Research Article

Lounis Ouarda*, Bourenane Malika, Nacer Eddine Yousfi, and Bouderah Brahim

Improving the efficiency of intrusion detection in information systems

https://doi.org/10.1515/jisys-2022-0059
received April 05, 2022; accepted May 05, 2022

Abstract: Policy Interaction Graph Analysis is a Host-based Intrusion Detection tool that uses Linux MAC Mandatory access control policy to build the licit information flow graph and uses a detection policy defined by the administrator to extract illicit behaviour from the graph. The main limitation of this tool is the generation of a huge signature base of illicit behaviours; hence, this leads to the use of huge memory space to store it. Our primary goal in this article is to reduce this memory space while keeping the tool's efficiency in terms of intrusion detection rate and false generated alarms. First, the interactions between the two nodes of the graph were grouped into a single interaction. The notion of equivalence class was used to classify the paths in the graph and was compressed by using a genetic algorithm. Such an approach showed its efficiency compared to the approach proposed by Pierre Clairet, by which the detection rate obtained was 99.9%, and no false-positive with a compression rate of illicit behaviour signature database reached 99.44%. Having these results is one of the critical aspects of realizing successful host-based intrusion detection systems.

Keywords: host-based intrusion detection system, PIGA, system calls, genetic algorithm, equivalence class

1 Introduction

To respond to the security problem, a security policy must be defined according to the information system we wish to secure and the security objectives we wish to achieve. This policy expresses the confidentiality, integrity, and availability properties that the information system must respect. Intrusion Detection Systems (IDS) refer to a hardware/software platform for monitoring network or system activities to detect malicious activities.

There are two main types of IDS: Host-based IDS (HIDS) and Network-based IDS (NIDS). HIDS is characterized by analysing events/traces generated by the system (e.g. analysis of system calls) [1]. NIDS analyses the data passing through the network [2]. From a methodological point of view, IDS can also be classified into two classes: one that seeks to detect anomalies (this is known as a behavioural approach) and one that seeks to detect malware (this is known as the misuse approach) [2].

A behavioural approach can detect attacks that are still unknown at the time of modelling. However, building such a detection model can be difficult, and the modelling may lead to false alarms. Misuse detection allows the detection of a known attack along with the definition of a scenario. This approach uses a knowledge base called an attack signature base and a pattern search method to recognize the defined signatures.

*** Corresponding author: Lounis Ouarda,** Computer Science Department, Industrial Computing and Networking Laboratory-RIIR, University Oran 1, 31000, Oran, Ahmed Benbella, Algeria, e-mail: lounis.ouarda@edu.univ-oran1.dz
Bourenane Malika: Computer Science Department, Industrial Computing and Networking Laboratory-RIIR, University Oran 1, 31000, Oran, Ahmed Benbella, Algeria, e-mail: bourenane.malika@univ-oran1.dz
Nacer Eddine Yousfi: LSV Specification and Verification Laboratory, Computer Science Laboratory of ENS Paris-Saclay (CNRS), Paris, France, e-mail: neyousfi@yahoo.com
Bouderah Brahim: Computer Science Department, University of M'sila, M'sila, Algeria, e-mail: Brahim.bouderah@univ-msila.dz

Open Access. © 2022 Lounis Ouarda *et al.*, published by De Gruyter. This work is licensed under the Creative Commons Attribution 4.0 International License.

However, it does not allow the detection of unknown attacks, which results in a high false-negative rate and requires active maintenance of the signature base.

In practice, IDS based on behavioural approaches are not very secure. Furthermore, signature-based intrusion detection does not guarantee any security properties. The objective is simply to detect known violations. Therefore, we can deduce that these intrusion detection solutions do not consider the sequences, and they are limited to the control of interactions and do not deal with complex activities. Moreover, these solutions are oriented towards the detection of attacks, so instead of analysing each attack separately, knowing that there are multiple types and several variants of these attacks with a specific signature for each type of attack, why not try to seek the origin of these attacks to ensure that the security properties reached by these attacks will not be compromised (we are referencing solutions oriented towards the effects of attacks). Hence, the idea of formalizing system security properties was treated by the authors in ref. [3–5].

Authors in ref. [3] propose Policy Interaction Graph Analysis (PIGA) HIDS, a tool for detecting malicious behaviour by a system trace analysis. To do so, it uses signatures representing illicit behaviours. These signatures are generated in pre-processing and are used in the detection process. However, the signature base is large for a real system and requires significant memory for the detection process. For example, for a complete Fedora system with a graphical interface, the signature database is over 500 MB [5].

Motivated by this shortcoming of PIGA, this article is suggested a way to reduce the memory needed to store signatures while preserving the signature quality. For that, specific contributions can be summarized as follows:

- The proposed mathematic model is original and not limited to PIGA HIDS compression signatures but also applicable to other types of databases like NSL-KDD.
- The proposed model applies to both oriented and non-oriented graphs, unlike Pierre Clairet's model, which applies to a non-oriented graph.
- The proposed model applies to both Briffaut's [3] and Cornabas' [4] graphs; the only difference is that the use of the first graph implies the signatures of illegal behaviour, and the use of the second graph implies the signatures of legal behaviour
- As far as we know, this is the first time that an optimization method has been proposed on PIGA HIDS, taking into account the detection rate, the false-negative rate, the false-positive rate, and the signature compression rate. However, the authors in ref. [5] studied just the compression rate.

In the rest of this article, more details of the approach are explained, and experimental results are presented to demonstrate the effectiveness of the proposed method.

2 Related works

Several types of research were carried out in the field of intrusion detection. Those attempts examined different techniques to design an efficient system that can detect malicious events in real-time. Those approaches include deep learning, machine learning [6], neural network [7], and last and not least, research that used a formal model for the formalization of security properties [3–5].

In this section, some practical approaches proposed by researchers during the past decade are reviewed, especially those that allow analysing system calls. Besides, the achievements and limitations of each of those works are highlighted in Table 1.

As stated above, HIDSs have been very successful in addressing the security needs of the system. However, most of the work done in this area remains difficult to compare because each has its approaches, formalisms, and prototypes. Other researchers are oriented towards a new way of researching new paradigms, for that we can say:

> "*The merging and proper use of the principles and concepts of security mechanisms* (*access control and intrusion detection*) *can create a new powerful approach, capable of compensating for their limitations. In this context, we found two important works, those of Jeremy Briffaut and Pierre Clairet*."

Table 1: Comparison with the previous approach to HIDS

Article	Data set	Technique	Strength	Weakness	Result
[5]	Information flow graph, which represents the set of information transfers authorized by an access control policy PIGA HIDS	(1) The use of modular decomposition on the PIGA HIDS graph to obtain a modular quotient graph (2) Deletion by inclusion; it deletes all redundant signatures from the signature base	Shows high compression signature database rate	None of the results is given about detection rate, false-negative and false positive	Compression signature database rate is 98.61%
[6]	ADFA-LD	Unsupervised machine learning-based system anomaly detection framework using (RNN-AE) and (RNN-DAE)	The best result was recorded at 0.8828 in terms of area under the receiver operating characteristic curve (AUROC)	The system call traces only reflect some parts of users; other attributes can be considered, such as system call, API information, and multiple artefacts considering network, registry, and process	0.8828 in terms of AUROC
[8]	ADFA-LD ADFA-WD	Analysis based on Tfidf values of n-gram terms along with dimensionality reduction using truncated SVD	(1) Higher detection rate and accuracy (2) Computation efficient As it uses small Sized feature vectors for analysis	Marginal overhead is Involved in computing the Tfidf values of n-gram terms for performing dimensionality reduction	0.962 in precision, 0.953 in the recall, and 0.038 in the false-positive rate
[7]	NSL-KDD, CICIDS2017, ADFA-LD, and ADFA-WD	Convolutional Neural Network	Proposed HIDS for CICIDS2017 has the highest performance with 99.29% ACC (accuracy), followed by ADFA-LD with 95.34% ACC and NSL-KDD with 83.43% ACC. Finally, ADFA-WD with 77.01% ACC	—	99.29% Best detection accuracy for the CICIDS2017 dataset

Our work is interested in compressing the PIGA HIDS signatures database by considering the detection rate, false positive, and false negative.

