computation

MDPI

Article

Algebraic Structures Induced by the Insertion and Detection of Malware

Agustín Moreno Cañadas [1,*], Odette M. Mendez [2] and Juan David Camacho Vega [1]

[1] Departamento de Matemáticas, Universidad Nacional de Colombia, Edificio Yu Takeuchi 404, Kra 30 No. 45-03, Bogotá 11001000, Colombia
[2] Departamento de Matemáticas, Universidad Nacional de Colombia, La Nubia, Manizales 170003, Colombia
* Correspondence: amorenoca@unal.edu.co

Abstract: Since its introduction, researching malware has had two main goals. On the one hand, malware writers have been focused on developing software that can cause more damage to a targeted host for as long as possible. On the other hand, malware analysts have as one of their main purposes the development of tools such as malware detection systems (MDS) or network intrusion detection systems (NIDS) to prevent and detect possible threats to the informatic systems. Obfuscation techniques, such as the encryption of the virus's code lines, have been developed to avoid their detection. In contrast, shallow machine learning and deep learning algorithms have recently been introduced to detect them. This paper is devoted to some theoretical implications derived from these investigations. We prove that hidden algebraic structures as equipped posets and their categories of representations are behind the research of some infections. Properties of these categories are given to provide a better understanding of different infection techniques.

Keywords: additive functor; cybersecurity; computer virus; malware; metamorphic virus; poset; poset representation

MSC: 68M25; 16G20; 16G30; 16G60

Citation: Cañadas, A.M.; Mendez, O.M.; Camacho, J.D. Algebraic Structures Induced by the Insertion and Detection of Malware. *Computation* **2023**, *11*, 140. https://doi.org/10.3390/computation11070140

Academic Editor: Yudong Zhang

Received: 7 June 2023
Revised: 7 July 2023
Accepted: 7 July 2023
Published: 11 July 2023

1. Introduction

Nowadays, the daily life of human being is significantly affected by computers and informatic systems, making cybersecurity one of the main concerns to be addressed by government agencies and companies to protect users from diverse threats arising from Internet use. Such threats are mainly provoked by malware, i.e., a malicious software designed to perform some unauthorized, often harmful or undesirable acts. Computer viruses, trojan horses, worms, and ransomware are examples of malware [1–4].

According to Cohen [5], who is considered the pioneer researcher in computer viruses, a virus is a program that is able to infect other programs by modifying them to include a possibly evolved copy of itself. Cohen wrote the first program of this type which is currently known as the Stoned boot virus. Another example of a computer virus is Stuxnet [6] considered the first cyber-warfare weapon ever.

Perhaps the simplest kind of malware is a Trojan horse which tries to appeal to and interest the user with some useful functionality to entice the user to run the program. In particular, they have been used to steal passwords. Rootkits, AIDS TROJAN DISK, Qbot (malware specialized in stealing user data) and TrickBot (malware focused on stealing financial data) are examples of Trojan horses.

Worms are also examples of malware. They are network viruses, primarily replicating on networks. Usually, these programs execute themselves automatically on a remote machine with minimal user intervention. Particularly, worms do not require a host program. SQL Slammer, Melissa (which is a macrovirus), Morris, as well as Netbus, Subseven, Deep Throat, Back Orifice and Concept, are some of the most known worms [1].

Ransomware is a kind of malware which encrypts data on a computer to prevent users from accessing their computer files or systems. Cybercriminals hold the data until a ransom is paid. It is worth pointing out that the FBI has observed that one of the most frequent attacks carried out over the last few years by cybercriminals is realized via some ransomware. Wannacry, LockBit, Cryptolocker, Sodinokibi/REvil, and Phobos are examples of this type of attack. According to Ploszek et al. [7], crypto Ransomware is the most dangerous among the different Ransomware attacks. These attacks allow the encryption of images, videos and any valuable user files.

At the beginning of the antivirus industry, malware detection was based on heuristic features that identified particular malware by creating a reliable fingerprint. During the detection, an antiviral engine checked the presence of the malware fingerprint in a file against known malware fingerprints stored in the antivirus database. String Scanning, Wildcards and Mismatches are examples of the first virus detection programs. Wildcards were used to detect the metamorphic virus w32/Regswap. These techniques allowed finding the sequence 83EB 0274 1683 EBOE 74OA 81EB 0301 0000 which identifies the w32/Beast virus [1].

Fingerprints associated with infected files were sensitive to small changes in files. Furthermore, malware writers invented metamorphic and polymorphic viruses, which give rise to hundreds of thousands of new virus versions, making the previous detection approaches ineffective. In this line, malware detection systems have been developed based on traditional machine learning (support vector machines, decision trees, naive Bayes classifier, etc.) and deep learning algorithms based on recurrent neural networks (RNNs) [8,9].

It is worth pointing out that several authors need to be more convinced of the RNN's effectiveness for intrusion detection due to their vulnerability against adversarial attacks. These authors have preferred the use of images and to train convolutional neural networks to learn feature malware [10–12]. Another advancement in dealing with the use of machine learning in NIDS was proposed by Iglesias and Criado [13], who used time series, visibility graphs and multiplex networks to analyze the behavior of attackers' computers. They pointed out that tools such as Snort used to analyze network traffic and protocol have disadvantages (e.g., no zero-day attacks detection [14]) to network intrusion detection.

Kaspersky [15] developed detection malware tools based on machine learning techniques. In such a case, hash functions, and unsupervised learning confluent to extract file features that can be computed quickly and directly retrieved from the structure of the executable, like a file format description. Authors refer to [16,17] for good surveys regarding recent trends of the deep learning use for malware detection, in particular, for descriptions of cloud-based malware detection, mobile-device-based malware detection, and IoT-based malware detection.

1.1. Motivations

Currently, there needs to be more malware investigations dealing with its relation to the theory of representation of algebras. A comprehensive algebraic study of malware insertion detection will give rise to a better understanding of different cyber-attacks; works in this direction have been proposed by Webster [18]. This paper proves that attacks of type Linux/Slapper or Scalper and some other metamorphic attacks in confluence with detection techniques as those presented by Kaspersky based on machine learning methods give rise to categories of representations of partially ordered sets. In particular, obfuscation techniques associated with metamorphic attacks define categorical equivalences between these categories.

1.2. Contributions

The main results of this paper are Theorems 2–4, and Corollary 1. Theorems 2 and 3 prove that some malware insertion-detection algorithms associated with some hierarchical attacks define particular families of partially ordered sets (posets).

Corollary 1 proves that posets introduced in Theorem 3 define hierarchical attacks without hidden malware.

Theorem 4 proves that malware insertion-detection algorithms give rise to categorical equivalences between categories of representations of posets.

This paper is structured as follows; Main definitions and notation are given in Section 2, we present an overview of definitions and notation regarding malware (Section 2.1) and posets (Section 2.2). We present the main results in Section 3. Section 4 gives an example of the results obtained in Section 3. Concluding remarks are given in Section 5.

2. Preliminaries

This section is devoted to revising basic definitions and notation regarding malware insertion and detection, as well as, partially ordered sets and their $\mathbb{F}$-linear representations [1–4,19–22].

2.1. Malware

As explained in the introduction, malware is malicious software designed to perform some unauthorized, often harmful or undesirable acts [1]. The development of malware research has encouraged the introduction of sophisticated infection-detection malware techniques. Recently discovered computer viruses and worms such as Stuxnet [6] and its variations are examples of the research progress on the subject.

2.1.1. Computer Viruses

The typical structure of a computer virus consists of the following three subroutines [1]:

- *Infect-executable*. This routine finds available executable files to infect them by copying its code.
- *Do-damage* or *Payload*. This is responsible for delivering the malicious part of the virus.
- *Trigger-Pulled*. Determines whether all the conditions required to deliver the payload are satisfied.

In the earlier stages of the antivirus industry, malware detection on computers had as a main goal to create a reliable fingerprint of a malicious file via its heuristic features. For instance,

- Code fragments.
- Hashes of code fragments or the whole file.
- File properties.
- Combinations of these features.

The obtained fingerprint is compared with those stored in an antivirus database. However, malware writers introduced new versions of code virus for which the fingerprint approach is inefficient. Currently, computer viruses include *decryptors* to hide their functionality, encryption keys can be generated in different ways, such as constant, random but fixed, sliding, and shifting, often the encryption is carried out by applying an xor operation (e.g., W95/Memorial virus). However, other encryption techniques dealing with symmetry key cryptography (e.g., the IDEA family of viruses) and public-key cryptography have been used to encrypt viruses. *Polymorphic* and *metamorphic* computer viruses are examples of the use of decryptors.

Polymorphic viruses can mutate their decryptors to a high number of different instances that can take millions of different forms. The 1260 virus is an example of a polymorphic virus, it includes two sliding keys to decrypt its body and some junk instructions, which are nothing but garbage in the code [1].

Metamorphic viruses create new virus generations that look different. They have one single-code body that carries data as code.

Formally speaking a metamorphic virus can be defined as follows [4]:

Let $\Psi_P(d, p)$ be a function computed by a computer program P. Then a pair v and v' of recursive functions are said to be a metamorphic virus if it satisfies the following identities:

$$\Psi_{v(\delta)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p), \\ \Psi_{\delta}(d,p(v'(S(p)))), & \text{if } I(d,p), \\ \Psi_{\delta}(d,p) & \text{otherwise.} \end{cases}$$

and

$$\Psi_{v'(\delta)}(d,p) = \begin{cases} D'(d,p), & \text{if } T(d,p), \\ \Psi_{\delta}(d,p(v'(S'(p)))), & \text{if } I'(d,p), \\ \Psi_{\delta}(d,p) & \text{otherwise.} \end{cases}$$

where (d,p) is a running environment consisting of data d and programs p stored on computers. $D(d,p)$, $D'(d,p)$, and $S(p)$ are recursive functions. Whereas, $T(d,p)$ is called the *injury condition* and $I(d,p)$, $I'(d,p)$ are called *infection conditions*.

The difference between polymorphic and metamorphic viruses is that each form of a polymorphic virus has the same kernel and forms associated with metamorphic viruses have their own kernel.

As an example, the following are two generations of the metamorphic virus W95/Regswap:

1.	5A	pop edx		58	pop, eax
2.	BF**04000000**	mov edi, 0004h		BB**04000000**	mov ebx, 0004h
3.	**8B**F5	mov esi, ebp		**8B**D5	mov edx, ebp
4.	B8**0C000000**	mov eax, 000Ch		BF**0C000000**	mov edi, 000Ch
5.	**81**C2**88000000**	mov add, edx, 0088h		**81**C0**88000000**	mov add, eax, 0088h
6.	**8B**1A	mov ebx, [edx]		**8B**30	mov esi, [eax]
7.	**89**9C86**18110000**	mov [δ]		**89**B4BA**18110000**	mov [δ']
8.	δ =esi+eax*4+00001118, ebx		δ' =edx+edi*4+00001118, esi.		

