International Journal of Basic and Applied Sciences, 13 (2) (2024) 38-46

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS

Research paper

Towards an architecture for monitoring communications in social networks based on graphs -using honeypot

Kanga Koffi [1] *, Kamagate Béman Hamidja [2] , Brou Aguie Pacôme Bertrand [1], Asseu Olivier [3], Oumtanaga Souleymane [4]

[1] Doctor in computer science: : option: software engineering and database : INPHB doctoral school , Teacher – researcher at ESATIC (African Higher School of ICT: Republic of Ivory Coast) https://orcid.org/0000-0002-5246-4304
[2] Doctor in computer science : network and security option : INPHB doctoral school Teacher - researcher at ESATIC (African Higher School of ICT: Republic of Côte d'Ivoire) Laboratory of Information and Communication Sciences and Technologies , African Higher School of ICT, LASTIC-ESATIC, Abidjan, Côte d'Ivoire, 18bp 1501 Abidjan
[3] UMRI STI, INPHB INP—Houphouët Boigny, Yamoussoukro, Côte d'Ivoire
[4] Computer Science and Telecommunications Research Laboratory, Félix Houphouët-Boigny National Polytechnic Institute, Ivory Coast, Yamoussoukro
**Corresponding author E-mail: kangamiage@gmail.com*

Abstract

In this paper, we are proposing an architecture for monitoring communications in social networks. The main objective is to establish a system that can detect possible attempts to intrude communications in a social network environment using honeypots. To do this, we reviewed the various works related to intrusion detection concerning architectures, algorithms and software tools in this area. Concretely, our proposal includes four (4) components that we have presented while defining the role of each of them. We implemented this architecture in a python environment with the associated algorithms: One-Class SVM - FOREST ISOLATION as well as their combination. The results show that our architecture produces a more refined level of intrusion detection by applying combinations of these different algorithms. Anything to ensure that intrusions detected by honeypots would be reliable using our proposal.

Keywords: *Intrusion Detection; Intrusion Architecture; Honey Pot-Intrusion Detection Algorithm.*

1. Introduction

Talking about communication monitoring in graph-based social networks using honeypots deserves explanation. In fact, monitoring communications in social networks consists of implementing strategies and executing these strategies in order to control the actors and the data exchanged by them. Thus defined, the monitoring of communications in social networks consists of detecting anomalies in the communications of the different actors in these networks. All this, in a social network environment such as Facebook, tweeter, etc. In these networks, the different actors constitute the nodes and the different communications between these nodes constitute the links. Thus defined, the set constituted by the nodes and the links form a graph. Mathematically, Let's G be this set that can be defined as follows:

$G = (\omega, \varphi)$ with $\omega =$ sets of nodes and $\varphi =$ set of links or arcs connecting these nodes (1)

Also, these social networks can be presented as graphs where the nodes represent the users and the edges represent the interactions between them. Furthermore, analyzing these graphs allows us to understand the relationships and structures within this network, such as communities, influencers, reasons for communication, etc.

As for social media monitoring tools [1], these are software, platforms and techniques that allow users to monitor, analyze and measure conversations and activities on social networks. There are different types of social media monitoring tools that can be used for different purposes such as: monitoring [2], analysis [4] and security [5].

In these social networks, we can distinguish 2 types of graphs which are:

- Undirected graphs (figure 1): where the links have no specific direction (for example, 1 and 4 are friends).

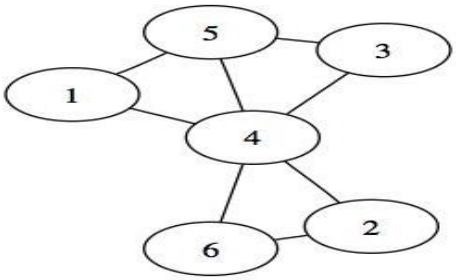

Fig. 1: Undirected Graph.

- Directed graphs (fig 2): where the links have a direction (for example, 3 determines 2).

