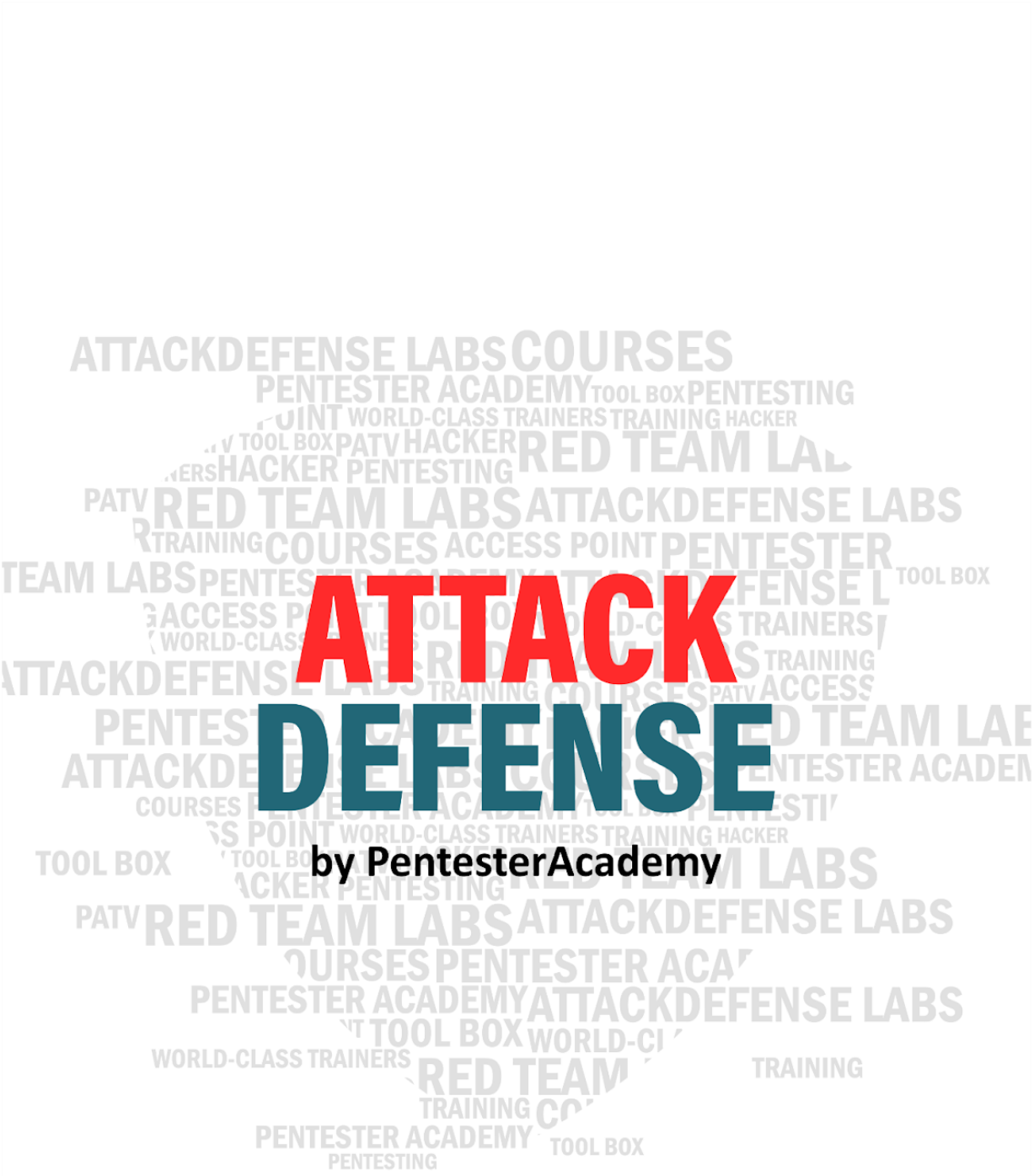
ATTACK
DEFENSE
by PentesterAcademy

Name	Log Anomaly Detection Basics
URL	https://www.attackdefense.com/challengedetails?cid=141
Type	Forensics : Webserver Log Analysis

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic.