Figure 1 shows examples of different generations produced by metamorphic viruses.

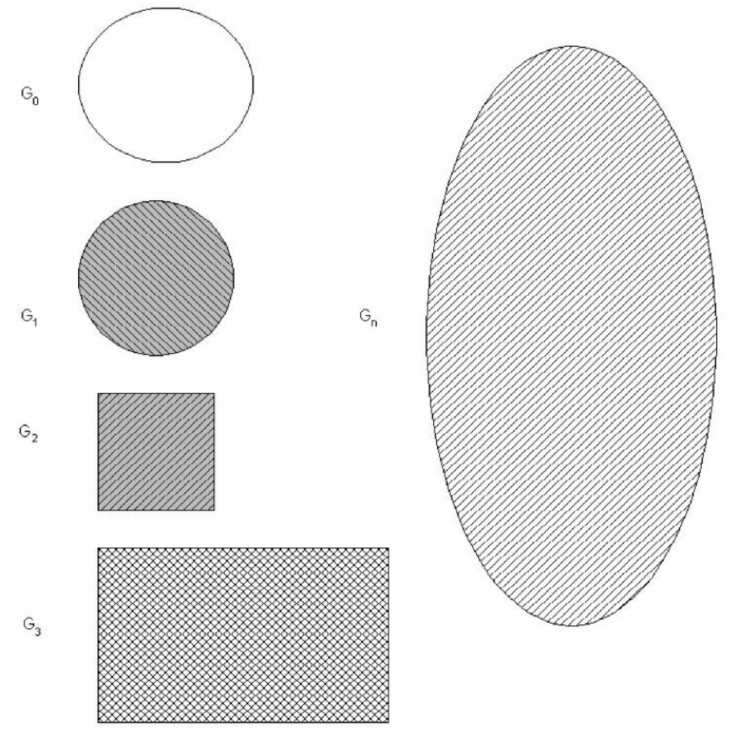

Figure 1. Generations of a complex metamorphic virus [1].

Konstantinou [4] implemented a Hidden Markov Method to detect metamorphic attacks. He implemented (via a virus construction kit) code obfuscation techniques, like instruction reordering and garbage insertion, to produce the metamorphic versions of a virus. We remind the readers that, instruction substitution, instruction permutation, garbage code, variable substitution, and altering control flow are examples of obfuscation techniques. They have been used by viruses and worms as Evol (2000), Zmist, Zperm, Regswap, and Methaphor [1–4].

Some computer worms like Linux/Scalper develop so-called hierarchical attacks to control remote networks. In such a case, each infected node receives crucial information, such as the IP address of the adversary host and the addresses of the infected nodes. This type of information is provided to the remaining nodes until all target network nodes are infected.

Classical approaches to detect malware based on its fingerprint became ineffective due to its vulnerability to zero-day attacks. Recently, Kaspersky [15] implemented machine learning methods to detect packed routines. Their method consists of analyzing file features resistant to small changes. According to this approach, the machine learns suitable hash values $h(x)$ associated with scanned files, and a similarity function is defined to determine whether or not two of these files are similar.

Similar files constitute a so-called *hash bucket*. These hash buckets classify the scanned files into two regions, named *simple regions* or *hard regions*. Files in simple regions of a hash bucket are either pure benign or pure malware, and no further feature analysis is required. Similarity pairs in these regions are of the form $(h(x_1), 0)$ or $(0, h(x_2))$. In hard regions of a hash bucket, the files can be benign and malware, and deep feature analysis is developed for more precise detection. Similarity pairs in hard regions are of the form $(h(x_1), h(x_2))$.

Suppose the infection builds a hierarchical network, as in the case of a scalper attack. Each node contains benign and malware files in simple and hard regions. If vectors consisting of bits are used to denote such files, then a fixed node N_i has a structure $\mathfrak{c}(N_i)$ of the form.

$$\begin{aligned}
\mathfrak{c}(N_i) &= \mathfrak{S}_i \cup \mathfrak{H}_i, \quad 1 \leq i \leq s_i, f_{rs}, g_{ln}, h_{l'n} \in \{0,1\}.\\
\mathfrak{S}_i &= \{f_{1i}, f_{2i}, \ldots, f_{mi}\}.\\
\mathfrak{H}_i &= \{(g_{l_1((t_i+j))}, h_{l'_1((t_i+j))}), (g_{l_1+1((t_i+j))}, h_{l'_1+1((t_i+j))}), \ldots, (g_{l_1+r_i((t_i+j))}, h_{l'_1+r_i((t_i+j))})\},\\
\mathfrak{H}_{i,1} &= \{g_{l_1((t_i+j))}, g_{l_1+1((t_i+j)}, \cdots, g_{l_1+r_i((t_i+j))}\},\\
\mathfrak{H}_{i,2} &= \{h_{l'_1((t_i+j))}), h_{l'_1+1((t_i+j))}), \ldots, h_{l'_1+r_i((t_i+j))})\}.
\end{aligned} \tag{1}$$

where $\mathfrak{S}_i$ ($\mathfrak{H}_i$) denote the set of simple (hard) files contained in N_i. It is assumed that all the files have the same size.

Figure 2 shows the entries of the matrix $\mathfrak{c}(N_i) = \left(\begin{array}{c|c} \mathfrak{S}_i & 0 \\ \hline 0 & \mathfrak{H}_{i,1} \\ \hline 0 & \mathbf{i}\mathfrak{H}_{i,2} \end{array}\right)$ of the node N_i, where $\mathfrak{S}_i$, $\mathfrak{H}_{i,1}$ and $\mathfrak{H}_{i,2}$ are matrix blocks of suitable size associated with files $\mathfrak{S}_i$ and $\mathfrak{H}_i$ (see identities (1)). In this case, we add as many zeroes as possible to satisfy restrictions related to the inclusion of garbage instructions and size of the files. We also assume the notation $g + \mathbf{i}h'$ for each pair of the form (g, h').

Obfuscation techniques as xor and row and column permutations can be applied to the elements of the node $\mathfrak{c}(N_i)$ to obtain new versions of the detected viruses.

A node N_i in a hierarchical attack is said to be *strong* (*weak*) if its files belong to a simple (hard) region. Files in a strong node are either of pure benign type or pure malware type. We let $\odot$ ($\ominus$) denote a strong (*weak*) node in a hierarchical attack.

Henceforth, we will assume that nodes associated with a hierarchical attack have the structure given by the matrix shown in Figure 2.

$$\mathfrak{c}(N_i) = \left(\begin{array}{ccccc|ccccccc} f_{11} & f_{12} & \cdots & \cdots & f_{1s_i} & 0 & 0 & 0 & \cdots & 0 & 0 \\ f_{21} & f_{22} & \cdots & \cdots & f_{2s_i} & 0 & 0 & 0 & \cdots & 0 & 0 \\ f_{31} & f_{32} & & \cdots & f_{3s_i} & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{m1} & f_{m2} & \cdots & & f_{ms_i} & 0 & 0 & \ddots & \cdots & 0 & 0 \\ 0 & 0 & \cdots & \cdots & 0 & g_{l_1((t_i+j))} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & \cdots & 0 & \mathbf{i}h'_{l'_1((t_i+j))} & g_{l_1+1((t_i+j))} & 0 & \cdots & 0 & 0 \\ 0 & 0 & & \cdots & 0 & 0 & \mathbf{i}h'_{l'_1+1((t_i+j))} & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & g_{l_1+r_i((t_i+j))} \\ 0 & 0 & \cdots & & 0 & 0 & 0 & \cdots & \cdots & 0 & \mathbf{i}h'_{l'_1+r_i((t_i+j))} \end{array}\right)$$

Figure 2. Matrix $\mathfrak{c}(N_i)$ associated with a node in a hierarchical attack.

2.1.2. Using Information Theory to Detect and Insert Malware

As we have seen in previous sections, polymorphisms make infeasible static detection of viruses. We remind the reader that there are two kinds of polymorphisms (those obtained by data encryption and those obtained by data compressing). Machine learning methods have been developed to detect different file features such as *N-grams, statistical features, and entropy*. Particularly, entropy features are based on the entropy computation of the file or some of its areas. Bearing in mind that benign files tend to have low entropy values, whereas obfuscated or packed files tend to have high entropy values [23].

Lyda and Hamrock [24] introduced the idea of using entropy (over the entire file) to classify packed malware. It is worth noting that nowadays, distinguishing between packed and non-packed executable files is a strong line of investigation for malware analysts. For instance, Mantovani et al. [23] implemented a machine-learning classifier based on the union of features to identify different forms of packing. Lee et al. [25] used machine learning to recover original files from backup system files (infected with ransomware) via entropy techniques. Perdisci et al. [26] proposed studying specific packer features in the portable executable file format. Whereas, Ugarte-Pedrero et al. [27] suggested that entropy is the main feature of detecting packed files. They used the Zeus botnet, one of the first bot families to adopt low entropy packing schemes.

Raphel et al. [28] used entropy to recognize polymorphic samples which use xor-based encoders. Their approach is based on five steps (extraction of files or appropriated file fragments, computation and concatenation of such fragments, computation of the entropy for concatenated fragments and construction of a suitable similarity distance matrix).

We also recall that Lim et al. [29] proposed to analyze the different files as vectors or streams of bytes to analyze some statistical features.

Entropy has also been used as a helpful feature to insert malicious files. In such a case, the analyst splits a target file into shares or chunks to insert a low entropy pattern of bytes between each share; then, the malicious file is reconstructed in memory to bypass the action of high entropy file detectors. Menéndez et al. [30,31] used the entropy-based tools EnTS and EEE to detect and conceal malicious files into executables. They also used VirusTotal to reproduce the behavior of some anti-virus engines. Detect It Easy (DIE), PEiD, PackerID, NFD, ExeScan, and Manalyze are popular tools to analyze malware. In particular, DIE and PEiD have a component for entropy analysis [23,32].