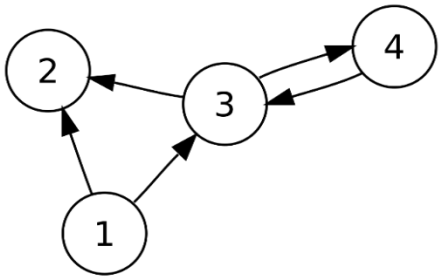

Fig. 2: Directed Graph.

Honeypots are decoy systems or computer resources designed to attract and detect malicious activity. In the context of social networks, a honeypot can be a fake profile or a fake resource attracting cybercriminals or malicious users.
There are several types of honeypots, each with their own advantages and disadvantages. Honeypots can be divided into three main categories [5], which are high interaction, mid interaction and low interaction honeypots. Today with the advent of social networks associated with the high rate of growth in communications between different actors, we are witnessing data exchanges in several forms; ranging from personal data to professional and strategic data. In this context, not securing these communications at the basis of these exchanges could alter the quality of the underlying data. To achieve this, in this paper we aim to propose a tool to contribute to the security of communications in these networks.
The rest of our paper is as follows:

- Section 2 presents the various works to our knowledge relating to the surveillance and/or security of communications in social networks
- Section 3 presents our problem
- Section 4 is devoted to our contribution
- Section 5 has a discussion and we will end with a conclusion while providing perspectives

2. State of the art

The state of the art in social media intrusion detection has evolved rapidly in recent years, with the emergence of new techniques and tools. In this part of our work, we will explore this constantly evolving environment through highlighting intrusion detection architectures, intrusion detection software tools as well as the different intrusion detection algorithms to our knowledge. .

2.1. Intrusion detection architecture [6]

Intrusion detection architectures can be classified into three main categories: distributed architectures, hierarchical architectures and centralized architectures.

2.1.1. Centralized approach (fig 3)

In this architecture, there are two types of nodes: collector nodes and analysis nodes. Collector nodes are IDSs (NIDS/HIDS) that operate locally and report intrusions to the central node or expert node, which correlates the information.
The centralized intrusion detection approach involves collecting security data from different network security devices (such as firewalls, routers, switches, etc.) onto a single central server, where it is analyzed to detect malicious activity. This approach allows for centralized monitoring and efficient analysis of security data, but can also introduce single point of failure risks. Indeed, when the expert node breaks down, the entire system is unusable and security is compromised.

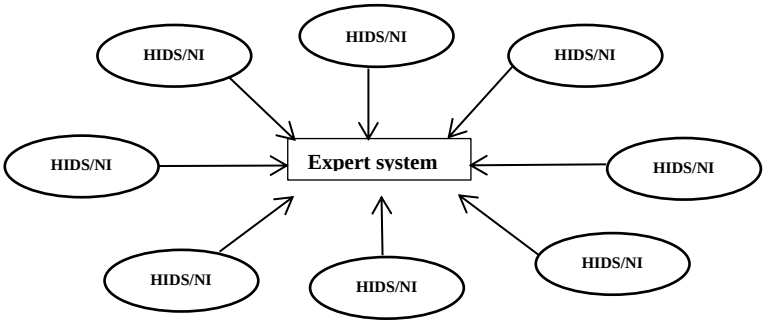

Fig. 3: centralized architecture for intrusion detection.

2.1.2. Hierarchical Approach (fig 4)

To avoid the central node being a SPOF, the hierarchical approach proposes that several nodes are responsible for these correlation operations. The system is broken down into several communication groups. Each group is a subset of the fixed hierarchy. An IDS is designated as a per-group analysis node and is responsible for communicating with higher-level groups.

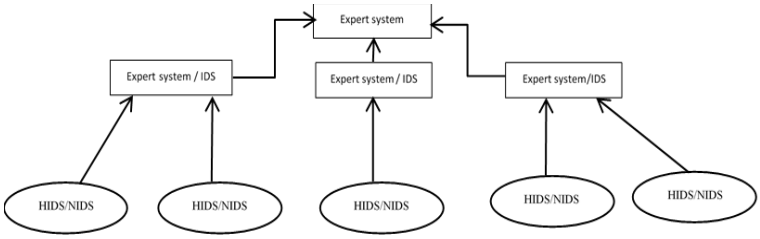

Fig. 4: Hierarchical Architecture for Intrusion Detection.

