ESIEE Paris

Artificial Intelligence and Cybersecurity

# Network Security

Lab 3 Report

*Bruno Luiz Dias Alves de Castro*
*Victor Gabriel Mendes Sündermann*

April 22, 2023

# Contents

# 1 Introduction

This report is constituted of two parts, the first one aims to create and configure a DNS server and to study the attacks that can be done on it. The attacks tackeld in this part are: DNS cache poisoning, spoofing DNS responses and sniffing and spoofing of packets. The second part is an implementation of a Firewall to filter packets and block the communication between two machines.

# 2 Part1: Local DNS Attack

## 2.1 Setting up the environment

In order to set up the Virtual Machines we used the Oracle VMBox to set them up and utilized a Debian Linux image, the IP of the machines were set as following: User: 10.0.2.18 Attacker: 10.0.2.17 Server: 10.0.2.16

After creating and configuring the VM's they were connected to the same NAT network which has the address:

For the server to be initialized it's necessary to run a DNS server software, for this lab the BIND9 was used and the following figure shows that rhe server was able to communicate with external webites:

```
bruno@debian:~$ dig www.google.com

; <<>> DiG 9.16.33-Debian <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32340
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         24      IN      A       216.58.213.68

;; Query time: 4 msec
;; SERVER: 192.168.64.1#53(192.168.64.1)
;; WHEN: Fri Apr 21 17:39:42 CEST 2023
;; MSG SIZE  rcvd: 59
```

Now that our server is running we can visualize the packets that went trought it with Wireshark, the picture bellow shows the results from a ping to Google's website:

| No. | Time | Source | Destination | Protoc | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000000 | 192.168.64.9 | 192.168.64.8 | DNS | 74 | Standard query 0x4fd9 A www.google.com |
| 2 | 0.000038081 | 192.168.64.9 | 192.168.64.8 | DNS | 74 | Standard query 0xbed1 AAAA www.google.com |
| 5 | 0.001634928 | 192.168.64.9 | 192.168.64.1 | DNS | 74 | Standard query 0x4fd9 A www.google.com |
| 6 | 0.001649344 | 192.168.64.9 | 192.168.64.1 | DNS | 74 | Standard query 0xbed1 AAAA www.google.com |
| 7 | 0.042901933 | 192.168.64.1 | 192.168.64.9 | DNS | 102 | Standard query response 0xbed1 AAAA www.go |
| 8 | 0.042902016 | 192.168.64.1 | 192.168.64.9 | DNS | 90 | Standard query response 0x4fd9 A www.googl |
| 11 | 0.059208757 | 192.168.64.9 | 192.168.64.8 | DNS | 86 | Standard query 0xc67f PTR 68.213.58.216.in |
| 13 | 0.060252060 | 192.168.64.9 | 192.168.64.1 | DNS | 86 | Standard query 0xc67f PTR 68.213.58.216.in |
| 14 | 0.070718332 | 192.168.64.1 | 192.168.64.9 | DNS | 183 | Standard query response 0xc67f PTR 68.213. |
| 17 | 1.075495205 | 192.168.64.9 | 192.168.64.8 | DNS | 86 | Standard query 0xf0e6 PTR 68.213.58.216.in |
| 19 | 1.076872109 | 192.168.64.9 | 192.168.64.1 | DNS | 86 | Standard query 0xf0e6 PTR 68.213.58.216.in |
| 20 | 1.079581629 | 192.168.64.1 | 192.168.64.9 | DNS | 183 | Standard query response 0xf0e6 PTR 68.213. |
| 23 | 2.069563616 | 192.168.64.9 | 192.168.64.8 | DNS | 86 | Standard query 0xae97 PTR 68.213.58.216.in |
| 25 | 2.070210737 | 192.168.64.9 | 192.168.64.1 | DNS | 86 | Standard query 0xae97 PTR 68.213.58.216.in |
| 26 | 2.071302578 | 192.168.64.1 | 192.168.64.9 | DNS | 183 | Standard query response 0xae97 PTR 68.213. |
| 29 | 3.077294369 | 192.168.64.9 | 192.168.64.8 | DNS | 86 | Standard query 0xb388 PTR 68.213.58.216.in |
| 31 | 3.078264302 | 192.168.64.9 | 192.168.64.1 | DNS | 86 | Standard query 0xb388 PTR 68.213.58.216.in |
| 32 | 3.080826665 | 192.168.64.1 | 192.168.64.9 | DNS | 183 | Standard query response 0xb388 PTR 68.213. |
| 3 | 0.001453732 | 192.168.64.8 | 192.168.64.9 | ICMP | 102 | Destination unreachable (Port unreachable) |
| 4 | 0.001453816 | 192.168.64.8 | 192.168.64.9 | ICMP | 102 | Destination unreachable (Port unreachable) |
| 9 | 0.044053603 | 192.168.64.9 | 216.58.213.68 | ICMP | 98 | Echo (ping) request  id=0x94dc, seq=1/256, |