Nowadays, an interesting problem in cryptography is proving the leakage resilience of cryptographic implementations. Side-channel attacks (SCA) may be one of these implementations' most significant threats [33]. In this kind of attack, a secret key implemented in a device (e.g., a smart card) is retrieved by analyzing the side channel signals obtained from its physical implementation. Low entropy masking schemes (LEMS) have been introduced to guarantee high security against SCA attacks with less randomness than traditional masking schemes. Analysis of these types of schemes has been implemented by Li et al. [34],

who studied leakage characteristics of multiplicative LEMS. Whereas Zhang et al. [35] trained deep learning assisted with a new metric to improve SCA attacks. Security of LEMS has also been studied by Grosso et al. [36], Ye et al. [37], and Zhang et al. [38].

Network security and channel capacity have been studied by Hua et al. [39], Adesso et al. [40], And Yilmaz et al. [41], who introduced a method to estimate the maximum amount of information leakage by some signals generated by the execution of some instructions in a processor.

2.2. Partially Ordered Sets and Their Representations

A *partially ordered set* or *poset* is a pair $(\mathcal{P}, \leq)$, where $\mathcal{P}$ is a possibly empty set endowed with a relation $\leq$, which is

- Reflexive, i.e., $x \leq x$, for any $x \in \mathcal{P}$,
- Antisymmetric, i.e., $x \leq y$ and $y \leq x$ implies $x = y$, for any $x, y \in \mathcal{P}$.
- Transitive, i.e., $x \leq y$ and $y \leq z$ implies $x \leq z$, for any $x, y, z \in \mathcal{P}$.

Henceforth, if there is no confusion, we will write $\mathcal{P}$ instead of the pair $(\mathcal{P}, \leq)$ to denote a poset.

Often, finite posets are described by their Hasse diagram, which is a system of sets with the form $\{\mathcal{P}; \mathfrak{C}_r, \mathfrak{L}\}$, where r is a fixed positive real number (small enough) and for each point $p \in \mathcal{P}$, it is defined a unique point $(x_0, y_0) \in \mathbb{R}^2$ and a unique circle $c \in C_r$ with center (x_0, y_0) and radius r.

The set $\mathcal{L}$ consists of non-horizontal lines connecting circles of C_r, according to the following rule:

- A line l connects two circles c and c' with centers (x_0, y_0) and (x_0', y_0') associated with points p and p' in $\mathcal{P}$ if and only if p and p' is a covering (i.e., if there is $z \in \mathcal{P}$ such that $x \leq z \leq y$ then $x = z$ or $y = z$).

As an example, Figure 3 shows a Hasse diagram of a finite partially ordered set $\mathcal{P} = \{a, b, c, d, e, f\}$ such that $a < d$, $b < d$, $b < e$, $c < e$, and $c < f$.

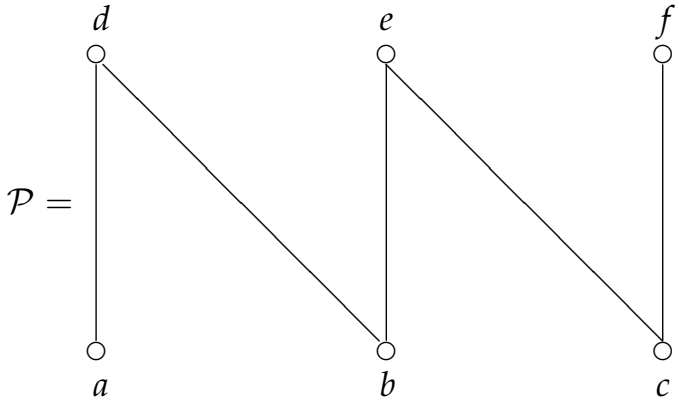

Figure 3. Hasse diagram of the poset $\mathcal{P} = \{a, b, c, d, e, f\}$.

A poset $\mathcal{P}$ is said to be a *chain* if for pair of points $x, y \in \mathcal{P}$, it holds that $x \leq y$ or $y \leq x$ (i.e., any pair of points in a chain are comparable). A poset $\mathcal{P}$ is an antichain if its points are incomparable.

The width $w(\mathcal{P})$ of a poset $\mathcal{P}$ is the size of its largest antichain (e.g., the width of a chain is 1).

If R is a commutative ring and $\mathcal{P}$ is a finite poset then a *$\mathcal{P}$-subspace* U is a system of modules with the form

$$U = (U_0; U_x \mid x \in \mathcal{P}) \tag{2}$$

where U_0 is an R-module, U_x is a submodule of U_0 for any $x \in \mathcal{P}$, and

$$U_x \subseteq U_y \text{ provided that } x \leq y \text{ in } \mathcal{P}. \tag{3}$$

If R is a field then a $\mathcal{P}$-subspace is said to be an *R-linear representation* (or *representation*) of the poset $\mathcal{P}$ [19–22].

If $U = (U_0; U_x \mid x \in \mathcal{P})$ and $V = (V_0; V_x \mid x \in \mathcal{P})$ are representations of a poset $\mathcal{P}$ then their sum $U \oplus V$ is a representation given by the following identity.

$$U \oplus V = (U_0 \oplus V_0; U_x \oplus V_x \mid x \in \mathcal{P}). \tag{4}$$

The representation 0 has 0 as ground vector space. Furthermore, a representation U is said to be *indecomposable* if whenever $U = U_1 \oplus U_2$ then either $U_1 = 0$ or $U_2 = 0$. Otherwise, U is said to be *decomposable*.

A morphism between two representations $U = (U_0; U_x \mid x \in \mathcal{P})$ and $V = (V_0; V_x \mid x \in \mathcal{P})$ is an R-linear map $\varphi : U_0 \to V_0$ such that $\varphi(U_x) \subseteq V_x$. φ is an isomorphism if $\varphi(U_x) = V_x$, for any $x \in \mathcal{P}$.

The composition of morphisms between representations is given by the usual composition of R-linear morphisms. The identity morphism associated with a representation U is denoted 1_U such that if $\varphi : U \to V$ is a morphism then $\varphi 1_U = \varphi = 1_V \varphi$.

We let rep $\mathcal{P}$ denote the category of representations of a poset $\mathcal{P}$, which is a Krull-Schmidt category.

$\underline{\dim}\ U$ denotes the dimension of a representation U of a poset $\mathcal{P}$. It is an integral vector of the form

$$\underline{\dim}\ U = (d_0; d_x \mid x \in \mathcal{P}) \tag{5}$$

where d_0 is the dimension $\dim_R U_0$ of the vector space U_0 as vector space. Whereas, $d_x = \dim_R U_x / \sum_{z \in x_{\blacktriangle}} U_z$, for any $x \in \mathcal{P}$.

$$\begin{aligned} x_{\triangle} &= \{z \in \mathcal{P} \mid z \leq x\}, \quad x_{\blacktriangle} = x_{\triangle} \backslash \{x\}. \\ x^{\triangledown} &= \{z \in \mathcal{P} \mid x \leq z\}, \quad x^{\blacktriangledown} = x^{\triangledown} \backslash \{x\}. \end{aligned} \tag{6}$$

One of the problems regarding the theory of representation of posets consists of giving a complete description of the indecomposable representations of the categories rep $\mathcal{P}$ defined by finite posets $\mathcal{P}$.

Up-to-date, the algorithms of differentiation have been the main tool to classify posets, the algorithm of differentiation with respect to a maximal point introduced by Nazarova and Roiter and the algorithm of differentiation with respect to a suitable pair of points are the most remarkable algorithms to reach such a classification. They are functors with the main goal of reducing the dimension of the posets involved in the classification process.

The following is the definition of the algorithm of differentiation with respect to a suitable pair of points also known as $D - I$ or DI [19]: Let a and b be two points in a finite poset $\mathcal{P}_{(a,b)}$ then the pair (a, b) is said to be *suitable* for DI, if $\mathcal{P}_{(a,b)}$ can be written as a sum of the form

$$\mathcal{P}_{(a,b)} = a^{\triangledown} + b_{\triangle} + C \tag{7}$$

where $C = c_1 < c_2 < \cdots < c_n$ is an n-point chain ($n \geq 0$).

The derived poset $\mathcal{P}'_{(a,b)}$ is a subset of the modular lattice generated by $\mathcal{P}_{(a,b)}$ such that

$$\mathcal{P}'_{(a,b)} = (\mathcal{P}_{(a,b)} + C^+ + C^- + b_{\triangle}) \backslash \{C\} \tag{8}$$

where C^+ and C^- are n-point chains such that $C^+ = c_1^+ < c_2^+ < \cdots < c_n^+$, and $C^- = c_1^- < c_2^- < \cdots < c_n^-$. $c_i^- < c_i^+$ for all $1 \leq i \leq n$, $a < c_1^+$. Points in $\mathcal{P} \backslash \{C\}$ inherit the relations given by $\mathcal{P}_{(a,b)}$. In particular, relations between these points and points c_i^+ and c_i^- are given by the relations between them and points c_i.

Figure 4 shows Hasse diagrams of a poset $\mathcal{P}_{(a,b)}$ with a suitable pair of points and its corresponding derived poset.

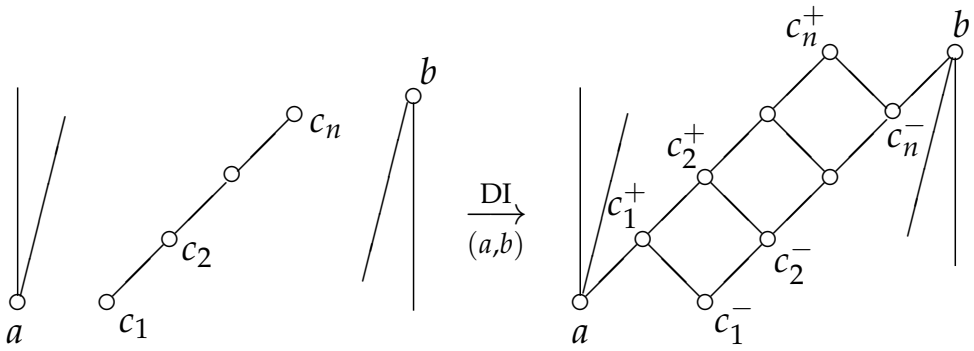

Figure 4. Hasse diagrams of a poset $\mathcal{P}_{(a,b)}$ with a suitable pair of points (a, b) and its corresponding derived poset $\mathcal{P}'_{(a,b)}$.

