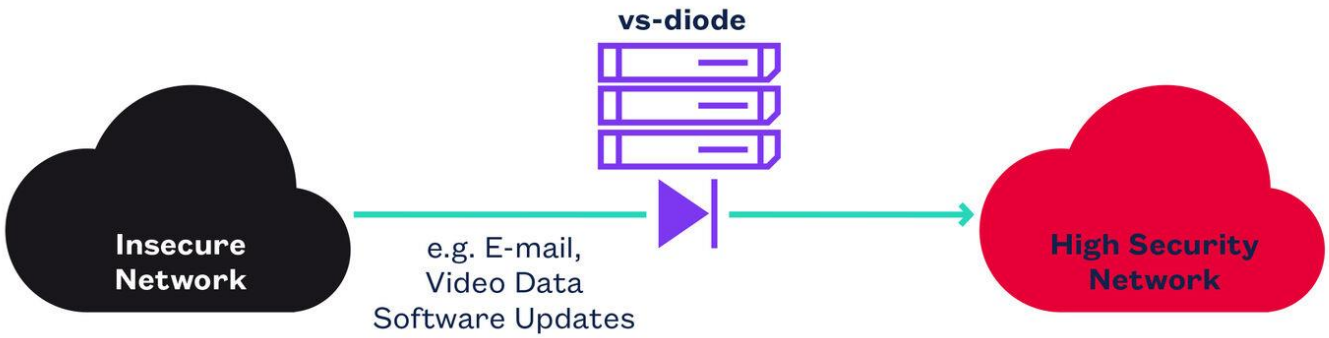
DIODE

PROJECT



Communication protocols defense curse

Shlomit Ashkenazi 323024034

Stav Avitan 323968859

Overview

1. This project engaged in software-diodes.
2. Focus on our own possible solution for a software diode.
3. This solution handles transition of files to a network restricted area through the software diode.

Requirements

1. Python3.10
2. tqdm python library
   (you can install using: pip insall tqdm)

Testing

1. we simulated a true scenario (over our own home network and devices).
2. If you don’t have a “Lab” - You can use Docker for testing.

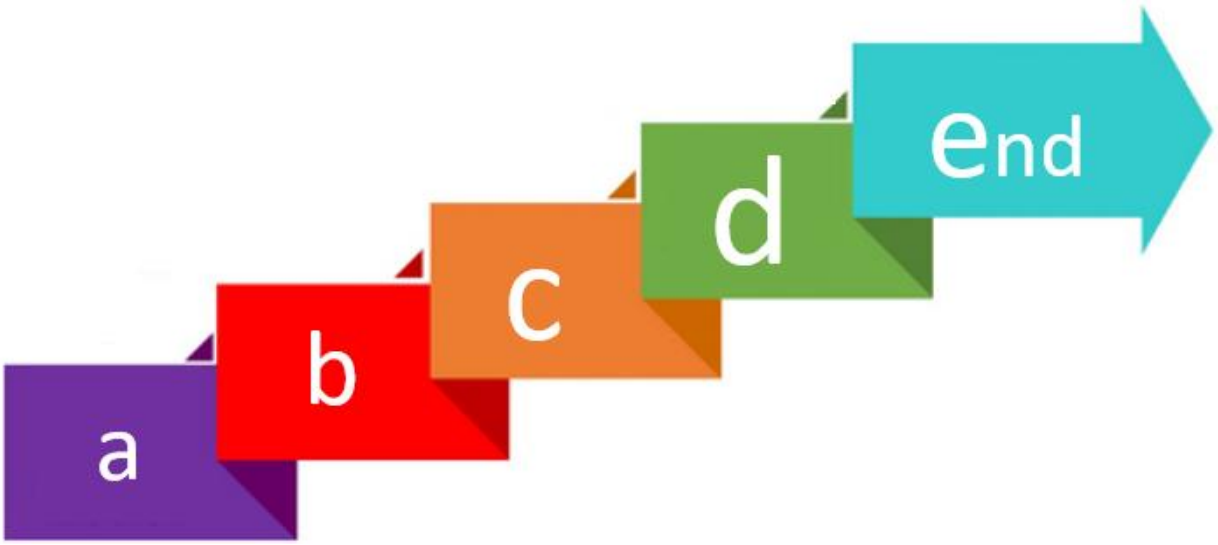
Links (git)