3 The overall description of the problem

We consider a system execution trace containing a set of system calls; each system call defines an interaction between two system entities through an elementary operation of type *read, write, transition* or *execution*. Furthermore, we consider the date and time of each interaction's beginning and end; this information is necessary to enumerate the indirect information flows. In fact, the principle of causality must be respected in the indirect information flow to detect the interaction sequences representing a forbidden activity. Finally, we are interested in optimizing host-based intrusion detection in the static case. To accelerate this detection, we apply minimization on the information flow graph; our approach is described through the following two phases:

3.1 Interactions grouping

To do this, we first define groups of interactions and thus reduce their number; then, we consider all interactions in the execution trace before forming the flow graph. After that, we classify them into groups; each group contains the interactions with the same source and target security contexts. Therefore, all interactions with the same elementary operation will be combined into one interaction with the same elementary operation type and the smallest start date and time, and the largest end date and time.

Sample: be the following interactions:

$$\text{user1} \rightarrow^{r}_{\text{Jan 02 (12:10:12, 12:12:15)}} \text{file3}$$

$$\text{user1} \rightarrow^{r}_{\text{Jan 02 (13:21:12, 13:58:01)}} \text{file3}$$

$$\text{user1} \rightarrow^{r}_{\text{Jan 02 (14:11:10, 14:15:26)}} \text{file3}$$

$$\text{user1} \rightarrow^{r}_{\text{Jan 02 (14:17:15, 15:20:00)}} \text{file3}$$

These four interactions are grouped into the interaction: $\text{user1} \rightarrow^{r}_{\text{Jan 02 (12:10:12, 15:20:00)}} \text{file3}$ having the same source and destination and Jan 02 12:10:12 as the smallest start date and time Jan 02 15:20:00 as the largest end date.

3.2 Size reduction of the information flow graph

To remedy the problem of false negatives encountered when using the deletion by inclusion in the work of Clairet et al. [5], we define another solution in the second step that allows the size reduction of the information flow graph. An information flow is either a path in this graph or a composition of several paths. First, we define the notion of equivalence classes; each class contains all the possible paths having the same source and destination; thus, each class is considered to be decomposed into equivalence subclasses. Each subclass contains equal length paths. Then, we enumerate all the possible paths between two vertices of the graph. We minimize them to obtain one or more signatures using an algorithm that takes into account, at the same time, the capacity of the signatures obtained to represent the maximum of paths. Therefore, its capacity to detect the maximum of illicit behaviours and the error rate that these signatures can cause hence the number of false negatives and false positives obtained. These two notions are essential because they will be used in our solution to determine the percentage of accuracy of each signature.

This solution allows us to minimize the flow graph, which reduces memory consumption while testing the signatures' effectiveness and therefore keeps the IDS very efficient.

In our proposed solution methods, we consider these hypotheses:

1) We define the notion of equivalence classes; each class contains all paths with the same source and destination and is composed of several equivalence subclasses. All paths in a subclass have the same length, which is defined by the number of arcs in the path.
2) The enumeration of all paths between two vertices of the graph must be represented by one or more signatures compressing the paths.
3) The set of all signatures must have the same decision, which is assigned by the initial graph to each intercepted and tested system call.
4) The compressed paths in the form of signatures must have a maximum detection rate and minimum false positive/false negative rates.
5) These signatures must be stored with a minimal memory space, which must not exceed the space of the initial graph.

Therefore, the objectives and the constraints of our work can be summarized as follows:

- minimize the CPU load and memory consumption;
- increase the detection rate;
- minimize the false positive and the false negative rates;
- no addition of information in the new system; and
- no deletion of information from the new system.

Accordingly, our proposed approach covers the following points:

Hybrid: this means combining both the behavioural and signature approaches. In fact, we will benefit from the behavioural approach while limiting its drawbacks by using the signature approach. Most of the previous works have shown the usefulness of such a combination.

Performance/Efficiency: the proposed approach guarantees the system's performance via reducing the memory consumption and the CPU load of the IDS while maintaining its efficiency by providing a high detection rate and a minimum false positive/negative rate. Guaranteeing the efficiency of HIDS with a low impact on the system's performance is the main challenge. Therefore, it comes down to resolving a problem with two contradictory objectives. This point is manifested by using heuristics to keep high-quality signatures only.

4 Solving our optimization problem in the offline case

The offline case represents the case where the system calls are already collected in a system trace before the beginning of the analysis operation. We distinguish three phases running sequentially:

- First phase: The construction of the information flow graph is done from an SELinux security policy [3].
- Second phase: In which our information flow graph minimization solution is applied.
- Third phase: The effectiveness of our intrusion detection solution present in the system call trace is tested.

4.1 Mathematical modelling

Our model is based on a graph, which highlights the security property "confidentiality"; it represents in a general way the relations between the various entities concerned. The main interest of this approach is the processing of rare events by their chronological appearance order.

For the resolution of the offline case, we propose a mathematical formalization of the Problem in which the parameters described in Table 2 are used. The formulation is given below.

Table 2: Parameters used in the mathematical modeling

	Notation	Description
Indexes	i	Numbering the sets and paths
	j	Numbering the equivalent classes
	L_{path}	Path length
	k	Numbering the signatures
	L_{sign}	Signature length
	NbrSign	Number of signatures
	NbrCl	Number of equivalent classes
	NbrSCL	Number of equivalent sub-classes
Path	$\text{Path}_{L_{\text{path}},i,j}^{(\text{Sc}_{\text{Source1}},\text{Sc}_{\text{target}}L_{\text{path}})}$	A path number i, having as source node $\text{Sc}_{\text{Source1}}$ and as destination node $\text{Sc}_{\text{target}}$ belonging to the equivalence class number j and having L_{path} as a length
Decision variable	$x(\text{Sign}_{L_{\text{sign}},k}^{(j,i)})$	Decision variable, which gives an integer number
Class	$\text{Cls}_{j}^{(\text{Sc}_{\text{Source}},\text{Sc}_{\text{target}})}$	Equivalence class containing all the paths starting with the node $\text{Sc}_{\text{Source}}$ and ending with the node $\text{Sc}_{\text{target}}$ with j as the number of this equivalence class
Subclass	$\text{SCL}_{L_{\text{path}},i}^{j}$	Equivalence subclass number i belonging to equivalence class j and all its paths having an identical length L_{path}
Signature	$\text{Sign}_{L_{\text{sign}},k}^{(j,i)}$	A signature number k compressing the paths in subclass number i that belongs to class number j having a length of L_{sign}
The arc source	$\text{source}(\text{arc}_i)$	Function that has as a parameter an arc and returns the source node of this arc
The arc destination	$\text{target}(\text{arc}_i)$	Function that has as a parameter an arc and returns the destination node of this arc
The arc start date	$\text{start}(\text{arc}_i)$	Function that has as a parameter an arc and returns the start date
The arc end date	$\text{end}(\text{arc}_i)$	Function that has as a parameter an arc and returns the end date
The destination	$\text{target}(\text{arc}_i)$	Function that has as parameter an arc and returns the target node of this arc
The source	$\text{source}(\text{arc}_i)$	Function that has as parameter an arc and returns the source node of this arc

Let the following decision variable:

$$x(\text{Sign}_{L_{\text{sign}},k}^{(j,i)}) = \begin{cases} 1, & \textbf{whether the signature in question is effective,} \\ 0, & \textbf{else,} \end{cases}$$

$$\forall L_{\text{sign}} = (L_{\text{sign}})\text{min},\ldots,(L_{\text{sign}})\text{max},$$
$$k = 1, \ldots, \text{NbrSign}, \quad j = 1, \ldots, \text{NbrCl}, \quad i = 1 \ldots \ldots, \text{NbrSCl}$$

For the offline case of solving the information flow graph optimization problem, we propose the following two objective functions:

$$\begin{cases} \textbf{Maximize} & F1(\text{Sign}_{L_{\text{sign}},k}^{(j,i)}) \\ \textbf{Maximize} & \text{CompRate}(G) \end{cases}$$

$$\text{with} \begin{cases} F1(\text{Sign}_{L_{\text{sign}},k}^{(j,i)})\textbf{:allows to test the satisfiability of a signature} \\ \text{CompRate}(G)\textbf{:compression ratio of G graph} \\ 1 \le k \le \text{NbrSign} \\ 1 \le i \le \text{NbrSCL} \\ 1 \le j \le \text{NbrCl} \\ (L_{\text{sign}})\text{min} \le L_{\text{sign}} \le (L_{\text{sign}})\text{max} \end{cases} \tag{1}$$

The $F1$ function is defined on a set of $SIGN$ signatures and gives a value between 0 and 1.

$$\mathbf{Max}\{\boldsymbol{F}1(\mathbf{Sign}^{(j,i)}_{L_{\text{sign}},\,k})|\ \mathbf{Sign}^{(j,i)}_{L_{\text{sign}},\,k} \in \mathbf{SIGN}\}$$

To explain our approach further, we propose the following different definitions:

Definition 1. Graph

In our case, the information flow graph is defined by a tuple $G = (S, A, E)$ such that:

- S is a finite set of vertices; each vertex represents the security context of the system entity.
- A is a set of pairs of vertices $\{(s_i, s_j) \in S^2\}$ representing the interaction between the two vertices
- E represents the values that an arc can take (elementary operation type: *Read, Write, Execute, Transition*, and the start and end date of the interaction) (Figure 1).