Differentiation DI or $D_{(a,b)} : \text{rep}\ \mathcal{P}_{(a,b)} \to \text{rep}\ \mathcal{P}'_{(a,b)}$ is defined by the following identities for a representation $U = (U_0; U_x \mid x \in \mathcal{P}_{(a,b)})$:

$$\begin{aligned} D_{(a,b)}(U) = U' &= (U'_0; U'_x \mid x \in \mathcal{P}'_{(a,b)}), \\ U'_0 &= U_0, \\ U'_{c_i^+} &= U_{c_i} + U_a, \\ U'_{c_i^-} &= U_{c_i} \cap U_b, \\ U'_x &= U_x, \text{ for the remaining points } x \in \mathcal{P}'_{(a,b)}, \\ \varphi' &= \varphi \in \text{Hom}_R(U, V), \text{ for any morphism-linear transformation } \varphi : U \to V \in \text{rep}\ \mathcal{P}. \end{aligned} \tag{9}$$

The following theorem is the main result regarding DI. For each i, $1 \leq i \leq n$, $p(a, c_i) = (U_0; U_x \mid x \in \mathcal{P}_{(a,b)})$ is an indecomposable representation for which, $U_0 = R$ is a field. $U_x = R$ is a field, for any $x \in \{a, c_i\}^{\nabla}$. It is zero for the remaining points in the poset.

Theorem 1 (Theorem 5.6, [19]). *The two-point differentiation with completion functor $F_{(a,b)} = C_{(a,b)}D_{(a,b)}$ induces a categorical equivalence between quotient categories*

$$\text{rep}\ \mathcal{P}/\langle p(a, c_1), p(a, c_2), \ldots, p(a, c_n)\rangle \xrightarrow{\sim} \text{rep}\ \overline{\mathcal{P}}'_{(a,b)}/\langle p(a)\rangle. \tag{10}$$

where $\langle p(a, c_1), p(a, c_2), \ldots, p(a, c_n)\rangle$ $(p(a))$ is the ideal consisting of morphisms which pass through direct sums of objects $p(a, c_i)$ $(p(a))$.

The Matrix Problem

The indecomposable representations of a poset $\mathcal{P}$ can be obtained as solutions of a matrix problem. To do that, we note that each representation of $\mathcal{P}$ gives rise to a matrix $M = M_{\mathcal{P}}$ (a *matrix representation*) whose columns are partitioned into strips M_x labeled by the points of the poset. Columns contained in the strip associated M_x consists of coordinates with respect to a fixed basis $\mathcal{B}$ of U_0 of generators of the subspace U_x. In this case, if C_x is the set of columns in the strip M_x then $span\ C_x = U_x$.

If M and M' are matrix representations of a poset $\mathcal{P} = \{x_i \mid 1 \leq i \leq n\}$ with $M = \begin{array}{|c|c|c|}\hline M_{x_1} & \ldots & M_{x_t} \\ \hline \end{array}$, $M' = \begin{array}{|c|c|c|}\hline M'_{x_1} & \ldots & M'_{x_t} \\ \hline \end{array}$ then the direct sum $M \oplus M'$ of M and M' is given by the formula

$$M \oplus M' = \begin{array}{|c:c|c|c:c|}\hline M_{x_1} & 0 & \ldots & M_{x_t} & 0 \\ \hline 0 & M'_{x_1} & \ldots & 0 & M'_{x_t} \\ \hline \end{array}$$

Two representations M and M' are said to be equivalent if one can be obtained from the other using the following admissible transformations:

- Elementary transformations of rows of the whole matrix.

- Elementary column transformations of the columns within each vertical strip.
- Addition of columns of a strip M_{x_i}.

Equivalent matrices give rise to isomorphic representations of the associated poset.

3. Main Results

We remind readers that an equipped poset $\mathfrak{P}$ is a poset whose points define a partition of the form $\mathfrak{P} = \mathfrak{P}^{\ominus} + \mathfrak{P}^{\odot}$. If $x \in \mathfrak{P}^{\odot}$ ($x \in \mathfrak{P}^{\ominus}$) then x is said to be a *strong point* (*weak point*). Relations R in equipped posets are partitioned into two sets $R = \mathfrak{R}^{\ominus} + \mathfrak{R}^{\odot}$, if a pair $(x, y) \in \mathfrak{R}^{\odot}$ ($(x, y) \in \mathfrak{R}^{\ominus}$) then we write $x \trianglelefteq y$ ($x \prec y$). In such a case if $x \preceq y$, i.e., $(x, y) \in R$ and $y \trianglelefteq z$, i.e., $(y, z) \in \mathfrak{R}^{\circ}$ then $x \trianglelefteq z$. Also, if $x \trianglelefteq y \preceq z$ then $x \trianglelefteq z$ [20,22].

We assume that the hierarchical attack (see Figure 2) model satisfies the following additional condition:

1. All the files associated with the malware infecting a network belong to an isolated strong node denoted M.
2. Each infected node x is encoded by finite sets of $\{0,1\}$-vector columns, $\mathfrak{S}_x \cup \mathfrak{H}_x$. Columns in $\mathfrak{S}_x$ encode either benign files or malware. Columns in $\mathfrak{H}_x$ encode hidden malware in hard regions.
3. The files in the malware node M are distributed among a fixed set of weak nodes $N_0, N_1, N_2, \ldots N_n$, where N_0 denotes the initial stage of the infection (hidden malware associated with hard regions are contained in $\mathfrak{H}_0$). $\mathfrak{c}(N_j) = \mathfrak{S}_j \cup \mathfrak{H}_j \subset \mathfrak{S}_{j+1} \cup \mathfrak{H}_{j+1}$, for any $0 \leq j \leq n-1$.
4. If a node P in the attacked network is infected by a node N_j for some $0 \leq j \leq n$ then it holds that $\mathfrak{S}_0 \cup \mathfrak{H}_0 \subset \mathfrak{S}_p$, where P is encoded by $\mathfrak{S}_p \cup \mathfrak{H}_p$. Particularly, if P is also infected by a weak node N_j, it holds that either $\mathfrak{S}_j \cup \mathfrak{H}_j \subset \mathfrak{S}_p \cup \mathfrak{H}_p$ or $\mathfrak{S}_j \cup \mathfrak{H}_j \subset \mathfrak{S}_p$.

The following result proves that a hierarchical attack structured by matrices $\mathfrak{c}(N_i)$ (Figure 2) defines an equipped poset.

Theorem 2. *A hierarchical attack defined by a strong node M as defined above and weak nodes $N_0, N_1, \ldots, N_n$ with the structure given by a matrix $\mathfrak{c}(N_i)$ (see Figure 2) and conditions (1)–(4) defines an equipped poset.*

Proof. We note that nodes in the infected network are the points in the equipped poset $\mathfrak{P}$. Strong nodes correspond to strong points, and weak nodes correspond to weak points in $\mathfrak{P}$. The stages of the infection start in N_0, continue to N_1 and so on. Since, for any pair of weak nodes N_i and N_j, $i < j$ it holds that $\mathfrak{S}_i \cup \mathfrak{H}_i \subset \mathfrak{S}_j \cup \mathfrak{H}_j$ with $H_j \neq \varnothing$ then N_i and N_j are weakly related. Moreover, $N_0, N_1, N_2, \ldots, N_n$ constitute a weak chain, in the sense that its points and relations between them are weak, we write $C = N_0 \preceq N_1 \preceq N_2 \prec \cdots \prec N_n$. Condition (3) proves that relations between weak points N_j and another point in $\mathfrak{P}$ are either weak or strong. Whereas, relations between N_0 and points $x \in \mathfrak{P} \backslash C + M$ are strong. Finally, we note that by definition the strong point M is incomparable with the other points of the poset $\mathfrak{P}$. Therefore, $\mathfrak{P}$ can be written as a sum with the form $\mathfrak{P} = N_0^{\triangledown} + C + M$. Where $N_0^{\triangledown} = \{x \in \mathfrak{P} \mid N_0 \trianglelefteq x\}$. Figure 5 shows an example of an equipped poset induced by a hierarchical attack. Double (single) lines denote strong (weak) relations. In this case, N represents an arbitrary set of infected nodes related to the weak chain $N_0 \preceq N_1 \preceq \cdots \preceq N_n$. □

If N_x is a node infected by a hierarchical attack with files of type f_{ij}, (g_{kl}, h_{mn}) for suitable indexes i, j, k, l, m, n, then $span_{\mathbb{Z}_2}\{f_{ij}, g_{kl)}, h_{mn}\}$ is said to be the *hull* of N_x, we let $\widetilde{U_{N_x}}$ denote the hull of the node N_x. Note that, $span\ \{\mathfrak{S}_x \cup \mathfrak{H}_x\} = U_x \subseteq \widetilde{U_{N_x}}$. $\widetilde{U_{N_x}} = U_{N_x}$ if and only if N_x is strong.

U_x^- denotes the strong subspace $span\{\mathfrak{S}_x\}$ of the subspace U_x. In such a case, $\widetilde{U_x^-} = span_{\mathbb{Z}_2+\mathbf{i}\mathbb{Z}_2}\{\mathfrak{S}_x\}$.

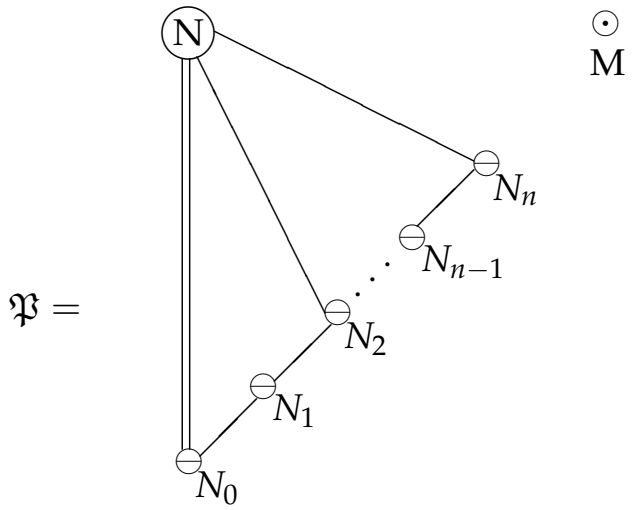

Figure 5. Diagram of an equipped poset $\mathfrak{P}$ induced by a hierarchical attack.

According to the definition of a hierarchical attack and its properties (1)–(4). We note that hidden malware associated with hard regions can be pinpointed by xoring files in U_{N_0} with files in weak nodes U_{N_j}. In such a case, it is built the span sum $\widetilde{U_{N_0}} + U_{N_j}$, $0 \leq j \leq n$.

The detection procedure determines the matrix $\mathfrak{D}$ shown in Figure 6, labeled above by the infected nodes (N_i, M and N) of a network. The bottom part (under the bold line) is labeled by corresponding symbols N_i^-, N_i^+. Such symbols denote subspaces spanned by columns whose entries are elements over $\mathbb{Z}_2 + \mathbf{i}\mathbb{Z}_2$. We let U_x denote the subspace associated with a point x.

