Implementation of an Optimized Steganography Technique over TCP/IP and Tests Against Well-Known Security Equipment

Mohamed Tarhda
Laboratory of Electrical Engineering & Energy Systems,
Faculty of Sciences,
Ibn Tofail University
Kenitra, Morocco
tarhdamo@yahoo.fr

Rachid El Gouri
Laboratory of Electrical Engineering and Telecommunications Systems,
ENSA
Ibn Tofail University
Kenitra, Morocco
elgouri.rachid@yahoo.fr

Laamari Hlou
Laboratory of Electrical Engineering & Energy Systems,
Faculty of Sciences,
Ibn Tofail University
Kenitra, Morocco
hloul@yahoo.com

***Abstract*—Nowadays we are witnessing a total convergence towards a digital world where information is digitized, conveyed and processed using highly developed techniques and tools. The development of broadband networks, including the internet, has made easy the manipulation, transmission and sharing of information. However, new security issues arise and they are particularly related to integrity, confidentiality and traceability of data. Facing this situation, network security has become very important and challenges related to the protection of exchanged data over the internet against unauthorized access and use have increased. In the current work, we propose to implement an optimized steganography technique over TCP/IP protocol [1]. We have also tested it against well-known security equipment using latest versions. We will see that they are inefficient to stop this kind of cover channels. Our work is like an alarm to every IT administrator to change their thinking about data lost prevention (DLP) and exfiltration of sensitive information.**

Keywords-steganography; TCP/IP; cover channel; firewall; hidden channel

I. INTRODUCTION

Networks based on TCP and IP protocols are widespread and their uses are endless whether professional or personal. However, while computer security has increased, network security has only been slightly improved. Many security issues are still present, especially in businesses and private networks. Hidden channels remain for the most part a source of unknown risk, and are in fact totally ignored by the public, while they represent a real threat. A hidden channel is defined as a communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy. In general, hidden channel implementations try to leave no trace [2]. Hidden channels pose problems for highly secure environments such as government and military agencies. In multi-tier security environments where users with high levels of security should not be able to deliver information to users with lower security levels, hidden channels can be used to bypass these policies. In a more traditional environment, hidden channels can be used by a malicious user to secretly communicate with a compromised machine, which makes attack detection difficult.

II. IMPLEMENTATION OF COVER COMMUNICATION OVER TCP/IP

A. Basic Idea

The key idea for our implementation is to exploit the TCP sequence filed and the IP identification field in order to insert confidential data that we want to share using a specific encoding, said RADIX-50. This optimized encoding approach allowed us to send more data in less TCP/ IP packets. Here we are considering IPv4 protocol only.

B. Proposed Approach

Although headers are not intended for communication, a sender can encode information in them. Once transmitted over the network, this information may be decoded by a receiver. The IPv4 packet header consists of 14 fields, of which 13 are required. The 14th (options) field is optional. The IPv4 packet header consists of 20 bytes (Figure 1). We choose to exploit properties given by the "Identification" field, which is primarily used for uniquely identifying fragments of an original IP datagram. This 16-bit field contains a random number, generated by the sender and used to identify fragments from the same IP packet. Having this said, we exploit this randomness to hide our useful data into the IP Header. To optimize the flow of the shared information, we can combine the TCP sequence field and the IP identification field to hold the confidential data that we want to share. We use Radix-50 encoding. This allowed us to code three characters in a 16-bit word and so risen data transfer speed [1, 6]. The choice of TCP and IP fields was based on their random characteristic that can make it difficult to detect by an unauthorized person.

C. The Choice of Method

A TCP sender generates an initial sequence number (ISN). This sequence number is used in the communication flow

between the sender and the receiver to keep track of each packet and to ensure that the conversation continues correctly (Figure 2).