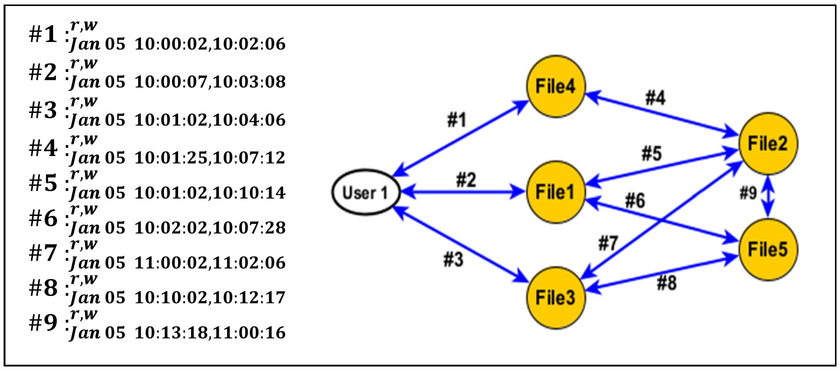
#1 : r,w Jan 05 10:00:02,10:02:06
#2 : r,w Jan 05 10:00:07,10:03:08
#3 : r,w Jan 05 10:01:02,10:04:06
#4 : r,w Jan 05 10:01:25,10:07:12
#5 : r,w Jan 05 10:01:02,10:10:14
#6 : r,w Jan 05 10:02:02,10:07:28
#7 : r,w Jan 05 11:00:02,11:02:06
#8 : r,w Jan 05 10:10:02,10:12:17
#9 : r,w Jan 05 10:13:18,11:00:16
User 1
File4
File1
File3
File2
File5
#1
#2
#3
#4
#5
#6
#7
#8
#9

Figure 1: Example of an information flow graph.

Example. On the following graph:

Taking for example the interaction $\mathbf{user}1 \xrightarrow[\mathbf{Jan\ 05\ 10{:}00{:}07,\ 10{:}03{:}08}]{\boldsymbol{r,w}} \mathbf{file}1$

The **user**1 security context can perform **read** and **write** operations on **file**1.

Definition 2. Summit

Let $\boldsymbol{G} = (\boldsymbol{S, A, E})$ be the information flow graph, a vertex or node $\boldsymbol{x_i}\ \forall \boldsymbol{i},\ 0 \le \boldsymbol{i} \le \boldsymbol{n} - 1$ in the graph G represents the security context of a system entity. A security context is a set of attributes (attri) associated with each operating system entity, representing the system's processes and resources. These entities can be separated into two sets: that of subjects, which are the active entities (processes), and that of objects, which correspond to passive entities (resources, files, and sockets).

$$x_i = \{\text{attr}_1, \text{attr}_2 \ldots . \text{attr}_n\}$$

Example. the security context Sc_{user} and Sc_{admin} correspond respectively to the security contexts associated with users user1, user2 and administrators admin. They are defined by:

$$\text{Sc}_{\text{user}} = \{\text{user1},\ \text{user2}\}$$

$$\text{Sc}_{\text{admin}} = \{\text{admin}\}.$$

Definition 3. Arc

Let $G = (S, A, E)$ be the information flow graph, an Arc represents all the information of a system call; it connects two summits, i.e., the two security contexts of the two system entities that interact through the system call. This arc is labelled by a tuple that defines the type of the elementary operation of the system call (*Read*, *Write*, and *Transition*) and the start and end dates of the system call.

$$\text{Arc} \equiv_{\text{def}} \left[\text{Sc}_{\text{Source}} \frac{\text{eo}}{[\text{sd}, \text{ed}]} \text{Sc}_{\text{target}} \right]$$

$$\text{Arc} \equiv_{\text{def}} \left\{ \begin{array}{l} \text{Sc}_{\text{Source}} \in \text{CSS},\ \text{Sc}_{\text{target}} \in \text{CS},\ \text{eo} \in \text{EO}, \text{sd} \in D, \text{ed} \in D, \\ \text{Sc}_{\text{Source}} \text{ performs the elementary operation eo} \\ \text{on } \text{Sc}_{\text{target}} \text{ starts on sd date and ends on} \\ \text{ed date} \\ \text{with} \\ \text{CSS: is the set of subject security contexts} \\ \text{CS: is the set of all contexts (subject and object)} \\ \text{EO: the whole of the elementary operation} \end{array} \right\}$$

Example. We take the previous graph, the arc $\textbf{user}1 \xrightarrow[\textbf{Jan 05 10:00:07, 10:03:08}]{r,w} \textbf{file}1$

where (**user**1 **and file**1) represent the security contexts, (*r, w*) represents the type of the elementary operation, (**Jan 05 10:00:07, 10:03:08**) represents the start and end dates of the interaction.

Definition 4. Elementary path

Let ***G*** = (***S, A, E***) be the information flow graph. In our graph, we use the notion of an Elementary Path to avoid cycles; an elementary path is represented by a set of system calls (set of arcs) respecting the causality principle. For the principle of causality to be respected, the terminal node of the first arc must be the same as the initial node of the second arc in the path. Likewise, the terminal node of the second arc must be the same as the initial node of the third arc and so on for the rest of the nodes in the path. Accordingly, the start date of the first arc must be less than or equal to the end date of the second arc and the start date of the second arc must be less than or equal to the end date of the third arc, and so on for the rest of the other nodes making up the path.

An elementary path is defined as follows:

$$\textbf{Path}_{L_{\text{path}}, i, j}^{(\text{Sc}_{\text{Source1}}, \text{Sc}_{\text{target}\, L_{\text{path}}})} = \textbf{arc}_1, \textbf{arc}_2, \ldots, \textbf{arc}_{L_{\text{path}}}$$

With: target (arc_i) = source (arc_{i+1}) and start $(\text{arc}_i) \leq$ end (arc_{i+1})

In more detail, the elementary path is defined as follows:

$$\textbf{Path}_{L_{\text{path}}, i, j}^{(\text{Sc}_{\text{Source1}}, \text{Sc}_{\text{target}})} \equiv_{\text{def}} \left\{ \begin{array}{c} \text{arc}_1, \text{arc}_2, \ldots, \text{arc}_{L_{\text{path}}} \\ \text{arc}_1 = \text{Sc}_{\text{Source1}} \xrightarrow[{[\text{sd1}, \text{ed1}]}]{\text{eo1}} \text{Sc}_{\text{target1}} \\ \text{arc}_2 = \text{Sc}_{\text{Source2}} \xrightarrow[{[\text{sd2}, \text{ed2}]}]{\text{eo2}} \text{Sc}_{\text{target2}} \\ \ldots\ldots. \\ \text{arc}_{L_{\text{path}}} = \text{Sc}_{\text{Source}L_{\text{path}}} \xrightarrow[{[\text{sd3}, \text{ed3}]}]{\text{eo3}} \text{Sc}_{\text{target}L_{\text{path}}} \\ \text{Avec:} \\ \text{target}(\text{arc}_1) = \text{source}(\text{arc}_2) \text{ and} \\ \text{start}(\text{arc}_1) \leq \text{end}(\text{arc}_2) \text{ and} \\ \ldots. \text{ and} \\ \text{target}\left(\text{arc}_{L_{\text{path}}-1}\right) = \text{source}\left(\text{arc}_{L_{\text{path}}}\right) \text{ and} \\ \text{start}\left(\text{arc}_{L_{\text{path}}-1}\right) \leq \text{end}\left(\text{arc}_{L_{\text{path}}}\right) \\ L_{\text{path}} : \text{length of the path, i : path number and} \\ \text{jnumber of the class to which it belongs} \end{array} \right\}$$