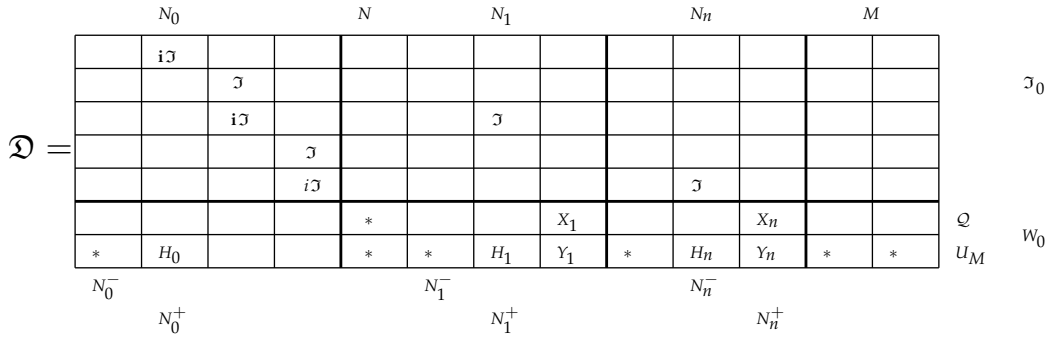

Figure 6. Diagram of an equipped poset $\mathfrak{P}^d$ induced by a malware detection.

$U_{N_{i-1}^-} \subset U_{N_i^-}$, $1 \leq i \leq n$ (these columns are denoted with the symbol $*$), these subspaces encode pure malware (and weak relations) associated with nodes N_i. $U_{N_j^+} \subset U_{N_{j+1}^+}$, $1 \leq j \leq j-1$. Columns associated with symbols H_i encode hidden malware pinpointed by the detection process. Such malware can be inserted into the node N_0 by adding some garbage entries denoted $\mathfrak{I}$ in the matrix $\mathfrak{D}$. Relations between subspaces associated with points $x \in \mathfrak{P} \backslash C = \{N_0 \preceq N_1 \preceq \dots N_n\}$ keep without changes their relations with the other points of $\mathfrak{P}$.

Relations between the infected files allow us to give the next result.

Theorem 3. *The insertion-detection matrix $\mathfrak{D}$ constitute an equipped poset $\mathfrak{P}^d = C^+ + C^- + M + N$. Where, $C^+ = N_0^+ \trianglelefteq N_1^+ \preceq \cdots \preceq N_n^+$ and $C^+ = N_0^- \preceq N_1^- \preceq \cdots \preceq N_n^-$ are chains, M and N and their relations are defined as for the poset $\mathfrak{P}$.*

Proof. By definition, point N_0^+ is a strong point. Furthermore, since files associated with the nodes N_i^- constitute malware satisfying the condition $U_{N_{i-1}^-} \subset U_{N_i^-}$, $1 \leq i \leq n$,

then points N_j^-, $0 \leq j \leq n$ constitute a weak chain. In particular, $N_j^- \preceq M$, for any j. The same argument for subspaces N_j^+ allow us to infer that $N_0^+ \trianglelefteq N_1^+$. Since $N_i^+ \preceq N_{i+1}^+$, $1 \leq i \leq n-1$, it holds that $N_0^+ \trianglelefteq N_i^+$. Moreover, $N_j^- \preceq N_j^+ \preceq N_{j+1}^+$, for any $0 \leq j \leq n-1$, $N_n^- \preceq N_n^+$. Since relations between points N_j^-, N_j^+ and points in subset $N \cup \{M\}$ are inherited by the relations that these points have with points $N_0, \ldots, N_n$. The following Figure 7 shows the poset $\mathfrak{P}^d$ defined by the insertion-detection matrix $\mathfrak{D}$. □

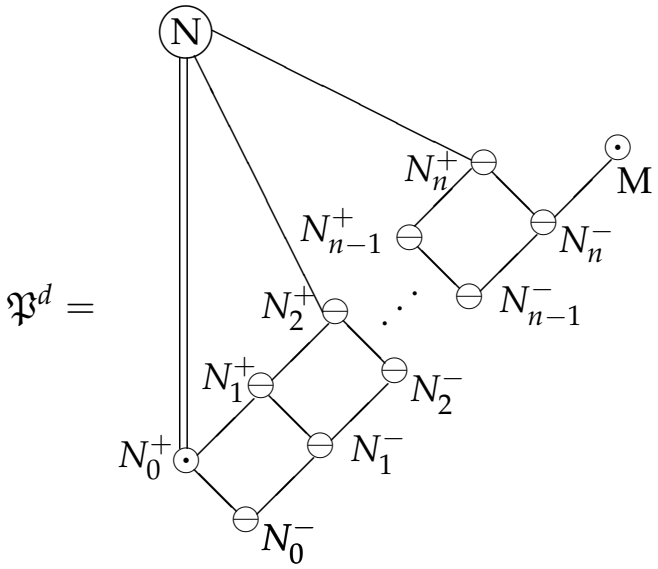

Figure 7. Diagram of an equipped poset $\mathfrak{P}^d$ induced by a malware detection.

Corollary 1. *The hierarchical attack defined by an equipped poset of type $\mathfrak{P}^d$ has no hidden malware.*

Proof. The malware used in this type of attack is encoded by subspaces $U_{N_j^-}$ and H_j which are induced by simple regions. □

In a more general setting, we can define a functor $\mathcal{D}_{(N_0,M)}$ induced by a hierarchical attack defined by an equipped poset of type $\mathfrak{P}$ and its associated detection algorithm defined by a corresponding equipped poset $\mathfrak{P}^d$. The following Figure 8 shows the poset $\mathfrak{P}$ and its detector $\mathfrak{P}^d$.

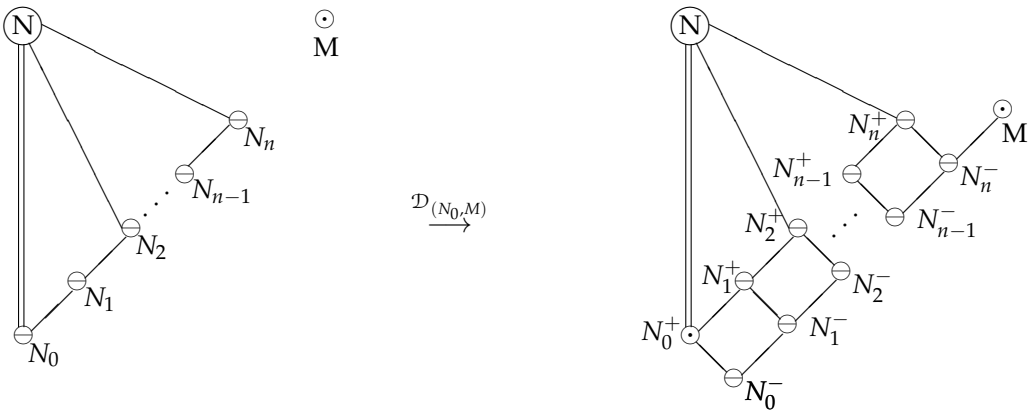

Figure 8. Diagrams of hierarchical attacks with and without hidden malware.

If we replace the field $\mathbb{Z}_2$ for the real numbers field and $\mathbb{Z}_2 + \mathbf{i}\mathbb{Z}_2$ for the complex numbers field. Then $(\mathbb{R}, \mathbb{C})$-column transformations between rows and columns of the matrices induced by the linear structure of posets $\mathfrak{P}$ and $\mathfrak{P}^d$ give rise to categories of

representations of the equipped posets $\mathfrak{P}$ and $\mathfrak{P}^d$. In such a case, a representation U of an equipped poset $\mathfrak{P}$ is a system of $\mathbb{C}$-subspaces of the form $U = (U_0, U_x \mid x \in \mathcal{P})$, with $U_x \subseteq U_y$ ($\widetilde{U_x} \subset U_y^-$) provided that $x \preceq y$ ($x \trianglelefteq y$).

A morphism $\varphi : U \to V$ between two representations U and V of an equipped poset $\mathcal{P}$ is a $\mathbb{C}$-linear transformation such that $\widetilde{\varphi}(U_x) \subset V_x$. Note that, $\widetilde{\varphi}(u + \mathbf{i}v) = \varphi(u) + \mathbf{i}\varphi(v)$, for any pair of appropriated vectors. φ is an isomorphism if and only if $\widetilde{\varphi}(U_x) = V_x$ for any $x \in \mathfrak{P}$.

Each representation U over the pair of fields $(\mathbb{R}, \mathbb{C})$ of an equipped poset can be represented by a matrix $\mathfrak{M}$ with entries over $\mathbb{C}$ separated into vertical strips $(\mathfrak{M}_x; x \in \mathfrak{P})$ labeled by the points of $\mathfrak{P}$. Columns in $\mathfrak{M}_x$ are generators of U_x.

The matrix problem associated with an equipped poset $\mathfrak{P}$ is defined as follows:

Two matrix representations of an equipped poset are said to be equivalent, if one can be obtained from the other via the following admissible transformations:

- Elementary transformations over $\mathbb{C}$ of rows of whole matrix.
- Elementary column transformations over $\mathbb{C}$ within each vertical strip.
- Additions of columns of a strip M_x to the columns of M_y if $x \preceq y$.
- Independent additions of the real and imaginary part of the columns of a strip M_x to the real and imaginary parts of a strip M_y if $x \trianglelefteq y$.

Note that, if $\mathfrak{P} = c_1 \prec c_2 \prec \dots c_{n-1} \prec c_n$ is a weak chain then $\varnothing$, $\mathfrak{P}(c_i)$, $1 \leq i \leq n$, $T(c_i)$, and $T(c_i, c_j)$, $1 \leq i < j \leq n$ are its only indecomposable representations, where

- $P(c_i) = (\mathbb{C}; (P(c_i))_x \mid x \in \mathfrak{P})$, $(P(c_i))_x = \mathbb{C}$, $x = c_j$, $i \leq j$, $(P(c_i))_x = 0$, if $j < i$.
- $P(\varnothing) = (\mathbb{C}; (P(c_i))_x = 0 \mid x \in \mathfrak{P})$.
- $T(c_i) = (\mathbb{C}; (T(c_i))_x \mid x \in \mathfrak{P})$, $(T(c_i))_x = span\{(1, \mathbf{i})^t\}$, $x = c_j$, $i \leq j$, $(T(c_i))_x = 0$, if $j < i$.
- $T(c_i, c_j) = (\mathbb{C}; (T(c_i, c_j))_x \mid x \in \mathfrak{P})$, $(T(c_i, c_j))_x = span\{(1, \mathbf{i})^t\}$, $x = c_s$, $i \leq s < j$, $(T(c_i, c_j))_x = \widetilde{\mathbb{C}} = span\{(1, 0)^t, (0, 1)^t\}$, if $j \leq s \leq n$. $(T(c_i, c_j))_x = 0$, if $s < i$.