The figure above represents this approach with 4 communication groups.
The hierarchical approach scales more easily than the centralized approach. However, if the node responsible for communication between two group levels is deactivated, an entire branch of the structure is disconnected from the hierarchy. Finally, high-level nodes are generally inaccurate when it comes to alert correlation because the data sent is usually incomplete or too abstract.

2.1.3. Distributed approach (fig 5)

It is an architecture that aims to reduce the risks of single point of failure present in other architectures. It is based on a structure entirely composed of nodes that are both collectors and analyzers. These nodes detect attacks locally and are able to correlate information from neighboring nodes to detect coordinated attacks.

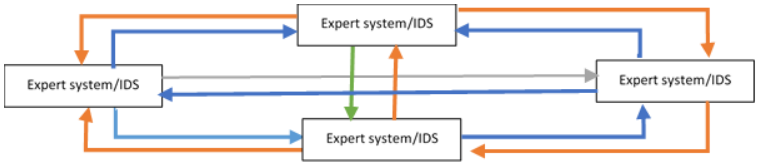

Fig. 5: Distributed Architecture for Intrusion Detection.

Fig 5 shows a network structure comprising several nodes with detection and correlation modules. Nodes detect intrusions locally and can use information from other nodes to complete their analysis.

2.2. Intrusion detection tools [8], [9]

There are multitudes of intrusion detection tools that we cannot make an exhaustive list; however, we will focus on the tools which, according to our various literature reviews, appeared to us to be the most used.

2.2.1. Honeyd

a) Definition

Honeyd is open-source software that allows you to create virtual honeypots (simulated network services) to attract and detect malicious activity on a computer network. It can mimic a range of network services and protocols, giving the appearance of a large network infrastructure when in reality it is a single machine. This makes honeyd a useful tool for security research and also as a security layer to detect and deflect attacks.

b) Honeyd Limits

The Honeyd software has several limitations which are:

- Performance: Honeyd can impact the performance of the server it is running on, especially when simulating a large number of network services.
- Configuration Complexity: Configuring Honeyd can be complex and requires a thorough understanding of network protocols and service behaviors.
- Maintenance: Honeyd requires regular maintenance to ensure that it continues to function properly and collect reliable data.

- Compatibility: Honeyd may not be compatible with all network and server configurations, which could limit its use in certain situations.

2.2.2. Specter honeypot

a) Definition

Specter is a clever honeypot or "deception" system. It simulates a complete machine, providing an attractive target to keep hackers away from production machines. Specter offers common Internet services such as SMTP, FTP, POP3, HTTP and TELNET which appear perfectly normal to attackers, but are actually traps for them to have fun and leave traces without even knowing they are connected to a decoy system, which does nothing that it appears to do, but records everything and notifies the appropriate people. Additionally, specter automatically investigates attackers while they are still trying to break in. Specter provides huge amounts of decoy data and generates decoy programs that will leave hidden marks on the attacker's computer. Automated weekly online updates to this honeypot's content and vulnerability databases allow it to constantly change without user interaction.

b) The limits

The limitations of the Specter Honeypot software are:

- Technical complexity: Deploying and managing such a system can be complex, requiring knowledge of networks, systems and IT security.
- False alerts: Honeypot systems can generate false alerts if legitimate activities occur on the simulated machine.
- Costs: Deploying and maintaining Specter can be costly in terms of time and resources.
- Limited view: A Specter system can only provide a limited view of malicious activity on a network. It cannot detect attacks that are directed at other systems or that bypass the honeypot system.