Example.

$$\textbf{user}1 \xrightarrow[\textbf{Jan 05 10:00:07, 10:03:08}]{r,w} \textbf{file}1 \xrightarrow[\textbf{Jan 05 10:01:02, 10:10:14}]{r,w} \textbf{file}2$$

This elementary path is composed of two arcs:
arc1: $\mathbf{user}1 \, {}^{r,w}_{\mathbf{Jan\ 05\ 10:00:07,\ 10:03:08}} \, \mathbf{file}1$ and
arc2: $\mathbf{file}1 \, {}^{r,w}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:10:14}} \, \mathbf{file}2$
With: [target(arc1) = file1 = source (arc2)] and

$$[\text{start(arc1)} = \mathbf{10:00:07} \text{ and end(arc2)} = \mathbf{10:10:14, 10:00:07} \le \mathbf{10:10:14}]$$

Our model is essentially based on the definition of equivalence classes and equivalence subclasses to facilitate the minimization of paths in the graph and thus the generation of signatures.

The definitions from definition one to definition four are essentially inspired by the work of ref. [3–5] and written according to our formalism.

Definition 5. Equivalence class

An equivalence class denoted $\mathbf{Cls}_j^{(\text{Sc}_{\text{Source}},\ \text{Sc}_{\text{target}})}$represents all elementary paths with the source node $\text{Sc}_{\text{Source}}$ and having as the target node $\text{Sc}_{\text{target}}$, each class is numbered by a number j.

$$\mathbf{PATHS} = \{\text{all Elemental Paths in the information flow graph } G\}$$

$$\mathbf{Cls}_j^{(\text{Sc}_{\text{Source}},\text{Sc}_{\text{target}})} \equiv_{\text{def}} \left\{ \begin{array}{l} \mathbf{Path}_{\boldsymbol{L}_{\mathbf{path}},\boldsymbol{i},\boldsymbol{j}}^{(\text{Sc}_{\text{Source1}},\text{Sc}_{target\boldsymbol{L}_{\mathbf{path}}})} \in \text{PATHS} \\ \text{With:} \\ \text{PATHS:set of all paths} \\ \text{Sc}_{\text{Source}}\text{:source node of paths} \\ \text{Sc}_{\text{target}}\text{:target node of paths} \\ \boldsymbol{L}_{\mathbf{path}}\text{:path length (differs from} \\ \text{one path to another)} \end{array} \right\}$$

Example. $\mathbf{Cls}_{\mathbf{file2},1}^{\mathbf{user1}}$ This class represents all the paths starting with user1 and ending with file2, it contains the following paths:

$$\mathbf{Path}_1 = \mathbf{user1}_{\mathbf{Jan\ 05\ 10:00:07,\ 10:03:08}}^{\mathbf{r,w}} \ \mathbf{file1}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:10:14}}^{\mathbf{r,w}} \ \mathbf{file}2$$

$$\mathbf{Path}_2 = \mathbf{user1}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:04:06}}^{\mathbf{r,w}} \ \mathbf{file3}_{\mathbf{Jan\ 05\ 11:00:02,\ 11:02:06}}^{\mathbf{r,w}} \ \mathbf{file}2$$

$$\mathbf{Path}_3 = \mathbf{user1}_{\mathbf{Jan\ 05\ 10:00:02,\ 10:02:06}}^{\mathbf{r,w}} \ \mathbf{file4}_{\mathbf{Jan\ 05\ 10:01:25,\ 10:07:12}}^{\mathbf{r,w}} \ \mathbf{file}2$$

$$\mathbf{Path}_4 = \mathbf{user1}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:04:06}}^{\mathbf{r,w}} \ \mathbf{file3}_{\mathbf{Jan\ 05\ 10:10:02,\ 10:12:17}}^{\mathbf{r,w}} \ \mathbf{file5}_{\mathbf{Jan\ 05\ 10:13:18,\ 11:00:16}}^{\mathbf{r,w}} \ \mathbf{file}2$$

Definition 6. Equivalence subclass (in our case)

An equivalence subclass noted $\text{SCL}_{\boldsymbol{L}_{\mathbf{path}},\ i}^{j}$, represents the set of elementary paths belonging to class j, starting with summit $\text{Sc}_{\text{Source}}$ and ending with summit $\text{Sc}_{\text{target}}$. All these paths have the same length.

$\boldsymbol{L}_{\mathbf{path}}$, the equivalence subclass, is numbered by i.

$$\text{SCL}_{\boldsymbol{L}_{\mathbf{path}},\ i}^{j} \equiv_{\text{def}} \left\{ \begin{array}{c} \mathbf{Path}_{\boldsymbol{L}_{\mathbf{path}},\boldsymbol{i},\boldsymbol{j}}^{(\text{Sc}_{\text{Source1}},\text{Sc}_{\text{target}\ \boldsymbol{L}_{\text{path}}})} \in \text{PATHS} \\ \text{With:} \\ \text{PATHS:set of all paths} \\ j\text{:number of equivalence class} \\ \boldsymbol{L}_{\mathbf{path}} \text{ : path length(the samefor all paths)} \\ i\text{:number of equivalence sub-class} \end{array} \right\}$$

Example. $\mathbf{SCL}_{2,1}^{1}$ is the first subclass that belongs to the class $\mathbf{Cls}_{\mathbf{file2},1}^{\mathbf{user1}}$ and has 2 as path length, the paths it groups are the following:

$$\mathbf{Path}_1 = \mathbf{user1}_{\mathbf{Jan\ 05\ 10:00:07,\ 10:03:08}}^{\mathbf{r,w}} \ \mathbf{file1}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:10:14}}^{\mathbf{r,w}} \ \mathbf{file}2$$

$$\mathbf{Path}_2 = \mathbf{user1}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:04:06}}^{\mathbf{r,w}} \ \mathbf{file3}_{\mathbf{Jan\ 05\ 11:00:02,\ 11:02:06}}^{\mathbf{r,w}} \ \mathbf{file}2$$

$$\mathbf{Path_3} = \mathbf{user1}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:00:02,\ 10:02:06}}\ \mathbf{file4}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:01:25,\ 10:07:12}}\ \mathbf{file}2$$

Definition 7. Signature

The signature in this article is an elementary path representative of a set of paths in the information flow graph G and going against a security property. A signature noted $\mathbf{Sign}^{(j,i)}_{L_{\mathbf{sign}},\,k}$ is defined as follows:

$$\mathbf{Sign}^{(j,i)}_{L_{\mathbf{sign}},k} \equiv_{\text{def}} \left\{ \begin{array}{l} \text{Elementary paths representative of a set} \\ \text{ of paths in the graph } G \text{ with} \\ j\text{:number of equivalence class} \\ i\text{:number of equivalence sub-class} \\ k\text{:number of signature} \\ L\text{sign:Signature length} \end{array} \right\}$$

Example:

The paths in the subclass $\mathbf{SCL}^{1}_{2,1}$are compressed by the following signatures:

$$\mathbf{Sign_1} = \mathbf{user1}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:00:07,\ 10:03:08}}\ \mathbf{file1}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:10:14}}\ \mathbf{file}2$$

$$\mathbf{Sign_2} = \mathbf{user1}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:04:06}}\ \mathbf{file3}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 11:00:02,\ 11:02:06}}\ \mathbf{file}2$$

$$\mathbf{Sign_3} = \mathbf{user1}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:00:02,\ 10:02:06}}\ \mathbf{file4}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:01:25,\ 10:07:12}}\ \mathbf{file}2$$

$$\mathbf{Sign_4} = \mathbf{user1}^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:00:07,\ 10:03:08}}\ \star\ ^{\mathbf{r,w}}_{\mathbf{Jan\ 05\ 10:01:02,\ 10:10:14}}\ \mathbf{file}2$$

By using the genetic algorithm, we choose the appropriate signatures and eliminate the others.