https://github.com/slomit1234/diode-final

Our solution

1. Our Solution consists from 4 different parts:
    1. **Source-user** – choose a file and send it over TCP to Proxy
    2. **Proxy** – acts as a buffer from all network users to the diode. Additional, we can add to it more security policies (such as file scanning, exec) to make the process as safe as possible. The Proxy transfers the data over TCP with the use in "Waterfall model"
    3. **Diode** – gets data only from the Proxy (otherwise, it would not continue in socket). the diode will transfer the data to the end-user (sensitive area) using TCP.
    4. **End-user** – agrees to communicate with the diode only (in real life you can do it by fixing the diode IP address).

2. High level Flowchart



a. The client sends a request to the proxy server.
b. The proxy server receives the request and forwards it to the diode.
c. The diode receives the request and sends it to the restricted area server.
d. The restricted area gets the file and saves it as “received” file.
e. Every Hop calculates the file MD5 on it’s own. (for comparing purpose).

3. Pros & Cons comparing to the RUDP solution

Pros:

1. TCP provides reliable and ordered delivery of data, which can be important for certain applications.
2. TCP includes built-in mechanisms for flow control and congestion avoidance, which can help prevent network congestion and improve overall performance.
3. TCP includes built-in error checking and retransmission mechanisms, which can help ensure the integrity of the data being transmitted.

Cons:

1. TCP includes more overhead than UDP, which can result in slower transmission speeds and increased network traffic.
2. TCP is more susceptible to network congestion and delays than UDP, which can negatively impact real-time applications.

3. TCP requires more processing power and memory than UDP, which can be a concern for devices with limited resources.

4. Diode Policies in our solution

   In our implementation (works over TCP), the diode maintains its one-direction policy by using a TCP server socket to listen for incoming connections from the proxy, and a TCP client socket to connect to the proxy. The diode only accepts incoming connections from the proxy and does not actively initiate connections to the proxy. This means that data can only flow from the proxy to the diode and not the other way around, maintaining the one-direction policy of the diode.

5. Waterfall model

the data transfer from the proxy to the diode using TCP follows a Waterfall model, in the sense that the data is pushed downstream in a one-way flow without any feedback or acknowledgement from

the receiver. In this implementation, the proxy sends the data to the diode using the send() method of the socket object. The diode then receives the data using the recv() method of the client socket object, and the received data is stored in the buffer. There is no feedback or acknowledgement sent back to the proxy, which makes this a one-way data transfer in a Waterfall model.

Getting Started

source user

The process always starts with a need, in our case passing files to the organization restricted area.