2.2.3. Back officer friendly (BOF)

a) Definition

It is a security technology used to detect and deflect malicious activity on a network, which can be easily managed and deployed by personnel with limited technical knowledge or resources, typically in a back-office or administrative role. These honeypots are designed to be simple to install and use, and often have a user-friendly interface, so they can be easily managed and monitored by non-technical personnel. The goal of a back-office friendly honeypot is to help organizations improve their security posture by adding an extra layer of security without requiring significant technical resources or expertise.

b) The limits

Although BOF honeypots have some advantages in terms of ease of use and deployment, they also have some limitations that need to be considered. Here are some of the main limitations:

- Limited functionality: Back office-friendly honeypots often have a simplified interface and fewer features than more advanced honeypots. This means they may not be as effective at detecting and deflecting complex or advanced attacks.
- Reduced accuracy: Since back-office honeypots have limited functionality, they may generate false alarms or miss real attacks. This can cause security personnel to be overwhelmed by irrelevant information and miss critical security events.
- Limited scalability: BOF honeypots are typically designed for small to medium-sized networks and may not be suitable for large enterprise networks with large numbers of devices.
- Reliance on vendor support: Since BOF honeypots are typically commercial products, they may need ongoing vendor support to ensure they remain effective and secure
- Cost: BOF honeypots can be expensive compared to other security technologies, especially if they require ongoing support and maintenance from the vendor.

2.2.4. Homemade honeypot

a) Definition

A Homemade honeypot is a type of honeypot that is built and configured by an individual or organization, rather than being purchased as a commercial product. Homemade honeypots can be a cost-effective way to improve security and detect malicious activity on a network, but they also require a certain level of technical expertise and resources to build and maintain. The main advantage of a Homemade honeypot is that it can be tailored to the specific needs of the organization, allowing for greater customization and control of the security environment. However, creating and maintaining a Homemade honeypot also requires a significant investment of time and resources. Also a risk of misconfiguration or other errors could compromise network security.

b) The limits

Homemade honeypot limits depend on various factors, including:

- Efficiency: It may not be as efficient as a commercial honneypot, as it may not have the same level of features and resources.
- Hardware: A Homemade honeypot needs a computer or device with sufficient processing power, memory, and storage to run the software and store data. If you don't have access to a suitable device, building a Homemade could be limited.
- Technical expertise: Setting up a Homemade honeypot requires a good understanding of networking, security and systems administration. If you don't have these skills, then building a Homemade honeypot might be beyond your limits

2.3. Intrusion detection algorithms [12]

There are several algorithms commonly used to detect intrusions in graph-based social networks. Namely:

- Community detection algorithms : This algorithm makes it possible to detect groups of users who have close relationships with each other, and to search for users who do not seem to belong to these groups. These users can be considered potential intruders.
- The Pattern Detection Algorithm : This algorithm makes it possible to search for recurring patterns in relationships between users, and to search for unusual patterns which may indicate an intrusion.
- The Centrality Detection Algorithm : This algorithm makes it possible to measure the importance of users in the network based on metrics such as degree centrality or proximity centrality. Users with high centrality values can be considered potential targets for

intrusions. It is important to note that there is no single solution for intrusion detection in graph-based social networks. It is often necessary to combine several of these algorithms to obtain the best results.

2.3.1. Community detection algorithm

We have different algorithms for community detection in complex networks, each with its own advantages and limitations. The most commonly used algorithms are:

a) Leuven

This algorithm uses a modularity maximization method to detect communities in networks. It is easy to use and implement, but it may yield suboptimal results.

b) Girvan-Newman

This algorithm uses a link cutting method to detect communities in networks. It is more robust than Louvain, but it is more difficult to use and implement.

c) Infomap

This algorithm uses a compression coding method to detect communities in networks. It is effective for large-scale networks, but it is more difficult to use and implement than Leuven.

2.3.2. Pattern detection algorithm

The most commonly used algorithms are:

a) A priori

This algorithm uses a hypothesis generation and regression technique to find frequent patterns in the data. It is efficient for large databases but can be slow for large dataset sizes.

b) Glow

This algorithm uses a depth-first search algorithm to find frequent patterns in the data. It is faster than the Apriori algorithm but can consume more memory.

c) FP-growth

This algorithm uses a data compression technique to accelerate the detection of frequent patterns. It is faster and less memory intensive than the Apriori and Eclat algorithms.