But a problem arises as the server is unreachable, this is caused by adding the following line of code requested at Step 1 from Task 2, ultimately causing the server to not be initialized.

```
options{
  dump-file "/var/cache/bind/dump.db";
};
```

# 3 Bonus: Evaluation function improvement

After implementing and running countless test to all algorithm proposed, we identified a serious problem with the evaluation function used in the project: It was the main piece holding our implementations back, and the perormance of our algorithms were suffering a lot because of it.

The default evaluation function implemented in the project is the ***scoreEvaluationFunction***. The evaluation function only takes into account the current score of the games. It gets the work done in some of the cases, but has two serious draw-backs: It doesn't take into account the number of food left in the maze, and it doesn't take into account the distance to the closest food. As a result, when the Pacman gets "isolated" in a part of the map without any food, it hasn't anything to maximize, and needs a ghost to approch in order to proceed the game.

To solve this problem, we sugest a new evaluation function, called ***betterEvaluationFunction***. It's implementation and functionallity is explained in the following section.

## 3.1 The new evaluation function: betterEvaluationFunction

The proposed evaluation function takes into account this two extra parameters (along with the score): *the number of food pallets left* and *the distance to the closest food*. The number of food pallets left encourages the Pacman to eat more pallets when stuck, and the distance to the closest food will prevent the Pacman of isolating itself in the maze.

the implementation is described in the following section.

### 3.1.1 betterEvaluationFunction implementation

To implement the new evaluation function, we created a new function in the code, as follows:

```
def betterEvaluationFunction(currentGameState):
  """
  Your extreme ghost-hunting, pellet-nabbing, food-gobbling, unstoppable
  evaluation function (question 8).

  DESCRIPTION: <write something here so we know what you did>
  """

  def closestFoodDistance(gameState):
    pacmanPosition = gameState.getPacmanPosition()
    foodList = gameState.getFood().asList()

    if len(foodList) == 0:
        return 0

    return min([manhattanDistance(pacmanPosition, food) for food in foodList])

  closestFood = closestFoodDistance(currentGameState)
  return currentGameState.getScore() - (10 * closestFood) \
    - (100 * currentGameState.getNumFood())
```

Table 1: betterEvaluationFunction implementation

For calculation the distance to the closest food, we used the the Manhattan Distance (implemented in the function itself), and the number of food is retrieved from the game state. They are multiplied by 10 and 100, respectively, in order to have a bigger impact in the final score, and subtracted from the current game score.

The results of our implementation is showed in the next section.

### 3.1.2 betterEvaluationFunction tests

To test the new betterEvaluationFunction, we repeated the same tests executed before, now using the this new evaluation function instead of the old scoreEvaluationFunction. We chose the Expectimax agent to run the tests. Its description and implementation can be found in Section **??**.

The result of the tests can be found in Table 2.

| Depth | 2 | 3 | 4 |
|---|---|---|---|
| Average | 1228.6 | 1340.15 | 1131.25 |
| Best | 1724 | 1757 | 1743 |
| Win Rate | 19/20 (0.95) | 20/20 (1.00) | 19/20 (0.95) |

Table 2: Expectimax test results with betterEvaluationFunction

Comparing both results of Tables **??** and 2, it's possible to observe a huge gain in performance. By only altering the evaluation function, we were able to obtain way batter results, even reaching a Win Rate of 100% in some test cases, and performing well in the cases with lower depths.

# 4 Conclusion

This lab was essential on the understanding of the DNS server and the Firewall, by allowing us to create and configure them first-hand. We were able to fully understand how to configure, run and realize attacks on a DNS server, we also got more used to the Linux environment as it was necessary to configure files and run commands on the bash terminal. On top of that we also gained knowledge on the form of communication between the machines by using the Wireshark and Scapy tool to visualize the traffic on the network and the packets sent.