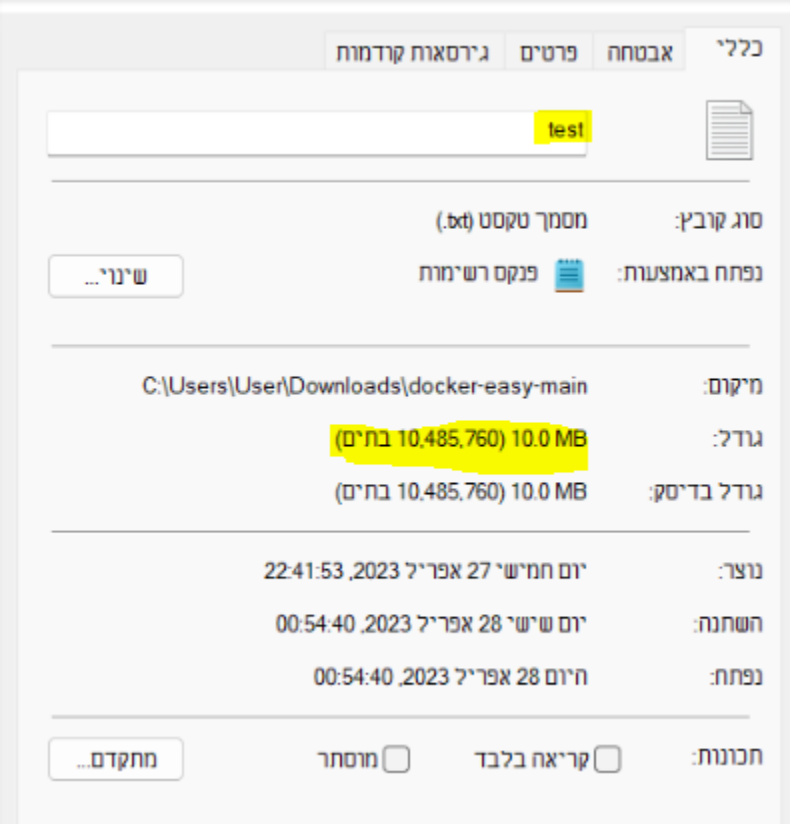
For generating a file, we have created “generate.py”. this short script will create you a file in every name or size

generate.py

```python
import os

filename = "test.txt"
filesize = 10 * 1024 * 1024 # 10 MB

with open(filename, "wb") as f:
    f.write(os.urandom(filesize))
```



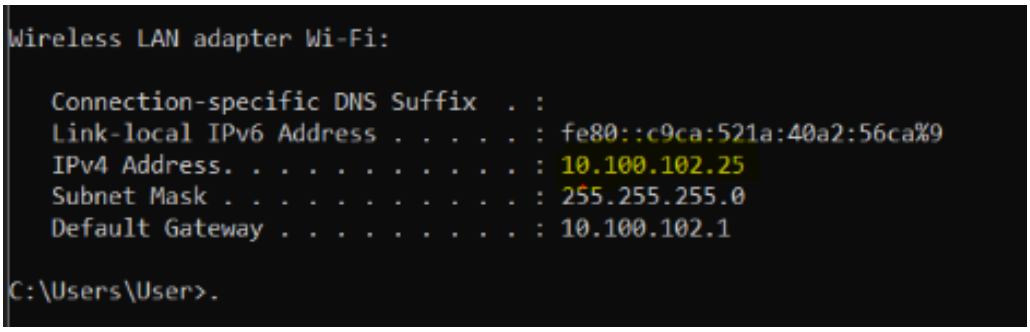
Now, let’s talk about the user itself.

The user connects to the Proxy and starts to send the file in chunks to the proxy over TCP (as requested).

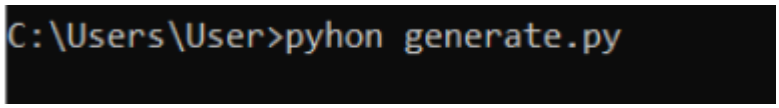
The user also has a progress bar that can be compare to the end-user progress bar to see the delay in the network.