Definition 8. Equivalence relation 1 (**in our case**)

Our first equivalence relation is defined on the set of graph paths, it is the relation when all paths have the same source and target node. So the equivalence relation $R1$ is defined on the set of paths PATHS such that $\forall\ \boldsymbol{x}, \boldsymbol{y} \in \mathbf{PATHS},\ \boldsymbol{x}\ \boldsymbol{R}1\,\boldsymbol{y}$, if $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same source and target. This relation is reflexive, symmetric, and transitive.

Definition 9. Equivalence class 1 (**in our case**).

Let **PATHS** be a set of paths, $R1$ an equivalence relation on **PATHS** and $\boldsymbol{x} \in$ **PATHS**. We call the equivalence class of $\boldsymbol{x}$ the set:

$$\dot{x} = \text{Cls1}(x) = \{y \in \text{PATHS},\ y\ R1\,x\}$$

With $R1$, the equivalence relation is defined in Definition 8.

Definition 10. Equivalence relation 2 (**in our case**).

Our second equivalence relation is defined on the set of graph paths; it is the relation where all paths have the same source and target node and the same path length. Thus, the equivalence relation $R2$ is defined on the set of paths **PATHS** such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{PATHS},\ \boldsymbol{x}\,\boldsymbol{R}2\,\boldsymbol{y}$ if $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same source and target and the same path length. This relation is reflexive, symmetric, and transitive.

Definition 11. Equivalence class 2 (**in our case, it defines the equivalence subclass**)

Let **PATHS** be a set of paths, $R2$ an equivalence relation on **PATHS** and $\boldsymbol{x} \in$ **PATHS**. We call the equivalence class of x the set:

$$\dot{x} = \text{Cls2}(x) = \{y \in \text{PATHS},\ y\ R2\,x\}$$

with $R2$ the equivalence relation is defined in 10.

Therefore, in our graph, we use the notions of equivalence classes, equivalence subclasses, and signatures to minimize the number of paths between two vertices of the graph. These notions are represented in the following figure, which shows our idea of graph minimization in a general way (Figure 2).

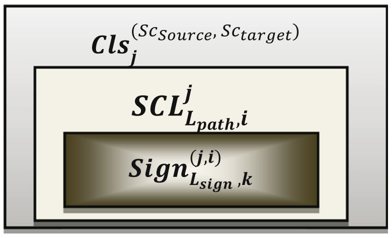
$Cls_j^{(Sc_{Source}, Sc_{target})}$
$SCL_{L_{path},i}^{j}$
$Sign_{L_{sign},k}^{(j,i)}$

Figure 2: Equivalence classes, equivalence subclasses, and signatures.

From the number of paths in a subclass, we form the signatures corresponding to this subclass using all possible combinations; each node can take the values attributed to the nodes of the corresponding paths.

We propose two criteria to classify a compressed signature. This classification is measured by the ability of the compressed signature to detect the most significant number of illicit behaviour and the smallest number of false negatives and false positives. The first criterion, "*Prediction*," measures the probability that this signature predicts any illicit behaviour. The higher this criterion is, the more effective the signature is. The prediction calculation function is as follows:

$$\begin{cases} \mathbf{Pre}(\mathbf{Sign}_{L_{\text{sign}},k}^{(j,i)}) = \dfrac{\mathbf{Tp}(\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)})}{z(\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)})} \\ \textbf{with} \begin{cases} z(\mathbf{Sign}_{L_{\text{sign}},k}^{(j,i)}) = \sum\limits_{j=1}^{\mathbf{NbrCls}} \sum\limits_{i=1}^{\mathbf{NbrSCl}} \mathbf{Path}_{L_{\text{path}},i}^{j} \textbf{ and} \\ 1 \le \mathbf{j} \le \mathbf{NbrCl} \textbf{ and} \\ 1 \le \mathbf{i} \le \mathbf{NbrSCL} \textbf{ and} \\ L_{\text{path}} = L_{\text{sign}} \end{cases} \end{cases} \tag{2}$$

Such that:

$\mathbf{Tp}(\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)})$ represents the number of true positives that this signature can detect.

L_{path} represents the same number for all signatures.

$z(\mathbf{Sign}_{L_{\text{sign}},\,i}^{j})$ represents the number of paths with the same path length as $\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)}$ belonging to all subclasses and classes.

The second criterion "***Sensitivity***" is the number of paths well classified by this signature. The sensitivity measures the capacity of a signature to detect all the illicit behaviours ensuring the least number of false negatives. The higher this criterion is, the smaller the number of false negatives is. The function for calculating the sensitivity of a signature is as follows:

$$\mathbf{Se}(\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)}) = \frac{\mathbf{Tp}(\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)})}{\mathbf{Tp}(\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)}) + \mathbf{Fn}(\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)})} \tag{3}$$

With:

$\mathbf{Fn}(\mathbf{Sign}_{L_{\text{sign}},\,k}^{(j,i)})$ is the number of false negatives detected by this signature.

For a signature to be effective, we propose the following mathematical model:

$$\begin{cases} \textbf{Maximize Pre}(\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}) \\ \textbf{Maximize Se}(\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}) \end{cases}$$

$$F1(\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}) = \alpha \star \textbf{Pre}(\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}) + \beta \star \textbf{Se}(\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}) \tag{4}$$

$$\textbf{with} \begin{cases} \alpha + \beta = 1 \quad (\textbf{4.1}) \textbf{ and} \\ 0 \le F1(\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}) \le 1 \end{cases}$$

In order to group, the two criteria of the signature efficiency test, we propose the weighted sum method presented above.

Then, α and β are coefficients whose sum equals 1; these coefficients reduce the multi-objective signature effectiveness testing problem to a single-objective optimization problem. Furthermore, the constraint (**4.1**) $\boldsymbol{\alpha} + \boldsymbol{\beta} = 1$ ensures that the value of the objective function $F1$ is contained in the interval [**0,1**], so the closer the value to 1, the more efficient the signature is inverse; a value close to 0 eliminates this signature.

We initially assume that the two coefficients, α and β, have the same level of importance, hence: $\boldsymbol{\alpha} = \mathbf{0.5}$ and $\boldsymbol{\beta} = \mathbf{0.5}$, and we change the values of these two criteria to see their influence on the quality of the results.

$$\begin{cases} \textbf{NbrSignComp} = \sum_{j=1}^{\textbf{NbCls}} \sum_{i=1}^{\textbf{NbrSCL}} x(\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}) \\ \textbf{with} \begin{cases} 1 \le j \le \textbf{NbCls} \text{ and} \\ 1 \le i \le \textbf{NbrSCL} \end{cases} \end{cases} \tag{5}$$

The function (5) calculates the number of compressed signatures after testing one by one by the objective function $\boldsymbol{F1}$. Thus, if this signature is effective, the decision variable $\boldsymbol{x}$ ($\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}$) associated with it will have a value of 1, so it will be counted. Otherwise, if it is not effective, its decision variable $\boldsymbol{x}$ ($\textbf{Sign}^{(j,i)}_{L_{\text{sign}},\, k}$) will have a value of 0, so the signature will not be counted.

$$\begin{cases} \textbf{NbrAllPath} = \sum_{j=1}^{\textbf{NbrCls}} \sum_{i=1}^{\textbf{NbrSCL}} \sum_{(L_{\text{path}})\textbf{Min}}^{(L_{\text{path}})\textbf{Max}} \textbf{Path}^{j}_{L_{\text{path}},\, i} \\ \text{with} \begin{cases} 1 \le j \le \textbf{NbrCls and} \\ 1 \le \boldsymbol{i} \le \textbf{NbrSCL} \\ \textbf{Min}(\boldsymbol{L}_{\textbf{path}}) \le \boldsymbol{L}_{\textbf{path}} \le \text{Max}(\boldsymbol{L}_{\textbf{path}}) \end{cases} \end{cases} \tag{6}$$

The function (6) calculates the number of all paths that we wanted to minimize.

$$\textbf{CompRate}(\boldsymbol{G}) = 1 - \frac{\textbf{NbrSignComp}}{\textbf{NbrAllPath}}. \tag{7}$$

Function (7) calculates the compression ratio of the paths in graph G. The compression ratio is in the interval [0,1], a better compression is the one that is close to 1. A compression ratio equal to 0 means that none of the paths is compressed, and our optimization is not interesting.