2.3.3. Centrality detection (measurement) algorithm

These algorithms allow you to know if a node in the graph that constitutes a social network is important. As a centralized detection algorithm we can list:

a) Degree centrality

This algorithm measures centrality based on the number of links or edges that enter or exit a node. Nodes with a large number of links are considered the most important.

b) Centrality of proximity

This algorithm measures centrality based on the distance between nodes. Nodes that are close to each other are considered the most important.

c) Centrality of betweenness

This algorithm measures centrality based on how often a node is on the shortest paths between other nodes. Nodes that lie on many shortest paths are considered the most important.

d) Centrality of Eigenvector

This algorithm measures centrality based on the number and centrality of nodes to which a node is connected. Nodes that are connected to very important nodes are considered important.

3. Problematic

The various existing works in the literature, to our knowledge are related to intrusion detection architectures, intrusion detection tools and also to honeypots used to attract possible intruders. After an analysis of these different elements (tools and work), it appears that they have been established in classic network contexts. Faced with this observation, there is a reason to think over the question of communications surveillance in the context of a social network connecting humans, software and hardware. So, the question that arises from this is: how to monitor communications in a social network? In other words, what graph-based tools for monitoring communications in social networks? This is the question that our contribution will focus on providing solutions.

4. Contribution

To make our contribution to the question raised by our research problem, we are proposing an architecture called HONEYSYS. This architecture is made up of four elements:

- A social network
- A component managing honeypot deployment
- A monitoring and data collection component
- A component for analyzing data collected by component 3

4.1. presentations of the components and their operation

Schematically, our architecture is as follows (fig 6)

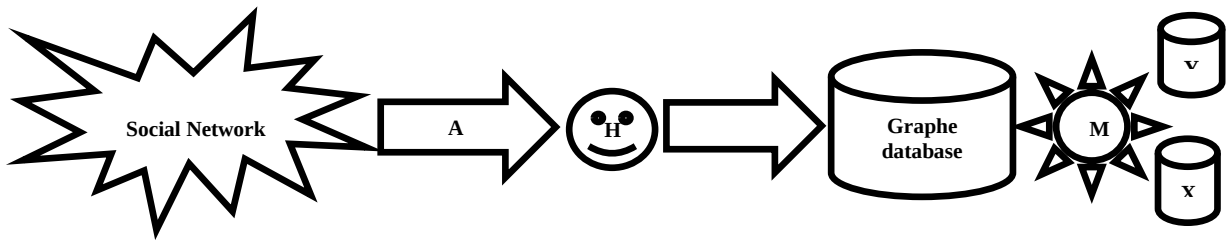

Fig. 6: Diagram of Our Architecture.

In this diagram, we have the following elements:
Social network: this element represents the social network used by our architecture. It could be chosen from the following social networks: Facebook, tweeter, YouTube, etc.
A : represents the different communication monitoring links carried out by the honeypot on the network
H : represents the honeypot carrying out the monitoring
B : represents the connection links to the database ,that allows the various information collected by the honeypot to be stored in H
Database graph : represents the database that stores the data collected by the honeypot. In our case, this database is a graph-oriented nosql database. We chose it because our architecture is implemented in a graph-oriented environment.
M : represents the analysis engine for the data stored in the database, in order to detect possible data resulting from intrusion or attempted intrusion.
X, Y ; represents the different data groups according to their centrality. So depending on the centrality of this data, our architecture performs an automatic density classification so as to know if this data comes from intrusion or intrusion attempts.