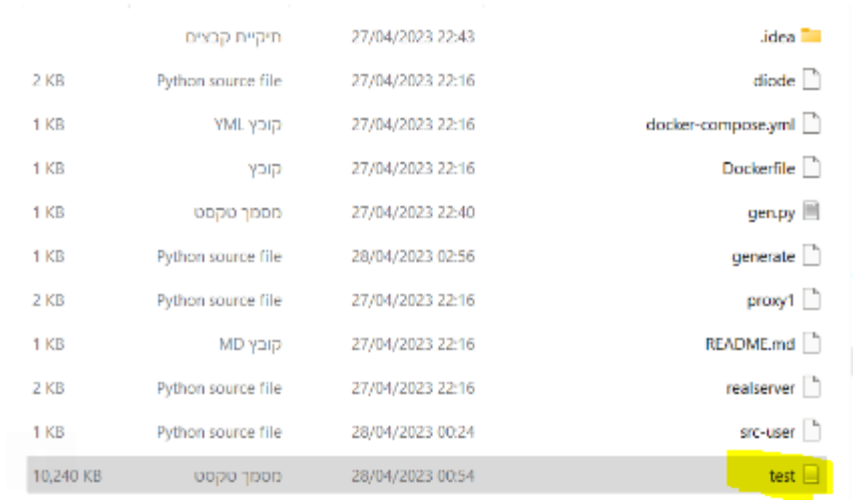
Testing –

You can see your own ip address using ipconfig

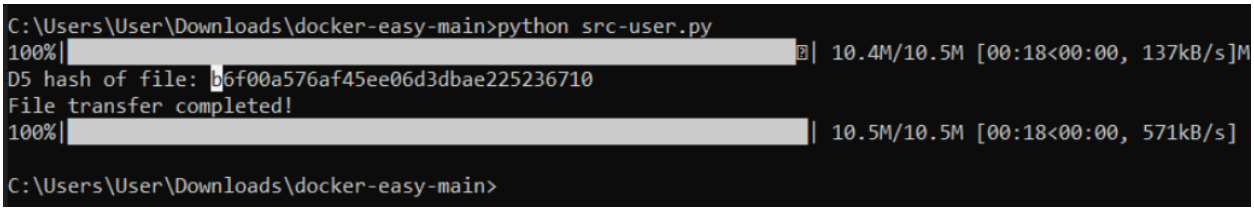


Than use “python generate.py” to generate file to send to the end-user





Than run “python src-user.py”



proxy

The proxy’s purpose is to be like a “middleman” between the “insecure network” to the diode.

It minimizes the attack surface, because then the diode's IP address and network information are hidden from the users. This means that the diode is only can put all of her security efforts between her and the proxy. This makes it harder for an attacker to directly target the end-user.

Enables additional security policies. The proxy can have control of everything that's going to the diode. it can be used to add more security policies such as file scanning, encryption, authentication feature and more..

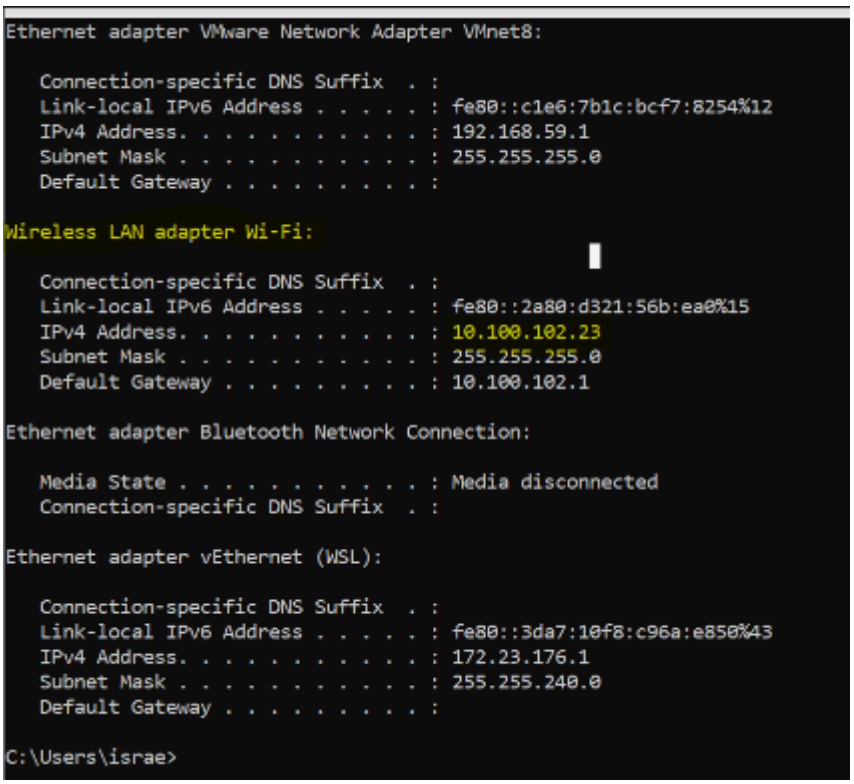
Allows monitoring. It will become easier to monitor and audit the traffic going to diode from the src-user.