4.2 Complexity and objectives of the problem

4.2.1 Theory of algorithm complexity

Our Problem is a graph optimization problem. This graph is an instance of considerable size. It consists of as many nodes as there are entities in the system. According to several works, graph optimization problems belong to the class NP-complete. Knowing that the Problem we are working on is NP-complete is an indication that the problem is difficult to solve. In other words, it is NP-hard, so it is better to look for approximate solutions using heuristics and meta-heuristics to find exact solutions [6].

4.2.2 Solving the Problem according to its size and objectives

There are two main families of methods to solve problems, especially optimization problems [9]: the exact (complete) methods, which guarantee the completeness of the solution, and the approximate (incomplete) methods, which are efficient but not complete. The exact methods guarantee the optimal solution for a given optimization problem, and their use is very interesting. However, they are limited to the case of small problems; in addition, they generate unacceptable execution times [1]. On the other hand, approximate methods apply to many problems and show that they can lead to satisfactory results in much more reasonable times [9]. Therefore, due to our problem's size, we were naturally driven to apply approximate methods.

Multi-objective optimization problems are mostly NP-complete or rather NP-hard. Moreover, their size changes from one Problem to another; depending on this size, two classes of methods have been distinguished: exact algorithms for small problems and heuristics for solving large problems and problems with two or more objectives.

Therefore, given the complexity of graph optimization, the size of our graph and the objectives to be reached, i.e., guaranteeing the efficiency and the performance of the system, we always come across the need to use heuristics to optimize the graph and to solve our Problem classified in the NP-hard class.

We choose a meta-heuristic Algorithm, mainly a Genetic Algorithm, to solve our problem. This choice is justified, like said above, by its complex class and problem objectives. Indeed, genetic algorithms have proven their efficiency in finding reasonable solutions to difficult problems of large size [10,11] and have been applied several times to 3-SAT problems [12] and graph problems such as graph colouring [13].

5 Resolution algorithm

Evolutionary algorithms represent an essential tool for solving optimization problems. Moreover, they are increasingly used in many fields. They are easy to implement and provide excellent performance at a low cost. Genetic algorithms are part of this family, and they explore domains with many solutions. In other words, the principle is to simulate the evolution of a population of various individuals to which we apply different genetic operators.

5.1 Application of genetic algorithms to the context of our information flow graph optimization work

The role of the genetic Algorithm here is

- To find a signature of illicit behaviour that checks both constraints:
  - $\boldsymbol{F}1(\mathrm{Sign}^{(j,\mathrm{i})}_{L_{\mathrm{sign}},k}) \geq \mathbf{0.5}$; the signature must be able to detect at least 50% of the illicit behaviours it represents; this favours the increase in detection rate
  - $\mathbf{Nbr}(\star) \leq ((\mathbf{Nbr}(\mathbf{node}) + \mathbf{Nbr}(\mathbf{Arc}))/2)$; the appearance of (*) in the signature should not be more than half of the number of nodes plus the number of arcs in the signature; it promotes the decrease of false positives. (* means any value)
- Minimize the number of paths efficiently.

– If $\boldsymbol{F}1(\mathbf{Sign}^{(j,i)}_{L_{\text{sign}},k}) \geq 0.5$ and **Nbr** (*) ≤ ((**Nbr**(**node**) + **Nbr**(**Arc**))/2), then we will keep the signature. Otherwise, it will be passed to the evolutionary process to produce new and more significant signatures.

5.2 The architecture of the Genetic Algorithm in our case

Genetic evolution begins with the initial population. The mechanism of generation of this population must be able to produce a population of non-homogeneous individuals that will serve as the basis for future generations. The choice of the initial population is important because it can make the convergence towards the optimum more or less rapid. If nothing is known about the problem to be solved, the initial population must be distributed over the entire research domain. In our research, we invest the approach to detecting illicit behaviours based on models that represent illicit behaviours. More specifically; one can have two global views of the subject:
– A generic model represents all types of illicit behaviour.
– A model for each type of illicit behaviour.

We are motivated by the establishment of host-based IDS in order to detect as many types of illicit behaviour as possible. However, each behaviour has its specificities. Proposing an initial population without considering this can lead to unsatisfactory results, especially when the applied genetic algorithm graph is huge. To remedy this shortcoming, our contribution was to propose an initial population for each type of illicit behaviour (diversification of the initial population). The size differs from one population to another, considering that the size of the chromosomes is not fixed. Figure 3 shows the structure of the chromosome.

Summit1	**Arc1**	**Summit2**	**Arc2**	**Summit3**

Chromosome

Figure 3: Chromosome representation for an elementary path of 3 summits.

A chromosome is a feasible solution in our implementation, consisting of a record of string saved into the MongoDB dataset. Chromosomes in Figure 3 represent an elementary path from **summit1** (source node) to **summit3** (target node). Figure 4 shows a real example of such a chromosome.

$user1$	r,w $Jan\ 05\ \ 10{:}00{:}07,\ \ 10{:}03{:}08$	$file1$	r,w $Jan\ 05\ \ 10{:}01{:}02,\ \ 10{:}10{:}14$	$file2$

Chromosome

Figure 4: Real Example of a chromosome of 3 summits.

Algorithm 1: Informal algorithm for information flow graph optimization using genetic algorithms

```
Optimization (G)
Start
1) Initialization of the initial population P_0 (generation 0 population)
–For every two nodes of the graph, name all the elementary paths between them while respecting the principle of causality;
–Classify these paths in classes;
–Partition each class into subclasses;
```

–For each subclass, signatures are generated, and the value of the *i*th gene of the signature is randomly chosen among the values of the *i*th gene of all the paths in this subclass;

–The signatures are generated;

2) Evaluation of the chromosomes (signatures) of the population P_i

–If $\mathbf{F1}(\mathbf{Sign}^{(\mathbf{j,i})}_{\mathbf{L_{sign}},\,\mathbf{k}}) \geq 0.5$ and $\mathbf{Nbr}\,(*) \leq ((\mathbf{Nbr}\,(\mathbf{nodes}) + \mathbf{Nbr}(\mathbf{Arc}))\,/\,2)$then keep the signature and go on to evaluate the next one

Otherwise, pass this signature to the evolutionary process to generate new individuals

–These signatures will constitute the procreators of the generation $\mathbf{P_{i+1}}$

3) Crossing two by two of the selected signatures to generate new individuals, the descendants

4) Generation of the new population P_(i + 1)

5) Repeat the process from 2 as long as the stopping criterion is not satisfied (in our case, the criterion is: generation N terminal not reached)

End

Figure 5 shows a global architecture of the genetic algorithm application for our case of information flow graph optimization.

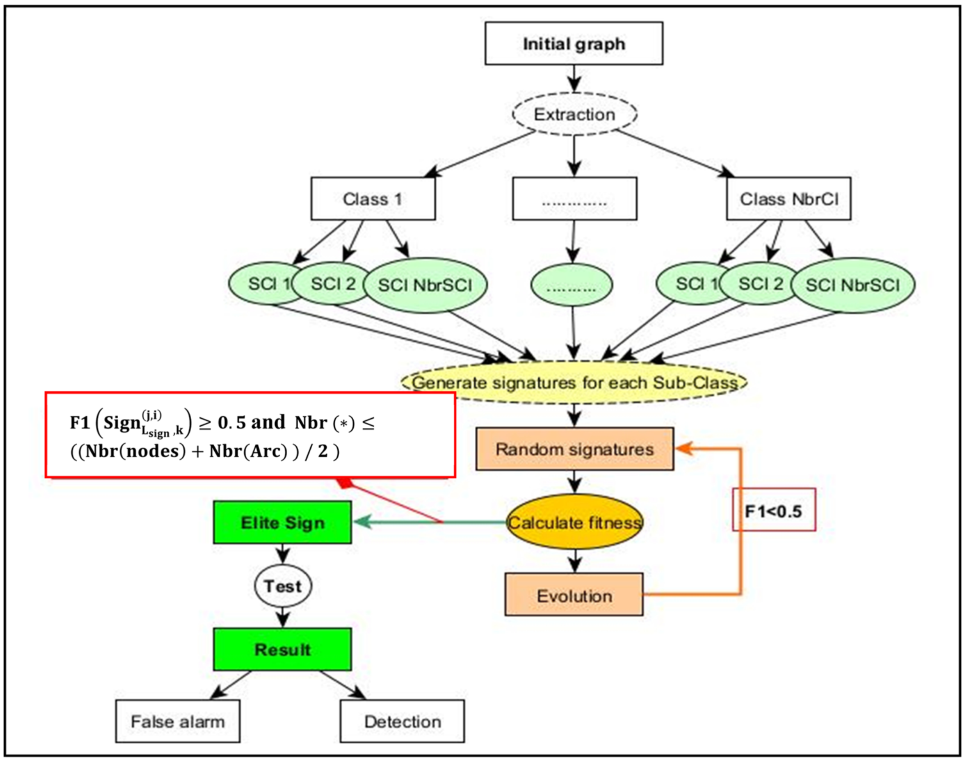
Initial graph
Extraction
Class 1
....
Class NbrCl
SCI 1
SCI 2
SCI NbrSCI
....
SCI 1
SCI 2
SCI NbrSCI
Generate signatures for each Sub-Class
$\mathbf{F1}\left(\mathbf{Sign}^{(\mathbf{j,i})}_{\mathbf{L_{sign}},\mathbf{k}}\right) \geq \mathbf{0.5}$ and $\mathbf{Nbr}\,(*) \leq ((\mathbf{Nbr}(\mathbf{nodes}) + \mathbf{Nbr}(\mathbf{Arc}))\,/\,\mathbf{2})$
Random signatures
F1<0.5
Elite Sign
Calculate fitness
Test
Evolution
Result
False alarm
Detection