Theorem 4. *The insertion-detection matrix $\mathfrak{D}$ defined over the pair of fields $(\mathbb{R}, \mathbb{C})$ associated with the equipped posets $\mathfrak{P}$ and $\mathfrak{P}^d$ induces the functor $\mathfrak{D}_{(N_0,M)} : \operatorname{rep} \mathfrak{P} \to \operatorname{rep} \mathfrak{P}^d$ such that for $U = (U_0; U_x \mid x \in \mathfrak{P}) \in \operatorname{rep} \mathfrak{P}$ it holds that*

$$\begin{aligned} \mathfrak{D}_{(N_0,M)}(U) &= (U_0^d; U_x^d \mid x \in \mathfrak{P}), \\ U_0^d &= U_0, \\ U_{N_i^+}^d &= \widetilde{U_{N_0}} + U_{N_i}, \quad 0 \leq i \leq n, \\ U_{N_i^-}^d &= U_{N_i} \cap U_M, \quad 0 \leq i \leq n, \\ U_x^d &= U_x, \quad \textit{for the remaining points } x \in \mathfrak{P}, \\ \varphi^d : U^d \to V^d &= \varphi : U \to V, \quad \textit{for any linear map-morphism } \varphi : U_0 \to V_0. \end{aligned} \tag{11}$$

Moreover, $D_{(N_0,M)}$ is a categorical equivalence between the quotient categories $\mathfrak{C} = \operatorname{rep} \mathfrak{P}/\mathcal{J}$ and $\mathfrak{C}^d = \operatorname{rep} \mathfrak{P}^d/\mathcal{J}^d$. Where, for fixed $U, V \in \operatorname{rep} \mathfrak{P}$, $\mathcal{J}$ is the ideal of $\operatorname{rep} \mathfrak{P}$ consisting of morphisms $\varphi : U \to V$ that pass through direct sums of the indecomposable objects $P(N_0)$, $T(N_0)$, and $T(N_0, N_i))$, i.e., $\mathcal{J} = \langle P(N_0), T(N_0), T(N_0, N_i)) \mid 1 \leq i \leq n \rangle$. The ideal $\mathcal{J}^d$ is defined in the same fashion, i.e., $\mathcal{J}^d = \langle N_0^+ \rangle$.

Proof. Firstly, we note that $\mathfrak{D}_{(N_0,M)}^d$ is an additive functor provided that for all morphisms $\varphi : U \to V$ and $\psi : V \to W$, it holds that, $\mathfrak{D}_{(N_0,M)}^d(\psi\varphi)(U_x) \subseteq W_x$, for any $x \in \mathfrak{P}$, $D_{(N_0,M)}^d(1_U) = 1_{U^d}$, and for any $U, V \in \operatorname{rep} \mathfrak{P}$, $Hom(U, V)$ is a $\mathbb{C}$-vector space by definition.

Note that, for fixed $U, V \in \text{rep}\,\mathfrak{P}$, it holds that, $J(U,V) \subset Hom(U,V) \subseteq Hom(U^d, V^d)$ and $\mathcal{J}(U,V) \subset \mathcal{J}^d(U,V)$. Moreover, if $[X,Y]$ denotes the morphism-subspace of $Hom(U,V)$ whose elements satisfy the condition

$$\varphi \in [X,Y] \text{ if and only if } X \supseteq \ker\varphi \quad \text{and} \quad \text{img}\,\varphi \subset Y. \tag{12}$$

Then it is easy to see that for fixed $U, V \in \text{rep}\,\mathfrak{P}$, and a morphism $\varphi : U \to V$ it holds that

$$\begin{aligned} Hom(U^d, V^d) &= Hom(U,V) + \mathcal{J}(U^d, V^d), \\ Hom(U,V) \cap \mathcal{J}(U^d, V^d) &= \mathcal{J}(U,V). \end{aligned} \tag{13}$$

Thus, any linear morphism $\delta : \mathfrak{C}(U,V) \to \mathfrak{C}^d(U^d, V^d)$ is an isomorphism.

The density of the functor $\mathfrak{D}_{(N_0,M)}$ follows from the same ideas used to carry out a hierarchical attack to the network defined by the equipped poset $\mathfrak{P}$. In such a case, we consider that $U^d_{N_0^+} = U^d_{N_0^+} \cap U_M \oplus X_0$, where X_0 is complementary subspace, $X_0 = span\{z_1, z_2, \ldots, z_s\}$. For these vectors we define corresponding vectors $w_1, w_2, \ldots, w_s$.

We note that for each i, $1 \leq i \leq n$. Any subspace $U^d_{N_i^+}$ of the poset $\mathfrak{P}^d \cup \{N_0^+ \trianglelefteq M\}$ can be written in the form

$$U^d_{N_i^+} = U^d_{N_{i-1}^+} \oplus U^d_{N_i^-} \oplus H_i \oplus Y_i \tag{14}$$

where Y_i denotes an appropriated complementary subspace. Note that, $H_i \subset U_{N_0^+} \cap U_M$ (corresponds to hidden malware) and $U_{N_i^-} \subset U_M$ (corresponds to pure malware).

Let $\{h_{i1}, h_{i2}, \ldots, h_{in_i}\}$ be a fixed basis then it is possible to define vectors of the form $e_{i1} + \mathbf{i}h_{i1}, \ldots, e_{in_i} + \mathbf{i}h_{in_i}$, for some suitable vectors $e_{i1}, \ldots, e_{in_i}$.

Let $\mathfrak{I}_0 = span\{w_1, \ldots, w_s, e_{i1}, \ldots, e_{in_i} \mid 1 \leq i \leq n\}$ and $\{y_{i1}, \ldots, y_{im_i}\}$ a basis of the subspace Y_i then the representation $L \in \text{rep}\,\mathfrak{P}$ such that

$$\begin{aligned} L_0 &= U_0^d \oplus \mathfrak{I}_0, \\ L_{N_0} &= U^d_{N_0^+} \cap U_M \oplus span\{z_1 + \mathbf{i}w_1, z_2 + \mathbf{i}w_2, \ldots, z_s + \mathbf{i}w_s\} \oplus \sum_{j=1}^{n_i}\sum_{i=1}^{n} e_{ij} + \mathbf{i}h_{ij}, \\ L_{N_i} &= U_{N_{i-1}} \oplus U^d_{N_i^-} \oplus H_i \oplus \sum_{j=1}^{n_i}\sum_{i=1}^{n} e_{ij} + y_{ij}, \\ L_x &= U_x^d \quad \text{for the remaining points } x \in \mathfrak{P}. \end{aligned} \tag{15}$$

is such that $L^d = U^d \oplus (P(N_0^+))^{\dim_{\mathbb{C}}\mathfrak{I}_0}$. We are done. □

4. Experimental Data

This section applies Theorems 3 and 4 and Corollary 1 to insert and detect images. Firstly, we show a 256×256 original image I extracted from specialized datasets such as FERET and Kagle. We then create subspaces associated with a poset $\mathcal{P} = N_0 \prec N_1 \prec N_2 \prec N_3 \prec N_4 \prec N_5 \cup \{M\}$ as follows:

- The subspace U_{N_0} associated with the weak point N_0 is given by a linear combination of images with the form

$$U_{N_0} = \sum_{i=0}^{9} DW_{(0,i)} + A_{10} + 0.01M_{10} \tag{16}$$

where $DW_{(0,i)} = \alpha_i A_i + \beta_i M_i$, $0.01 \leq \alpha_i, \beta_i \leq 0.02$.

- For $1 \leq j \leq 5$, each subspace U_{N_j} (associated with a weak point N_j) is given by linear combinations of images with the form

$$U_{N_j} = \gamma_j N_{j-1} + \delta_j M_{10+j} + A_{10+j}, \quad 0.01 \leq \gamma_j, \delta_j \leq 0.02. \tag{17}$$

  The embedded images M_j, $0 \leq j \leq 15$ span the subspace U_M (associated with the strong point M). They are considered malware for images A_j, the construction of the subspace U_{N_j} is considered the infection stage.
- For the detection process, we note that the subspaces $U_{N_j^-}$ are given by the images M_{10+j} (i.e., $U_{N_j^-} = span\{M_{10+j}\}$), $1 \leq j \leq 5$. These constructions constitute the first step for the detection process.
- The second step of the detection process consists on building subspaces $U_{N_j^+}$ given by linear combinations of images with the form

$$U_{N_j^+} = \sum_{i=0}^{10+j} \alpha_j A_j + \sum_{i=0}^{10+j} \delta_j M_j. \tag{18}$$

  If $t \in \{0, \ldots, 5\}$ and for $0 \leq j \neq t \leq 5$, it holds that $0.01 \leq \delta_j \leq 0.02$, $\delta_t = 1$ then $U_{N_j^+}$ reveals M_t as a kind of malware infecting the image, A_j.

Figures 9–14 show examples of images A_j (original images), associated with subspaces U_{N_j}, $U_{N_j^-}$ (first step), and $U_{N_j^+}$ (denoted H_j in the second step) for $0 \leq j \leq 5$. We compare the associated histograms. We note that the histograms associated with the second step suggest embedded malware.