4.2. Implementation of our architecture

The implementation of our architecture follows the following steps:
Step 1: Social Network Modeling

- Here we created a graph model of the social network by capturing the nodes and edges. To do this we used tools such as:
- NetworkX from Python: for constructing the graph
- Gephi: An interactive graph visualization and analysis tool

Step 2: Deploying Honeypots
Our architecture being based on honeypots which are lure tools, we created fictitious profiles or lure resources (decoy sculpin) on our social network.
In our case we implemented a technique (honeyhash) that allows the stealing of identification information on a social network (technique used in the detection of Pass-the-Hash attacks).
Step 3: Monitoring and Data Collection
Here we carry out a monitoring of all the nodes and messages exchanged by these nodes in order to collect all the forms of data about users, types of interaction, frequency and patterns and nature of behavior of network elements
Step 4: Data Analysis and Viewing Problematic Data Sources
In this part, we use anomaly detection algorithms to identify suspicious communication behaviors. In our case we use the following algorithms:

Table 1: List of Algorithms Used and Their Role

Algorithms	roles
One-Class SVM	Detect instances that differ significantly from data deriving from the same social network. Given that data from the same networks have common characteristics
Forest Insulation	Isolation Forest is used specifically to identify data sources that stand out significantly from the rest of the social network dataset in our architecture, i.e. anomalies. These anomalies may be fraud, errors, or any other type of rare or unusual observation.

- One-Class SVM (Support Vector Machine)[16]

The goal of One-Class SVM is to find a decision function that is positive for most of the training data (i.e., "normals") and negative for points that are considered anomalies.
This algorithm uses 2 functions:

- A decision function for a data source defined by:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i K(x_i, x) - \rho\right) \tag{2}$$

With
X= data source
X_i = the support vectors, which are the data sources closest to the decision boundary
α_i = Lagrangian coefficients associated with the support vectors, determined by the optimization
ρ= a threshold parameter (offset) learned during training, which determines the position of the decision boundary

- an optimization function defined by:

$$\min_{\mathbf{w},\xi,\rho} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N} \xi_i - \rho \tag{3}$$

Under the constraints

$$\mathbf{w} \cdot \phi(x_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, 2, \ldots, N \tag{4}$$

Where
W: the weight vector (in the transformed feature space).
ϕ(xi): the nonlinear transformation applied to the data (induced by the kernel).
ξi: slack variables allowing data sources to be inside the boundary (anomaly tolerance)
v: a regularization hyper parameter controlling the proportion of data considered as anomalies (higher values no longer allow anomalies).
N=: the total number of data sources.
The implementation of this algorithm associated with our architecture gives the result of Fig 7.

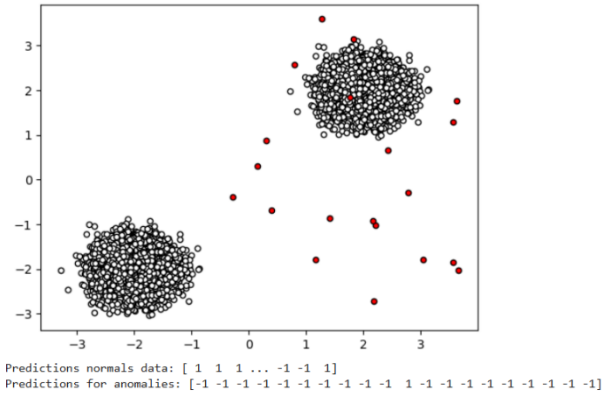

Fig. 6: Result of One Class – SVM.

Result = 20 anomaly source data (in red)
Normal data = 9980
Total number of data = 10000
In Fig 7, the data sources for which:
f(x) > 0 are considered as normal sources (in black color).
F(x) < 0 are considered an anomaly (therefore vulnerable to color intrusion (red)

- Forest Isolation (allows you to measure the local anomaly of a data point) [17]

The anomaly score for a data source in a dataset of size is given by:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{5}$$

With
E(h(x): Average depth of through trees or sources of insulation.
C(n): Normalization factor based on a data set of n size in such as :

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{6}$$

Where H(i) is the I^{th} harmonic number, approximated by ln(i)+ϒ (with ϒ being the Euler-Mascheroni constant, approximately equal to 0.577).
If s(x,n) tends to 1 therefore
The data source is most likely an anomaly, because it was isolated quickly (low isolation depth).
Otherwise, if s(x,n) tends towards 0.5 or zero therefore
The source is probably normal because it was isolated at a depth similar to the average normal spot.
If s(x,n) < 0.5 therefore
This would indicate that the data source is in a dense area, very unlikely for an anomaly.
The implementation of “isolation forest” in our architecture gives the result of Figure 10.