Figure 5: Application of the Genetic Algorithm for information flow graph optimization.

5.3 Construction of the quotient space

Using the equivalence relation defined in definitions 8 and 10, we form the equivalence classes and subclasses of all the obtained signatures.

Let ***R2*** be the equivalence relation defined in definition 10. Then, the equivalence subclasses realize a partition of signatures *SIGN*; this partition is obtained by grouping together the signatures “equal modulo the relation ***R2***.

Définition 12
Quotient space (used in our case)

Let SIGN be a set of signatures, $\boldsymbol{R2}$ an equivalence relation on SIGN and $x \in$ SIGN. We call the equivalence class of x the set:

$\dot{\boldsymbol{x}} = \mathbf{Cls}2(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbf{SIGN}\ ,\ \boldsymbol{y}\,\boldsymbol{R}2\,\boldsymbol{x}\}$ With $\boldsymbol{R2}$, the equivalence relation is defined in 10. We call the quotient space the set:

$$\mathrm{SIGN}/R2 = \ \dot{x}/x\ \in\ \mathrm{SIGN}$$

The definition of the equivalence relation $R2$ on the signatures gave rise to a partition or equivalence classes. However, conversely, this one can be transformed into an equivalence relation when we have a partition. As there is a round trip between the notion of equivalence relation and the notion of partition, which allows the transformation of equality more or less into true equality, the definition that we gave of the equivalence relation is relevant to generalizing the equality relation.

In other words, from the paths with which we associate an equivalence relation, we have seen that it constitutes our quotient space. Moreover, the elements of the equivalence class are identical, so we can choose one or two elements that can represent all the others. Here is a figure schematizing the construction of the quotient space (Figure 6).

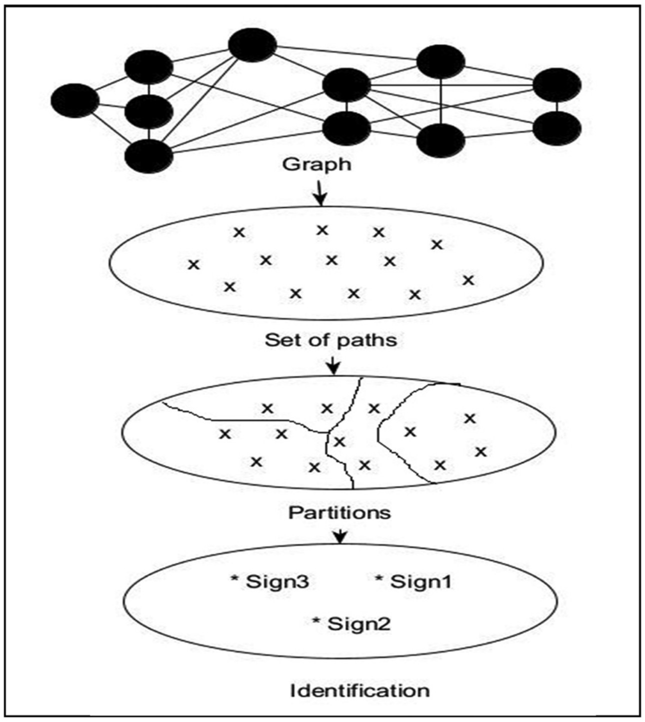

Figure 6: Construction of the quotient space.

6 Experimentations and results

At the end of this part, we have concluded the last phase of this article, namely the chosen programming language, the chosen graph, and the interpretation and discussion of the experimental results.

6.1 Programmation language

The programming language used to implement the solution proposed in Section 5 was Python. It was chosen because of the availability of libraries and documentation and its fast and efficient execution. The database used to record the results in MongoDB. It offers speed and efficiency in data processing and text indexing, allowing powerful querying and analytics.

6.2 Description of the chosen graph

The graph chosen is in Figure 7, the same graph used by Pierre Clairet to apply his optimization solution [5]. This graph was chosen in order to be able to compare the two solutions. This graph is composed of 11 vertices (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11) and 58 arcs. In this case, we work on the undirected graph. To evaluate the effectiveness of our method, we calculate the number of paths between two source/target pairs. From these paths, we classify them into classes and subclasses to generate signatures of illicit behaviour.

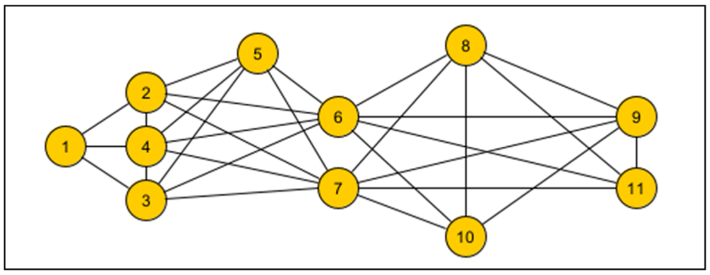

Figure 7: Graph example.

Here is an informal algorithm showing the steps followed to implement our solution.

Algorithm2: Algorithm showing the steps for the implementation of our solution

```
1) Graph in Figure 7, it represents the allowed information flows
2) We want to prohibit information flows between these source/destination pairs ([1, to all other nodes],
   [2, to all other nodes], [3, to all other nodes], [4, to all other nodes], [5, to all other nodes], [6, to all
   other nodes])
3) Classify these paths into equivalence classes and subclasses
4) For each subclass, create the corresponding collection (the collection is an array in the database)
5) For each subclass, create variables A1, A2, …. An containing the possible values, respectively, of the
   first node, the second node, the third node ….etc.
6) For each subclass, generate the signatures from the variables A1, A2, …. An
7) Execute Algorithm 1 from step 2 for each signature to keep or delete it (calculate the F1 function for each
   signature)
8) Calculate the compression rate (1-(Number of compressed signatures/number of all paths to be
   reduced)
9) Calculate the detection rate, false-positive rate, false-negative rate
10) Compare the results obtained with the results obtained by executing the optimization method of Pierre
   Clairet
```

6.3 The influence of the generation factor on the performance of the Genetic Algorithm

We proceed to study the influence of the number of generations on the performance and, more precisely, to determine the pace of the convergence of this performance to draw the necessary information concerning the stability of the Genetic Algorithm. The stability is a relevant factor revealing the degree of adaptation of the Algorithm and its power to recognize the types of attacks in a minimal time and proceed to construct the sought models as soon as possible.

The study carried out considers different generation lapses according to the path length. The results are collected at the end of each period. The experimental results are translated into graphs for a better expression. Take an example of the equivalence subclass, which contains all the paths from node 2 to all the other nodes with a length of 5. Each generation has 100 signatures. The following table shows the results obtained for each generation; the associated curve is presented in Figure 8 (Table 3).