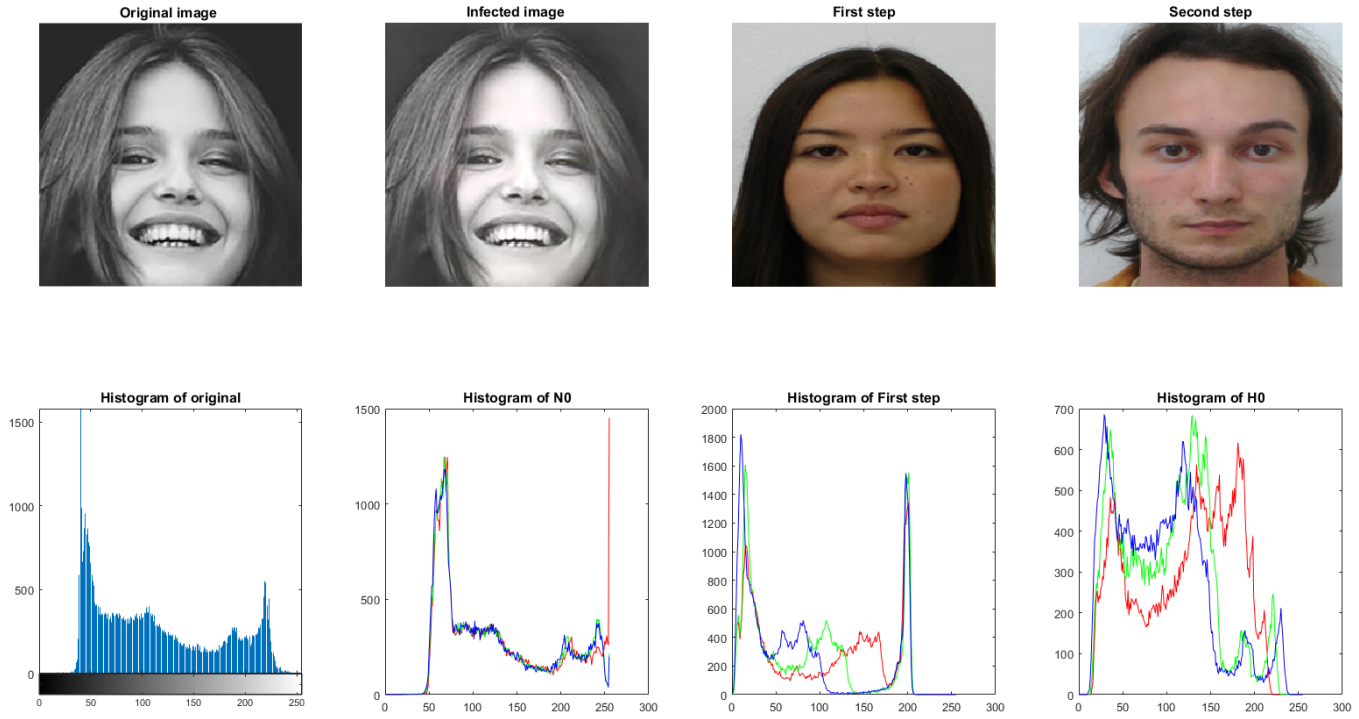

Figure 9. Images associated with the subspace N_0, in the first step of the malware detection process, we extract simple malware of type M_j generating subspaces $U_{N_j^-}$. In the second step, the algorithm extracts hidden (hard) malware M_j defining subspaces $U_{N_j^+}$ also denoted H_j.

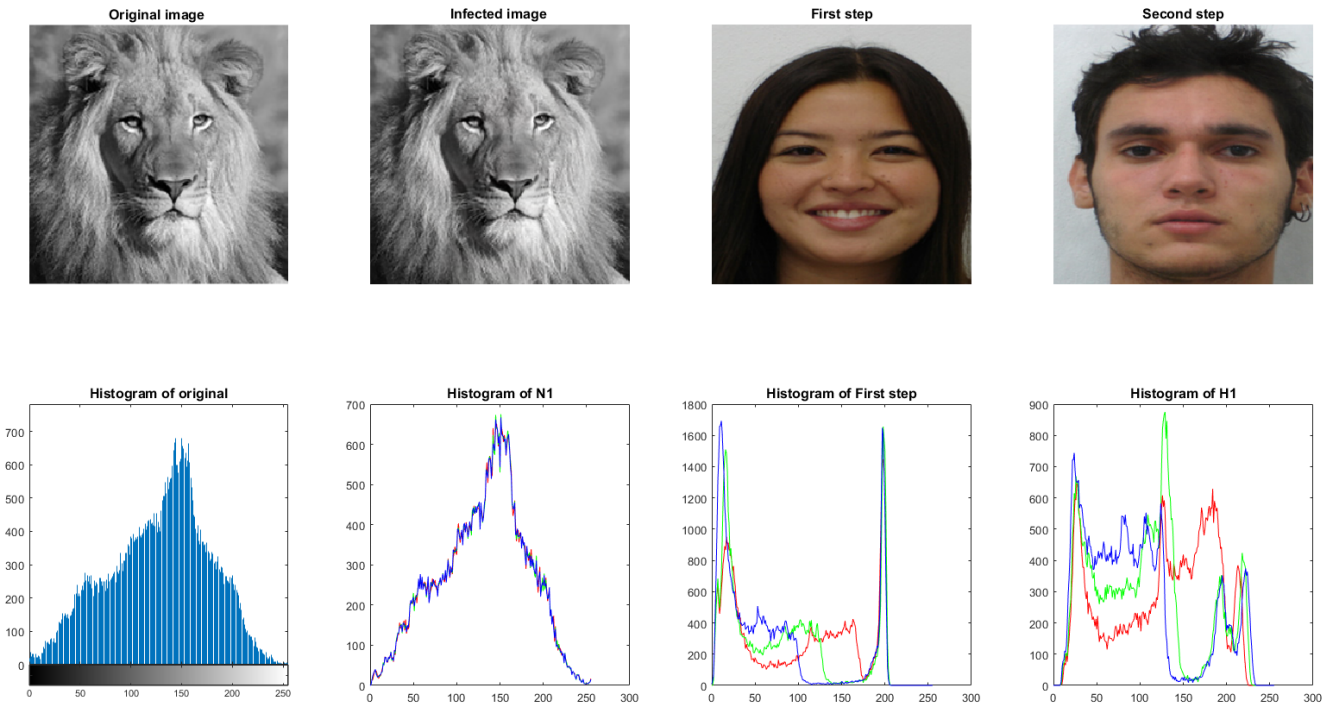

Figure 10. Images associated with subspaces U_{N_1}, $U_{N_1^-}$, and $U_{N_1^+}$.

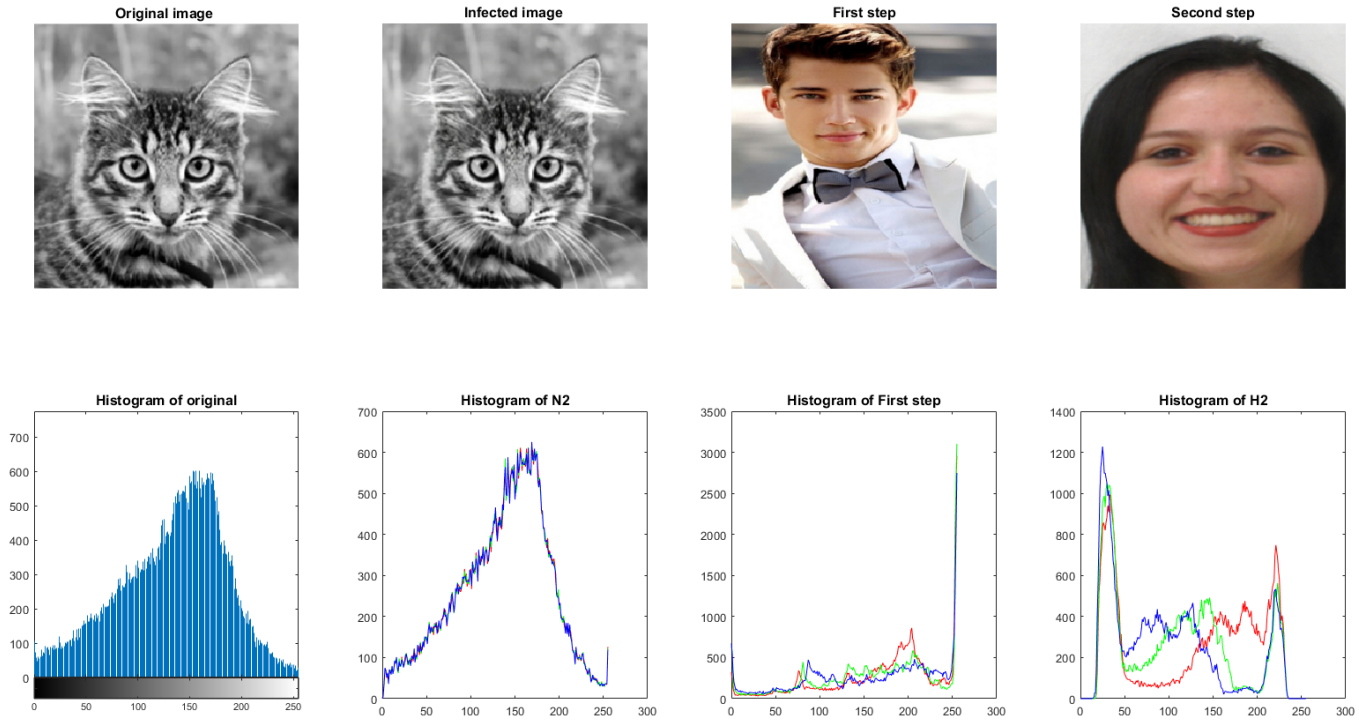

Figure 11. Images associated with subspaces U_{N_2}, $U_{N_2^-}$, and UN_2^+.