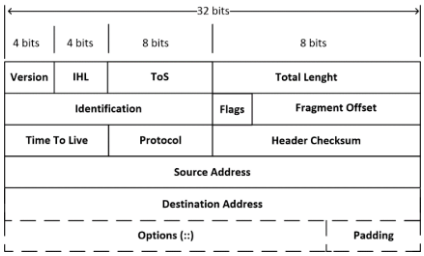


Fig. 1. IPv4 header

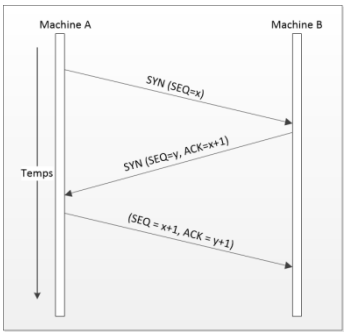


Fig. 2. TCP handshake

The ID field is 16 bit long and represents identification used to reconstitute different fragments. Each fragment has the same ID number. These numbers (ID field) are randomly generated and depend on the platforms in which they are located. Therefore, we focused on the concept of randomization, best suited for information dissimulation.

D. Encoding Radix-50

Message exchange over long distances is a very old need. Claude Chappe invented an optical communication system called the Tower Chappe. This was the first system using a communication standard with the control codes. The first computers (TX-0, TX-2 and the PDP series CED) used memory words of a multiple size of 6, 12, 18 and 36 bits. Various encodings were invented, Radix-50 being one of them. Radix-50 is a method designed for rapid and efficient integer conversion. It converts an integer of any size to a radix-50 alphabet character and vice versa. This library was originally designed for ID generation. Chains are encoded with the first most significant character. For example, the string "ABC" (Codes 11, 12 and 13) is encoded as: ((11*40)+12)*40+13=18093 in a 16-bit word. Possible values are in the range 0-63999. As a result more data are transmitted in less TCP/IP packets. This technique can be useful in the case of transmission of many strings. So, we could get reliably 3 characters in 2 bytes. However, the bad news is that radix-50 is limited to 40 characters (capital letters, numbers 0 to 9, ".", "$", "%" and space) that we could use. Anyway, in a multitude of scenarios, this is not a big problem.

E. Software Environment

Our program is C language based, as this language is a low-level language. It is powerful in the context of network and system programming. It allows manipulation and creation of packages. It works in different platforms: CENTOS, Debian, Windows, etc. For our tests and experiments, we chose VMware virtual platform, which is a leader in virtualization of IT infrastructure. VMware creates a closed environment in which we can create multiple virtual machines representing our test lab. We used VMware ESX that allows a more accurate resource management of each virtual machine and better performance. Our implementation was tested and experimented upon in Ubuntu operating system.

F. Implementation

Our program is written in C, it works in client-server mode, it is executable on Linux platforms, and it is based on sockets for the creation and manipulation of packets. It takes as argument a file containing the confidential message we want to share with the server. Figure 3 shows a part of the code related to Radix-50 encoding for the client mode.

```
//*********************modification/////////////
valeur=recv_pkt. ip. id;
d1=valeur%40;
    valeur=(valeur-d1)/40;
    d2=valeur%40;
    d3=(valeur-d2)/40;

d11=indicev(d1);

d22=indicev(d2);
d33=indicev(d3);


//***************** sequence*********************
valeur2=recv_pkt. tcp. seq;
if(valeur2!=64000){
d4=valeur2%40;
    valeur2=(valeur2-d4)/40;
    d5=valeur2%40;
    d6=(valeur2-d5)/40;

d44=indicev(d4);

d55=indicev(d5);
d66=indicev(d6);
}
else{
d44=0;

d55=0;
d66=0;

}

printf("\t\t %d %d \n",valeur,valeur2);


printf("Receiving Data: %c\n",d33) ;
fprintf(output, "%c",d33) ;
fflush(output) ;


printf("Receiving Data: %c\n",d22) ;
fprintf(output, "%c",d22) ;
fflush(output) ;


printf("Receiving Data: %c\n",d11) ;
fprintf(output, "%c",d11) ;
fflush(output) ;




printf("Receiving Data: %c\n",d66) ;
fprintf(output, "%c",d66) ;
fflush(output) ;
printf("Receiving Data: %c\n",d55) ;
fprintf(output, "%c",d55) ;
fflush(output) ;

printf("Receiving Data: %c\n",d44) ;
fprintf(output, "%c",d44) ;
fflush(output) ;



}
```

Fig. 3. Program C–Radix-50 decoding part in the server mode

The program takes over a number of parameters that can be used for network communication between a client, who sends the data, and a server, the one who receives them.