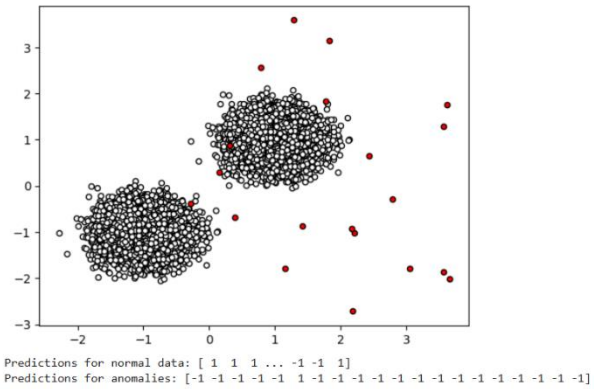

Fig. 7: Forest Insulation Result.

Result = 20 anomaly source data (in red color)
Normal data = 9980
Total number of data = 10000
An Analysis of graph (network) structures to spot unusual communication patterns or clusters of malicious activity.
Simulation result
Table 2 presents the simulation results of the different algorithms but also of the combination of these algorithms together. The objective of this combination is to propose an algorithm allowing better monitoring of communications to be applied to our architecture. In our case, the second algorithm of the combination is applied to the results of the first (for example, the B is applied to the "A" to give the result of "A+B"). The result obtained makes it possible to certify that 0.14% of A are really probable sources of intrusion.

Table 2: Results of Intrusions by Algorithm and Algorithm Combination

Algorithm	Number of sources	Source intruders	Rate
One-Class SVM (A)	10000	20	0.2
Forest Insulation (B)	10000	20	0.2
A+B	10000	14	0.14

Fig 9 shows the different intrusion detection rates presented in the table. However, the results of these 'A+B' combinations make it possible to certify that the results of previous detections are sources of anomalies.

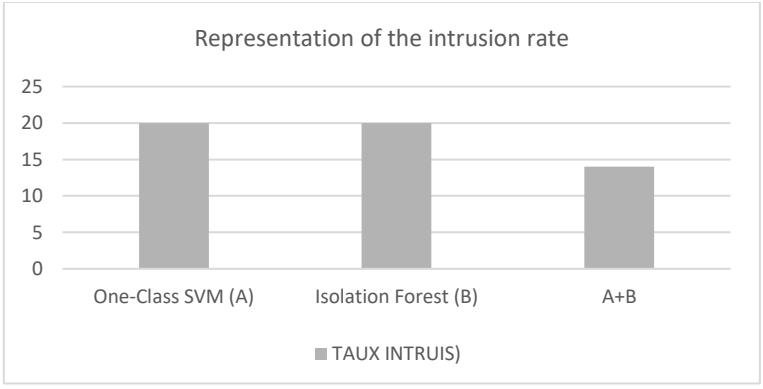

Fig. 8: Presentation of the Intrusion Rate by Algorithm.

5. Discussion and conclusion

In this work, we presented an architecture for monitoring communications in social networks. We presented the different components of this architecture, their roles and operation. We applied different algorithms to our architecture as well as a combination of these algorithms. The results produced by this combination allow us to certify that the latest results produced certify the level of intrusion in communications in social networks.
However, our architecture is faced with some challenges . These are :

- Consideration of ethics and confidentiality because the context of surveillance, the data to be monitored and the results of surveillance must comply with laws on the protection of personal data and privacy
- The effectiveness of Honeypots: Honeypots must be convincing enough to attract social media users without arousing their suspicions.
- Data analysis: algorithms must be adapted to process large quantities of data and effectively detect anomalies.