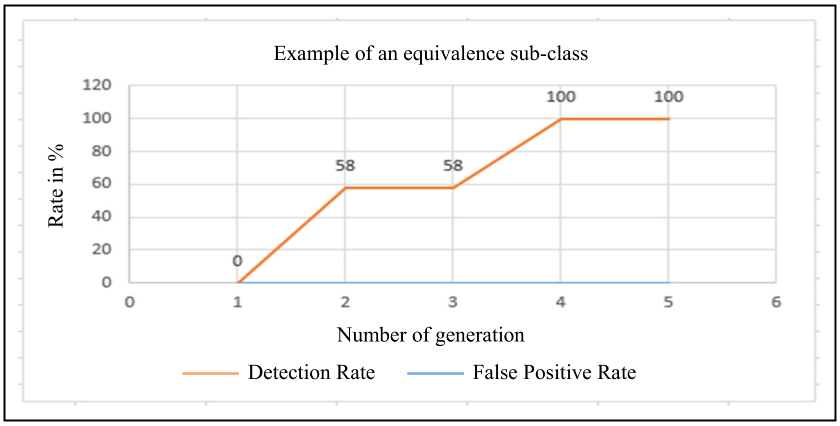

Figure 8: Detection result curve for an example of an equivalence subclass.

Table 3: Detection result for an example of an equivalence subclass

Generation	1	2	3	4	5
Detection rate (%)	0.00	57.62	57.62	100	100
False positive rate (%)	0.00	0.00	0.00	0.00	0.00

The evolution is launched during the second generation; we notice that the detection rate accelerates at each generation and stabilizes at 100% in the fourth generation, while the false positive rate stabilizes in the first generation.

7 Interpretation and discussion of the results

A good IDS can distinguish between normal and abnormal activity [1]. The challenge of this study was to satisfy two contradictory objectives; the detection of attacks and the system's performance. Therefore, a comparison will be made between these two notions. The same graph was used in both optimization methods (our's and Pierre Clairet's method), and the results for both are shown in the following tables (Tables 4 and 5):

Table 4: Results of the two methods in terms of Performance (Graph compression)

Methods	Number of paths	Number of generated signatures	Number of compressed signatures	Compression rates
Pierre Clairet	103,411	—	1,436	98.61%
Ours	110,295	—	620	99.44

The number of paths used in our method is more significant than those used in Pierre Clairet (110,295 against 103,411). The number of signatures obtained by applying the method of Pierre Clairet was 1,436,

Table 5: Results of the two methods in terms of detection (detection rate and error rate)

Methods	Number of paths	Detection rates	False-positive rates	False-negative rates
Pierre Clairet	103,411	—	—	—
Ours	110,295	99.9%	0%	0.1%

which gives a compression rate of 98.61%. In our method, the number of signatures obtained by applying genetic algorithms was 620, which gives a compression rate of 99.44%. To test the efficiency of an IDS, the criteria listed above must be calculated. In Pierre Clairet's method, none of these criteria was given clearly (no number showing their values). The table below shows another comparison (Table 6).

Table 6: Global comparison of the two methods

Methods	Graph	Detection Rates	False-positive rates	False-negative rates	Pre-processing time	Detection time (s)	Cycles in graph
Pierre Clairet	Same	✗	✗	✗	✗	11.125	✓
Ours	Same	✓	✓	✓	Week	6.75	✗

Our method has been developed aiming at a minimal impact on the operating system's performance on which it will be deployed while maximizing the efficiency of PIGA HIDS. Our method gave a high detection rate of 99.9%, a low false-negative rate of 0.1% and no false positives. Despite the promising results obtained by this model, it is not free of shortcomings. The proposed model does not consider the cycles in the graph, and the pre-processing time (phase 2) is considerable, about a week. However, the detection time (phase 3) takes just 6.75 s, which is better than Pierre's Clairet 11.125 s.

8 Conclusion and perspectives

As described in this article, our main contributions to improving the efficiency of host-based IDS PIGA have two parts: the definition of a new classification mathematical model based on the use of equivalence classes/equivalence subclasses and a hybrid method that combines a signature-based approach and anomaly-based approach. To the best of the authors' knowledge, our approach is the first to improve the efficiency of PIGA HIDS and discusses all the evaluation criteria (detection rate, false-positive rate, false-negative rate, and compression rate).

Effectively, the main goal of this article is to minimize the graph, that is, to minimize the number of paths between two nodes of the graph. The use of the equivalence class concept gives the notion of diversification of the initial population used for the genetic algorithms. It allows us to construct the quotient space that constitutes the minimization of the paths in the graph. Besides, the proposed model is extensible/scalable and can be used in other databases with another pattern recognition method or the same method to detect malicious behaviour. Moreover, the hybridization proposed for PIGA HIDS allowed us to create more accurate HIDs and can better fit their design by focusing on lowering false alarm rates. We compared its outstanding performance with Pierre Clairet's [5] method and demonstrated its robustness and generality through experiments.

As discussed earlier, we have succeeded in satisfying our contradictory objectives (keeping the efficiency of the PIGA HIDS without affecting the operating system's performance. The efficiency in this article means a high detection rate and a low false-positive and false-negative rates). Further, the set of all signatures reduces the memory space consumed and thus the CPU load.

As part of our future work, we are planning to test the efficiency of our method against other methods in the same field, especially that of Jonathan Cornabas. We aim to use fuzzy logic to choose the parameters α and β. In addition, we intend to design and implement an anomaly detection technique for the Linux platform using deep learning and a real database ADFA-LD, especially exploring the pre-processing of system call traces to generate fixed-size call sequences. These sequences are used to train an LSTM deep-based learning model. The model consists of several layers: the embedding layer, LSTM layer, distributed time layer, and softmax layer.

Finally, for optimization of the present work, we would be able to take into account the cycles in the graph and test our method on the real graph used by Pierre Clairet. For another optimization, we can add signatures of licit behaviour to detect new types of attacks (zero-day attacks), therefore minimizing false negatives.

Conflict of interest: Authors state no conflict of interest.

References

[1] Samrin R, Vasumathi D. Hybrid weighted k-means clustering and artificial neural network for an anomaly-based network intrusion detection system. J Intell Syst. 2018;27(2):135–47.

[2] Elmasry W, Akbulut A, Zaim AH. A design of an integrated cloud-based intrusion detection system with third party cloud service. Open Computer Sci. 2021;11(1):365–79.

[3] Brifaut J. Formalization and guarantee of system security properties: application to intrusion detection. PhD thesis. Orléans: Orléans University; 2007.

[4] Cornabas JR. Formalization of security properties for the protection of operating systems. PhD thesis. Orléans:Orléans University; 2010.

[5] Clairet P, Berthomé P, Briffaut J. Signature compression for PIGA IDS, 9th ed. France: MajecSTIC; 2012.

[6] Kim C, Jang M, Seo S, Park K, Kang P. Intrusion detection based on sequential information preserving log embedding methods and anomaly detection algorithms. IEEE Access. 2021;9:58088–101.

[7] Shams EA, Rizaner A, Ulusoy AH. A novel context-aware feature extraction method for convolutional neural network-based intrusion detection systems. Neural Comput Applic. 2021;33:13647–65.

[8] Subba B, Gupta P. A tfidf vectorizer and singular value Decomposition based host intrusion detection system framework for detecting anomalous system processes. Computers Sec. 2021;100(102084).

[9] Hemmak A, Bouderah B. New properties for solving the single-machine scheduling problem with early/tardy jobs. J Intell Syst. 2017;26(3):531–43.

[10] Resende PAA, Drummond A. Adaptive anomaly-based intrusion detection system using genetic Algorithm and profiling. J security Priv. 2018;1(4):e36.

[11] Gauthama Raman MR, Somu N, Kirthivasan K, Liscano R, Shankar Sriram VS. An efficient intrusion detection system based on hypergraph – Genetic Algorithm for parameter optimization and feature selection in support vector machine. ELSEVIER Knowl Syst. 2017;134:1–12.

[12] Huimin F, Yang X, Guanfeng W, Hairui J, Wuang Z, Rong H. An improved adaptive genetic algorithm for solving 3-SAT problems based on effective restart and greedy strategy. Int J Computational Intell Syst. 2018;11(1):402–13.

[13] Arindama D, Aayushb A, Pranavb D, Hoang L, Franke W, Tandrab P, et al. A genetic algorithm for total graph colouring. J Intell Fuzzy Syst. 2019;37(6):7831–8.