1) Client Mode

Options for client mode are:

- Dest <IP address>: The destination IP address of the target server.
- source <IP address>: The IP address of the client.
- file <file.txt>: The text file containing data.
- port_source : The source port.
- port _dest : The destination port.

The operations that the client mode performs are:

- Open the file passed as an argument.
- Extract characters of this file.
- Convert these characters in ASCII code.
- Call the encoding function that takes as argument the ascii characters and provides a result radix-50 encoding number.
- Create a socket that will be as a point of communication with the server.
- Create a package and fill in the TCP and IP headers.
- Take the number and encode it in the fields of identification and sequence.
- Send the packet through the network.

2) Server Mode

Options for server mode are:

- Server: This option tells the program to run in server mode.
- Source <IP address>: The IP address of the server.
- file <file.txt>: The file in which the received data will be saved.
- port_source: The source port.

The operations that a server can perform are:

- Get in listening phase on a given port of the machine.
- Receive the package and put it in a "buffer".
- Extract the IP and TCP headers.
- Search identification fields and sequence.
- Retrieve values.
- Call the decoding function that takes numbers as argument and retrieves the letters.
- Put the letters in the file passed as an argument.

G. Communication Flow

Figure 4 describes the communication flow between client and server. The main functions are:

- socket () creates a socket for establishing a communication channel.
- bind () provides a name for the socket descriptor returned by the socket () function.
- send () and the recv () allow respectively to send and receive data through a socket.
- close () releases the communication channel.

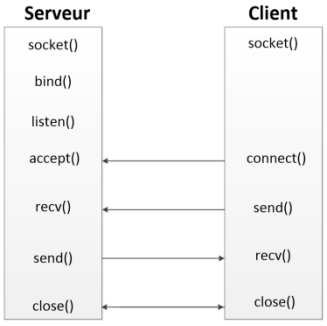


Fig. 4. Client-server communication flow

III. TESTS AND DISCUSSION

A. Test Environment

For our tests, we deployed a virtual environment containing all the information we need. In fact, virtualization of operating systems is a technique to run simultaneously multiple operating systems, as if they were working on separate computers, on a single physical computer. These operating systems are called virtual machines (VMs) (Figure 5).

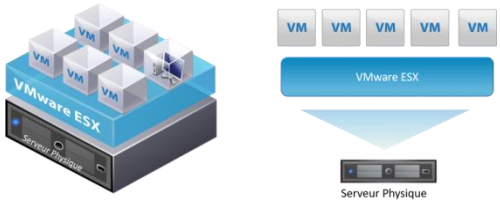


Fig. 5. VMware architecture

Virtualization of operating systems has several advantages:

- The use of a different operating system without restarting the computer.
- Testing under development products and systems without compromising a stable existing environment.
- Software testing in controlled environment, isolated and secure.
- Operating system migration from one computer to another, as virtual machines would work on any computer with a compatible hypervisor.

Having this said, in our case, virtualization allows us to perform simulations using a professional-network-like model.