6. Conclusion and perspectives

The objective of this work was to propose an architecture for monitoring communications in social networks. This surveillance was intended to detect intrusions into these networks. To do this, we reviewed the various works related to intrusion detection architectures , intrusion detection software tools as well as the different intrusion detection algorithms to our knowledge. From this work we have

identified the various limits and have made our contribution, the results of which make it possible to certify that our architecture applied to different algorithms gives certainty of intrusion as to the results of the first algorithms applied (One-Class SVM - ISOLATION FOREST).
Our future works and contribution could focus on the use of deep learning tools and auto encoders for intrusion detection in complex data sources.

References

[1] https://blog.netwrix.fr/2018/11/21/les-10-meilleurs-outil-logiciels-de-surveillance-de-windows-server/.
[2] https://www.blesk.ca (accessed 08/04/2024)
[3] https://netbasequid.com/blog/20-free-social-media-analytics-tools-fr/.(accessed 04/08/2024)
[4] https://www.zoho.com/fr/social/social-media-monitoring.html. (accessed 04/08/2024)
[5] https://www.fortinet.com/resources/cyberglossary/what-is-honeypot. .(accessed 08/05/2024)
[6] Riquet, D. (2015). Discus: A distributed network intrusion detection architecture based on a dedicated language (Doctoral dissertation, University Lille 1-Sciences and Technologies). https://hal.science/tel-01757859/document.
[7] Bouzayani, H. (2012). Quantitative model for intrusion detection: an IDS-HONEYPOT collaborative architecture (Doctoral dissertation, Université du Québec en Outaouais). https://di.uqo.ca/id/eprint/508/1/Bouzayani_Hatem_2012_m%C3%A9moire.pdf.
[8] Curran, K., Morrissey, C., Fagan, C., Murphy, C., O'Donnell, B., Fitzpatrick, G., & Condit, S. (2005). Monitoring hacker activity with a Honeynet. International Journal of Network Management , 15 (2), 123-134. https://doi.org/10.1002/nem.549.
[9] Moore, C. (2016, August). Detecting ransomware with honeypot techniques. In 2016 Cybersecurity and Cyberforensics Conference (CCC) (pp. 77-81). IEEE. https://doi.org/10.1109/CCC.2016.14.
[10] Kemppainen, S., & Kovanen, T. (2018). Honeypot utilization for network intrusion detection. Cyber Security: Power and Technology , 249-270. https://doi.org/10.1007/978-3-319-75307-2_15.
[11] Lee, S., Abdullah, A., & Jhanjhi, NZ (2020). A review on honeypot-based botnet detection models for smart factories. International Journal of Advanced Computer Science and Applications , 11 (6). https://doi.org/10.14569/IJACSA.2020.0110654.
[12] Pierrot, D., Harbi, N., & Darmont, J. (2018, May). Intrusion detection and decision support. In 12th Conference on Advances in Decision Systems (ASD 2018) . ACM. https://hal.science/hal-01761914/document.
[13] https://www.ripublication.com/irph/ijict_spl/ijictv3n10_02.spl.pdf (accessed 05/08/2024)
[14] https://www.honeyd.org/. (accessed 05/08/2024)
[15] Tiv, M., Gullifer, J.W., Feng, R.Y., & Titone, D. (2020). Using network science to map what Montréal bilinguals talk about across languages and communicative contexts. Journal of Neurolinguistics , 56 , 100913. https://doi.org/10.1016/j.jneuroling.2020.100913.
[16] Nguyen, T. Q. (2023). Unsupervised machine learning for the detection of illegitimate traffic (Doctoral dissertation, Paul Sabatier-Toulouse III University).
[17] Meriem, K., & Cérine-Maroua, B. An Improved K-means Clustering Algorithm.
[18] Jabiri, F. (2020). Applications of unsupervised classification methods to anomaly detection.
[19] Elmahalawy, A.M., Mousa, H., & Amin, K. (2022). A Comparative Study for Outlier Detection Strategies Based On Traditional Machine Learning For IoT Data Analysis. IJCI. International Journal of Computers and Information , 9 (1), 60-73.
[20] Barbariol, T. (2023). Improving Anomaly Detection for Industrial Applications.