek + Broker + ML "Process" ☺

Training ML Models DEMO

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Zeek + Broker + ML "Process" ☺

Zeek Broker and Python Hands-On (2)

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Conclusion

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY

Slides and Contact

- **Google Drive**: https://drive.google.com/drive/folders/1bMbuLpS9GVE_3d3bHhYeRHdIvFPNiaRw?usp=sharing
- **Shortened**: https://bit.ly/3TIZtDk

E-mails: eva.szilagyi@alzetteinfosec.com
david.szili@alzetteinfosec.com

References

- **Zeek Documentation**
  - https://www.zeek.org/documentation/index.html
- **Zeek Broker Documentation**
  - https://docs.zeek.org/projects/broker
- **Install Zeek**
  - https://docs.zeek.org/en/master/install.html
- **Zeek on DockerHub**
  - https://hub.docker.com/u/zeek
- **Try Zeek Online**
  - http://try.zeek.org

Questions?

Hack.lu – 2024 October

ALZETTE
INFORMATION SECURITY