It allows us also to test our approach of hidden communication via TCP/IP while providing usability and portability. Our platform is based on the VMware environment using the following components: VMware ESXi 6.0.0 Build 3620759, vSphere Client Version 6.0.0 Build 6855219 and VMware vCenter Server version 6.0.0 Build 5112529. VMware is one of the pioneers of virtualization in data centers. It provides customers with a highly available, resilient and flexible infrastructure. Figure 6 shows the two main models used for this section. They include the following machines:

- The two communicating parties: "Ubuntu-VM1" and "Ubuntu-VM2". Their operating systems is Ubuntu 18.04.1 LTS, updated on August 2018. They contain client and server instances.
- Virtual machine "Monitoring-VM" for monitoring and network analysis tools like wireshark and others.
- The firewall virtual appliance Fortigate "FortiGate-VM".
- The firewall virtual appliance "Forcepoint-NGFW".
- The virtual appliance vCenter to manage the VMware environment in a more efficient way.
- The virtual appliance "Forcepoint-SMC-Appliance" to manage the firewall solution forcepoint. It is called Forcepoint NGFW Security Management Center

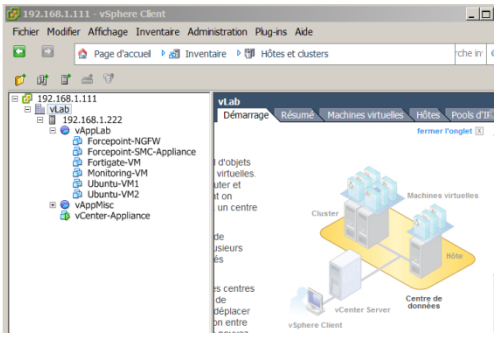


Fig. 6. Models used

First, we run successfully our program of cover channel over TCP/IP in the case of the following scenarios:

- Case 1: The client and server instances are running on the same machine, each program in a separate terminal. The adresses used are the loopback address 127.0.0.1. The client sends a string to the server, which displays the received message and puts it in the output file.
- Case 2: Client and server instances are running on two different machines that are on the same subnet (without active material between them).

In the previous two scenarios, we were able to transfer information to the server successfully. Below we will consider a third scenario, closer to reality, where we put a firewall between the two communicating machines. For testing, we chose two of the best firewalls in the world, which are widely deployed in professional and secure environments. The topology of the deployed network is shown in Figure 7.

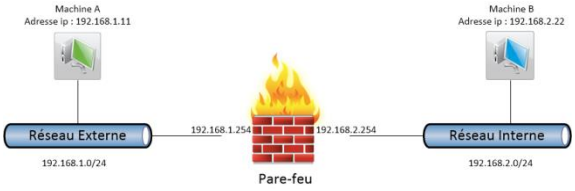


Fig. 7. Network topology for test purpose

B. *Fortigate Firewall*

1) *Presentation*

Fortinet is a leading network security editor [3]. Its products such as UTM Fortigate are high performance solutions adapted to increasingly intensive bandwidth networks and the proliferation of complex and virulent computer threats. In this part of our tests, we used the Fortigate-VM solution, which is a complete Fortigate firewall, presented as a virtual appliance. The Fortigate-VM virtual appliance is ideal for monitoring the virtual traffic on major virtualization platforms, including VMware vSphere, Hyper-V, and Xen. The firewall Fortigate-VM provides intrusion prevention, VPN management, anti-malware protection and other consolidated security features. Note that the appliance version FortiGate-VM is used FortiOS build0163 v6.0.2 (GA), Firmware June 5, 2018.

2) *Configuration*

For this test, Fortigate-VM has three network interfaces, representing the three network segments: Intranet, Extranet and Management. By default, the firewall blocks all connections and allows only connections that were explicitly permitted by the administrator. So far, in our setup we chose to allow HTTPS connections from the machine n°1 to machine n°2 (Figure 8). We also applied all security profiles to their default values (Antivirus, web filter, filter DNS, application control, IPS inspection and SSL inspection). HTTPS protocol was chosen because it is a common used rule and also to show that the parties wishing to communicate secretly could exploit the rules already used in the network. Having said this, our test results are the same regardless of the authorization protocols.