Our proxy is very minimal and just passing the information to the diode using TCP sockets.

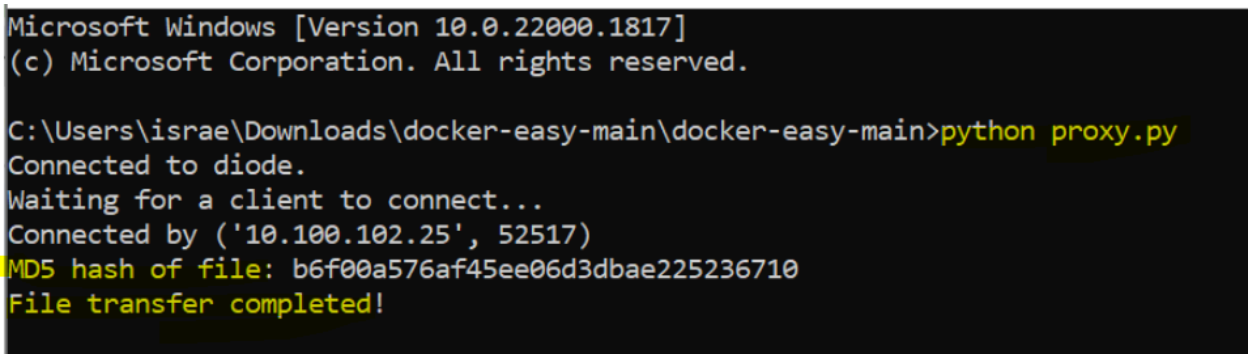
By comparing all the MD5 from all devices you can see that the HASH stayed the same.

Testing –

The proxy ipconfig (for src-user.py)



run “python proxy.py”



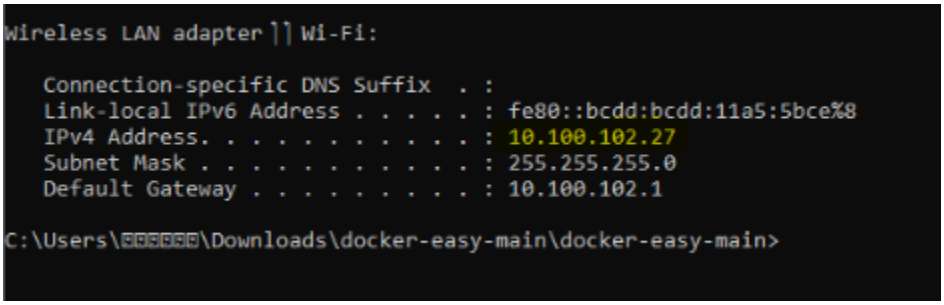
diode

The implementation of the diode ensures a one-directional data flow by opening a TCP socket and listening for incoming data from the proxy on a specific port. When data is received, it is processed and then forwarded to the end-user over a separate

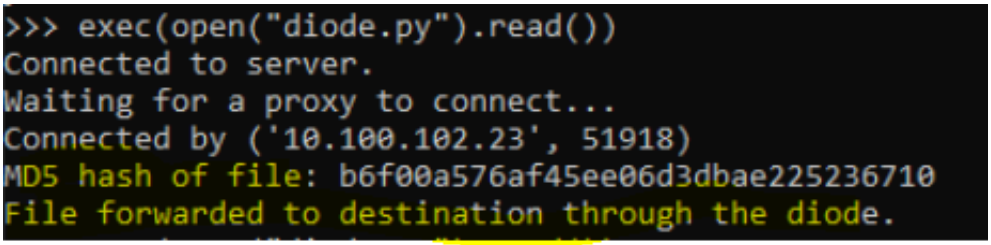
TCP connection. However, there is no communication back to the proxy over the same connection, so the data flow is one-directional.