Already parsed web access log file logs.txt is provided. The log file contains more than 1 million logs which follow a pattern/rule set. However, in addition to real logs, there are 5 anomalous log entries in the file.

Objective: You have to identify the rule set/pattern and then use it to find the 5 anomalous log entries.

Answer:

HEADER
http://example.net/access_object.php?param1=2828&param2=brachy&param3=22JUL2018

DELETE
http://example.net/access_object.php?param1=2931&param2=brachy&param3=33JAN2013

DELETE
http://example.net/access_object.php?param1=2823&param2=brachi&param3=20XYZ2019

CONNECT
http://example.net/access_object.php?param1=8119&param2=brachm&param3=23JUL2011

POST
http://example.net/access_object.php?param1=3970&param2=crachy&param3=08SEP2013

Solution:

Observe that each log is made up of multiple parts i.e. HTTP method, param1, param2 and param3.

Look closely at 10-15 logs, one can observe the following:

1. Different HTTP request methods are there in different logs
2. Param1 looks like a number from a sequence or range
3. Param2 is always a word starting with the phrase “brac” (sometimes uppercase and sometime in lowercase)
4. Param4 is date (DDMMMYYYY)

In order to spot the anomaly in these, write python code to read, tokenize the logs and then represent them in such a manner that the anomaly can be easily spotted.

Following code can be used

========== Code ==========

```
# Sample log: CONNECT
http://example.net/access_object.php?param1=1350&param2=brachy&param3=20APR2013

# Set for storing HTTP Methods
set_methods = set()

# Set for storing dates
set_dates= set()

# Set for storing Months
set_mon=set()

# Set for storing Year
set_year=set()

# List for Param1
num_list=[]


# Opening log file for reading
```

```
with open("logs.txt") as f:
    file_content = f.readlines()

# Removing newline
file_content = [x.strip() for x in file_content]

# Iterating over file_content line by line
for line_item in file_content:

                # Parsing and adding HTTP Method into set
        token_string = line_item.split(' ')
        set_methods.add(token_string[0])

                # Tokenizing remaining string
        params = token_string[1].split('&')

                # Param1
        p1 = params[0].split('=')
        num_list.append(p1[1])

                # Param2
        p2 = params[1].split('=')

                # Checking if param2 matches the criteria
        if "brac" not in str(p2[1]).lower():
                print "Anomalous entry:  "+p2[1]

                # Param3
        p3 = params[2].split('=')
        date_str = p3[1]

                # Adding date, month and year to separate sets
        set_dates.add(date_str[0:2])
        set_mon.add(date_str[2:5])
        set_year.add(date_str[5:])



# Printing all sets in the end
```

```
# HTTP Methods
print set_methods

# Dates
print set_dates

# Months
print set_mon

# Years
print set_year

# Soring the list of Param1
num_list.sort()
# We should print this list in second run because due to its big size, it will push the sets out of screen
#print num_list
```

=================

Put this code to a new file i.e. detect.py file and execute it. On running this code, one can get following output. The anomalies are highlighted.

```
student@attackdefense:~$ python detect.py
Anomalous entry:  crachy
set(['HEAD', 'TRACE', 'GET', 'HEADER', 'CONNECT', 'PUT', 'POST', 'OPTIONS', 'DELETE'])
set(['07', '02', '25', '26', '01', '20', '21', '22', '05', '08', '09', '28', '24', '03', '27', '06', '14', '11', '10', '13', '12', '15', '04', '17', '16', '19', '18', '23', '33'])
set(['MAR', 'FEB', 'AUG', 'SEP', 'APR', 'XYZ', 'JUN', 'JUL', 'JAN', 'MAY', 'NOV', 'DEC', 'OCT'])
set(['2020', '2019', '2018', '2015', '2014', '2017', '2016', '2011', '2010', '2013', '2012'])
student@attackdefense:~$
```

For the last one, uncomment the last line and print the num_list. After sorting the list, number 8119 is way out of the range. Hence, is an anomaly.

```
 '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '40
01', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001'
 '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4
01', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '4001', '8119']
```

Simply do grep on logs to find the complete log entry.