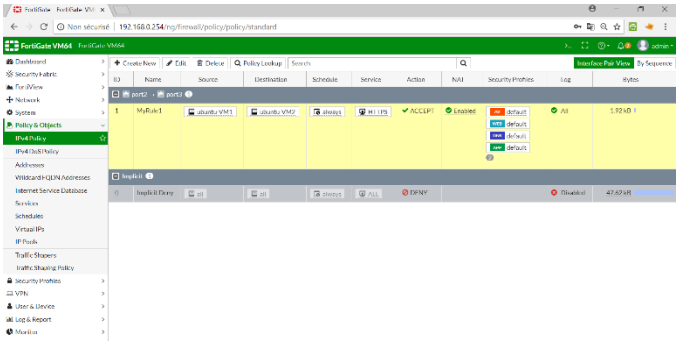


Fig. 8. Fortigate firewall policy setting

3) *Results*

Figure 9 shows that the secret communication between the two machines has been successfully established and the text was sent from one machine to another. Firewall was unable to block it or to log it as hidden channel. In other words, the text was transmitted without a trace.

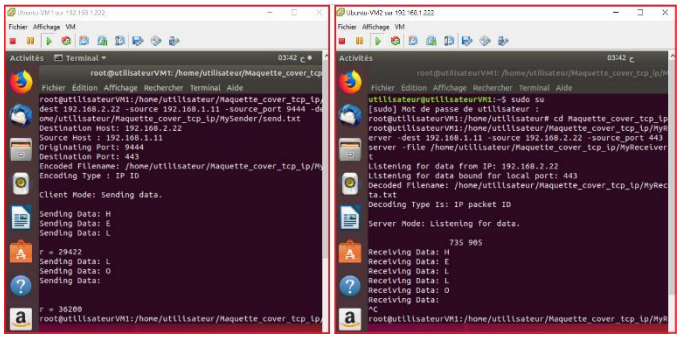


Fig. 9. Illustation of successful secret communication between two virtual machines in a separate subnet

C. Stonesoft NGFW Firewall

1) Presentation

Forcepoint is a leading cybersecurity editor. One of its flagship products is Forcepoint NGFW solution which is an award-winning new generation firewall that blocks malicious attacks and prevents data theft and intellectual property. This technology is increasingly adopted by security companies and organizations:

- The community of Gartner Magic Quadrant 2017 has appreciated the NGFW firewall. The performed analytical tests receive a rating of 95% [4].
- Forcepoint NGFW received the best score among 10 tested editors in the NSS Labs report (June 2017), an independent cybersecurity recognized worldwide, evaluating the effectiveness of security firewalls.
- It is deployed by more and more companies around the world to fight against threats and reduce operational complexity in security systems [5].

2) Configuration

In this test scenario the Fortigate-VM was replaced by the firewall Forcepoint NGFW and simulations were testing hidden communications between the endpoints with the restrictive rules in place (Figure 10).

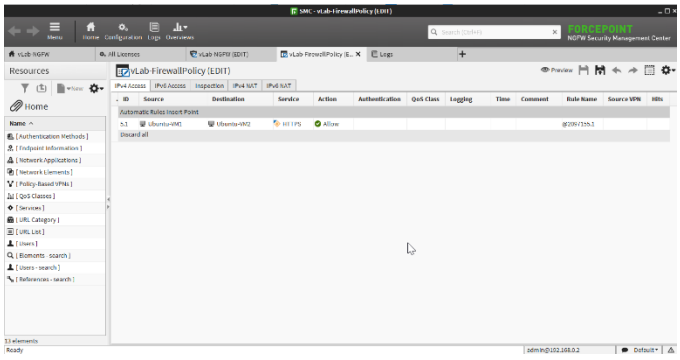


Fig. 10. Forcepoint NGFW firewall rules

3) Results

The communication between the two machines by exploiting the TCP and IP headers has been established, even with all the security and inspection profiles activated in the firewall rule (Figure 11).