Testing –

Network configuration (for proxy.py)



run “python diode.py”



end user

the end user is the target!
it has a progress bar to follow the file transmission.

The end-user will create a “received.txt” when done receiving the whole file data.

```
# Write the concatenated data to a file
with open('received_file', 'wb') as f:
    f.write(buffer)
```

The end-user will only receive data from diode.

Testing –

```
Wireless LAN adapter WiFi 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::55d9:daa8:ff02:9280%7
   IPv4 Address. . . . . . . . . . . : 10.100.102.17
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.100.102.1

Ethernet adapter Bluetooth Network Connection 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\magshimim>ipc
```

Change line 22 in end- user:

```
file_size = 10 * 1024 * 1024 #
```

change the size in here according to the size of the file, this is 10 MB.

run “python end-user.py”

```
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\magshimim\Downloads\docker-easy-main>python end-user.py
Waiting for a diode to connect...
Connected by ('10.100.102.27', 51245)
100%|                                                                                | 10.5M/10.5M [01:30<00:00, 228kB/s]
```

at the end you can see that the file md5 is identical (b6f00a576af45ee06d3dbae225236710)

Notes for Testing

The right order to run all files:

1. Python end-user.py
2. Python diode.py
3. Python proxy.py
4. Python src-user.py

Remember also to change the addresses and ports according the ipconfig command

You might need to turn off your firewall settings for this to work.

Questions & Answers

What are the different diode types? How did you implement your own diode? How it works in the communication level?

There are a few types of diodes, you can divide those into 2 groups: network diodes and hardware diodes.

Network diodes are software-based diodes that allow data to flow in one direction only between different network segments. They can be implemented using different devices like: firewalls, proxies, and more.

Hardware diodes are physical devices that allow data to flow in one direction only between two physically separate systems.

In our code, we implemented a network diode using Python and the sockets over TCP. The diode function is responsible for receiving data from proxy

1 and sending it to the end-user. but it does not allow any data to flow back from end-user to proxy. We did it by using separate sockets for receiving and sending data, and by not allowing any feedback from the receiving socket to the sending socket.

The diode works with a one-way flow of data between two network devices. In this case, the data can only flow from proxy1 to end-user, and not the other way around.

What problems can occur due to the unidirectional data flow? What are the drawbacks?

Limited functionality: Security always make things more complicated and less convenient, Since the data flow is only allowed in one direction, certain types of apps may not work, For example, any service that initiates a two-way communication

session through the diode, it will not be possible due to the one-way restriction.

Delayed or lost data: Where real-time data transfer is required, such as the use in skype or zoom than the unidirectional data flow can introduce delays or loss of data. This is because data can only flow in one direction, and if there is a need to send data back in the opposite direction, it will not be possible without breaking the unidirectional flow.

What is air-gapped?

An air-gapped system or network is a type of computer system that is completely isolated from other networks and the internet, often physically disconnected. the system is separated by a physical gap or air gap from other networks, meaning that data cannot be transferred to or from the system through network connections or the internet.

Our solution is unidirectional. Our users want to get telemetry data but still want a high level of security. Can you propose a method to protect the organization while limiting the exfiltration from it? Please suggest a solution.

A possible solution could be to take our current solution for one side and just make a new identical route to the one we have suggested. But the most important thing is to Identify the sensitive data that needs to be protected. Then, you can start protecting it using security policies and rules that constrain the movement of sensitive data.

You can also monitor the organization network and add an encrypted implementation when needed.