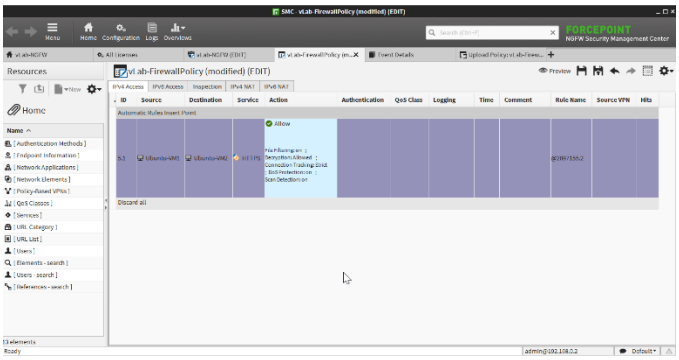
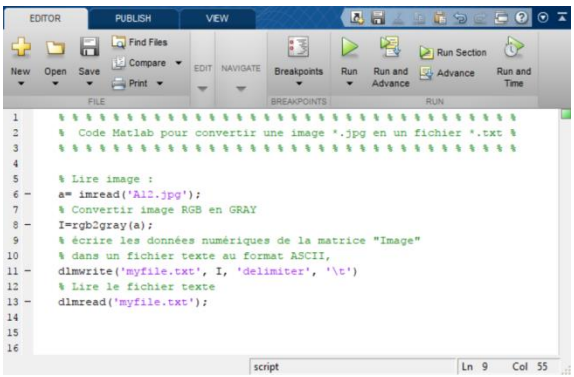


Fig. 11. Forcepoint NGFW firewall rules with all inspection profiles applied

D. Discussion

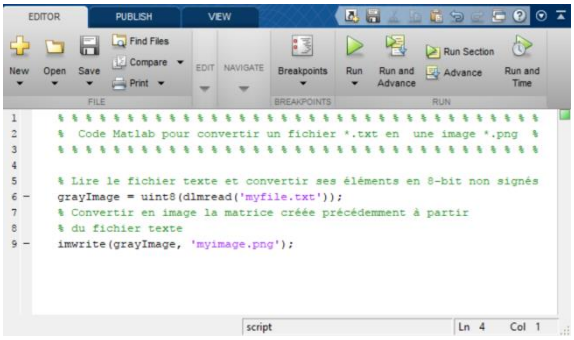
These tests show that the current firewall approach is powerless against the methods of covert channel through TCP/IP protocols. In our tests, we were able to transfer long texts like dozen-pages-meeting reports. Note here that there are programs like MATLAB that are able in a few lines of code to convert an image into text and vice versa (Figures 12 and 13). Hence, we underline the critical danger that can cause the existence of such hidden channels over TCP/IP protocols.



```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%  Code Matlab pour convertir une image *.jpg en un fichier *.txt %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Lire image :
a= imread('Al2.jpg');
% Convertir image RGB en GRAY
I=rgb2gray(a);
% écrire les données numériques de la matrice "Image"
% dans un fichier texte au format ASCII,
dlmwrite('myfile.txt', I, 'delimiter', '\t')
% Lire le fichier texte
dlmread('myfile.txt');
```

Fig. 12. Matlab code lines to convert image to text



```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%  Code Matlab pour convertir un fichier *.txt en  une image *.png  %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Lire le fichier texte et convertir ses éléments en 8-bit non signés
grayImage = uint8(dlmread('myfile.txt'));
% Convertir en image la matrice créée précédemment à partir
% du fichier texte
imwrite(grayImage, 'myimage.png');
```

Fig. 13. Matlab code line to convert text into image

According to our tests, as we can see, the detection of hidden channels is not always possible. Detecting and stopping

covert channel before data exfiltration is even more difficult because we need exchanged data to detect the hidden channel, which would result in the loosing of some data. Having this said, it should be noted that when legitimate users are not looking to establish covert channels themselves, the attacker would compromise communicating machines. Therefore, increasing security in endpoint machines reduces the risk of covert channel, as hidden channels generally require administrator privileges. Security of endpoint is a wide subject to be addressed exhaustively here, but we could give a minimum list of tips to apply in order to reduce the risk of attacks leading to hidden channel and exfiltration of data:

- Workstations should have at least one active antivirus and firewall.
- We should not install nor run any software that it is not digitally signed and delivered from reliable sources.
- Cryptography can be used to encrypt important data and limit their exfiltration.
- Operating systems and software should be updated regularly.
- Implementation of DLP solutions against data leak.
- Implementation of security analysis solutions to monitor events, user behaviors and access to data.
- Users should be aware of security issues.
- Physical access to the machines and the network should be restricted to authorized users.
- Security policy should be defined and applied.

The list is endless, but the application of these tips is already a good step to secure endpoints. An important step for securing machines is to apply the principle of least privilege. Each application, service or user should have access to only what he needs to do his/its work.

IV. CONCLUSIONS

Our implementation is a concrete illustration of the vulnerability of TCP/IP protocols and the robustness of the covert channels over TCP/IP. Furthermore, few applications exist to detect hidden channels and most companies do not try to identify if there is a hidden channel or not in their network. If the hidden channels are difficult to stop, we need to reduce their potential, especially in an environment with strong security requirements. The most effective method is to reduce the bandwidth or close the channel. For example, local area networks can be closed outside working hours.

V. CONTRIBUTION SUMMARY

Although the flaws in TCP/IP protocols exist by design, professionals of network security and the research community do not seem to pay enough attention to them. Thus, we believe that the present work on steganography over TCP/IP adds values as:

- Compared to existing research [1, 6], we have increased the flow of information sent over a TCP/IP packet (three characters instead of one character).
- Our secret communication testing through well-known firewalls and security equipment (with the latest versions and security updates) in professional network model scenarios shows their limitations in detecting hidden channels and in fighting data exfiltration. This is in itself very important because if taken seriously, it could lead to more research to counter this type of vulnerability.

REFERENCES

[1] C. H. Rowland, "Covert Channels in the TCP/IP Protocol Suite", First Monday, Vol. 2, No. 5, 1997

[2] M. Kouremetis, C. West, "Private information retrieval", 16th Annual Information Security Symposium, West Lafayette, IN, USA, March 24-25, 2015

[3] Fortinet, "Customer Reviews on Gartner Peer Insights", available at: https://www.fortinet.com/demand/gated/gartner-enterprise-firewall.html, 2018

[4] Gartner Peer Insights, "Forcepoint-Fortinet Comparison", available at: https://www.gartner.com/reviews/market/enterprise-network-firewalls /compare/forcepoint-vs-fortinet, 2018

[5] V. Satrom, "Forcepoint positionné "Visionnaire" par Gartner dans la Magic Quadrant des firewalls réseau", available at: https://www.forcepoint.com/fr/newsroom/2017/forcepoint-recognized-visionary-gartner-magic-quadrant-enterprise-network-firewalls, 2017 (in French)

[6] S. J. Murdoch, S. Lewis, "Embedding Covert Channels into TCP/IP", in: Information Hiding. IH 2005, Lecture Notes in Computer Science, Vol. 3727, pp. 247-261, Springer, Berlin, Heidelberg, 2005

AUTHORS PROFILE

Mohamed Tarhda works at the Electrical Engineering and Energy Systems Laboratory, Faculty of Science, University Ibn Tofail, Kenitra, Morocco.

Rachid El Gouri is professor at the Electrical Engineering and Energy Systems Laboratory at Faculty of Science. He is also professor at the Laboratory of Electrical Engineering and Telecommunications Systems, National School of Applied Sciences, University Ibn Tofail, Kenitra, Morocco.

Laamari Hlou is the head of the Physics Department at the Faculty of Science, Kenitra. He is also professor and director of the Electrical Engineering and Energy Systems Laboratory, Faculty of Science, University Ibn Tofail, Kenitra, Morocco.