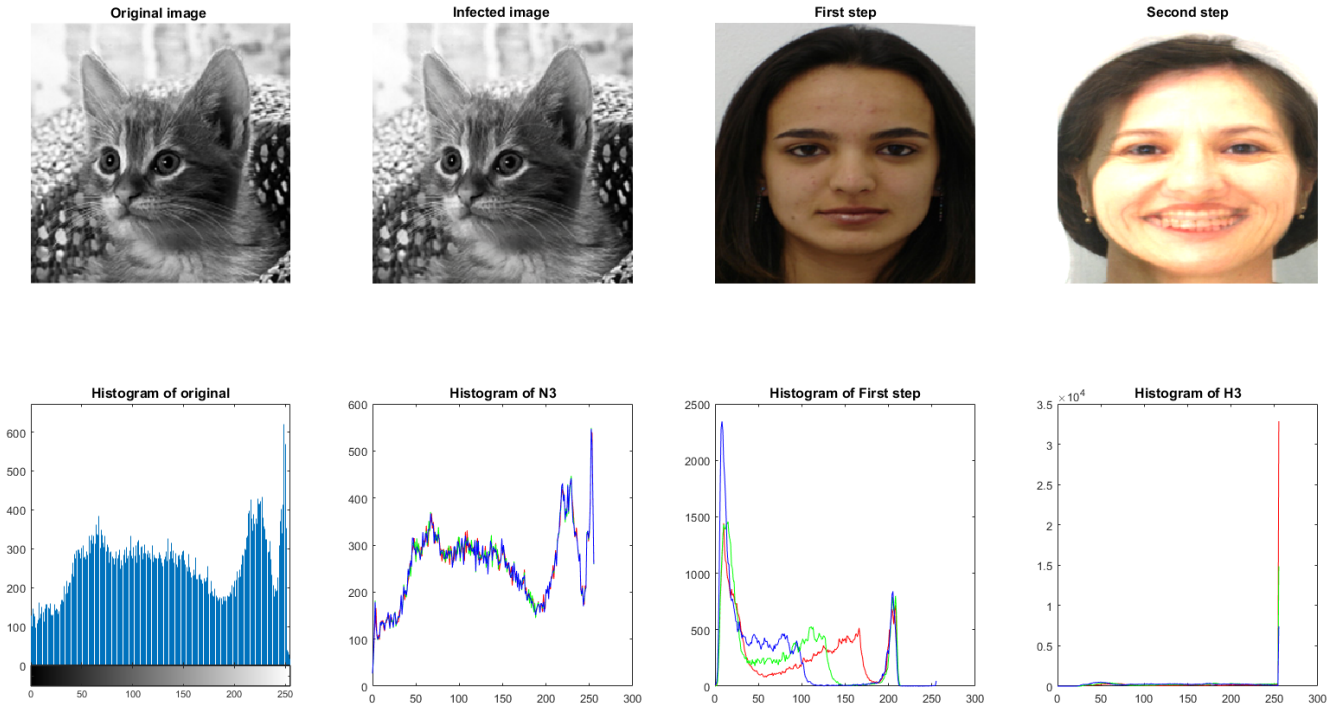

Figure 12. Images associated with subspaces U_{N_3}, $U_{N_3^-}$, and $U_{N_3^+}$.

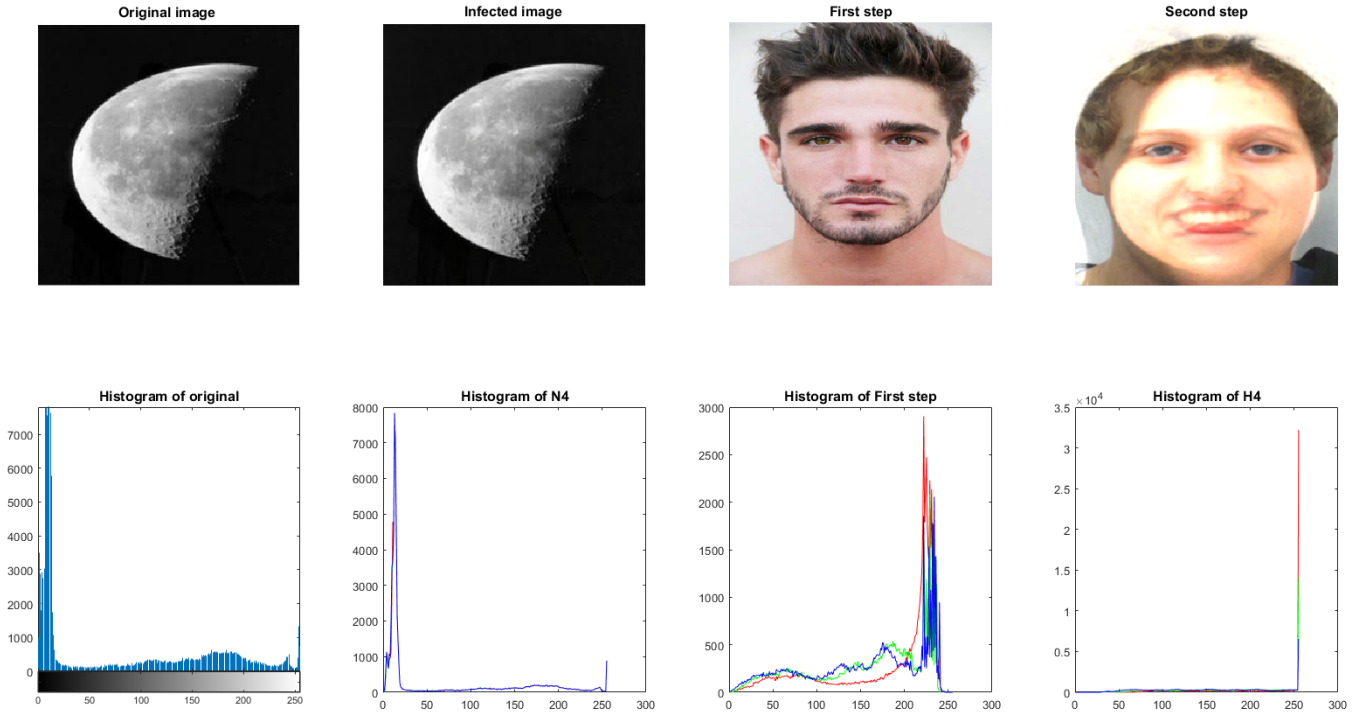

Figure 13. Images associated with subspaces U_{N_4}, $U_{N_4^-}$, and $U_{N_4^+}$.

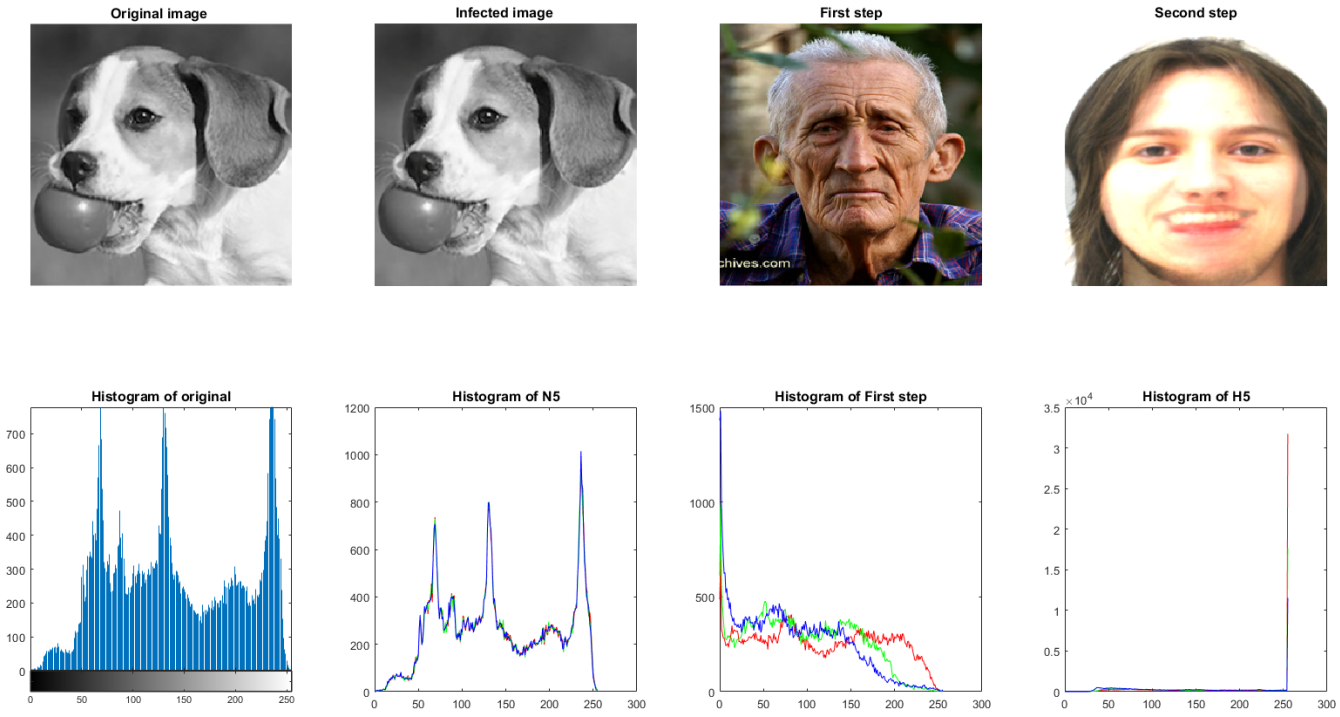

Figure 14. Images associated with subspaces U_{N_5}, $U_{N_5^-}$, and $U_{N_5^+}$.

5. Concluding Remarks and Future Work

Hierarchical attacks designed for peer-to-peer remote control via metamorphic worms induce different algebraic structures. On the one hand, the infection process defines so-called equipped posets. These posets constitute a mathematical model of a hierarchical attack where the nodes are either weak or strong, accordingly of whether the node represents an infection with either hidden malware or pure malware. Pure malware is relatively easy to detect, whereas hidden malware requires deep scanning analysis. Modeling such an analysis gives rise to categories of representations of equipped posets over the pair of fields $(\mathbb{R}, \mathbb{C})$, and malware insertion-detection defines a categorical equivalence between quotient categories.

Future Work

Since this work focuses on the algebraic properties of hierarchical attacks, it remains an open problem to determine the properties associated with more general types of infections and NIDS based on deep learning algorithms.

Another task to develop in the future is to apply the proposed theoretical framework to the real field of the intrusion and detection of malware.

Author Contributions: Investigation, writing, review and editing, A.M.C., O.M.M., J.D.C.V. All authors have read and agreed to the published version of the manuscript.

Funding: Center of Excellence in Scientific Computing (CoE-SciCo) Universidad Nacional de Colombia.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

$\mathbb{R}$	(Real numbers)
$\mathbb{C}$	(Complex numbers)
DI	(Algorithm of differentiation with respect to a suitable pair of points)
Poset	(Partially Ordered Set)
$\mathfrak{P}$	(Equipped poset)
$\odot$	(Strong point)
$\ominus$	(Weak point)
$\|\|$	(Strong relation in an equipped poset)

References

1. Szor, P. *The Art of Computer; Virus Research and Defense*; Pearson Education Inc.: Hoboken, NJ, USA, 2005.
2. Venkatachalam, S. Detecting Undetectable Computer Viruses. Master's Thesis, San José State University, San José, CA, USA, 2010; Volume 156.
3. Alzarooni, K.M.A.Y. Malware Variant Detection. Ph.D. Thesis, University College London, London, UK, 2012.
4. Konstantinou, E. *Metamorphic Virus: Analysis and Detection*; Technical Report; Royal Holloway, University of London: London, UK 2008.
5. Cohen, F.B. *A Short Course on Computer Viruses*; Wiley Professional Computing: New York, NY, USA, 1994.
6. Matrosov, A.; Rodionov, E.; Harley, D.; Malcho, J. Stuxnet under the microscope. *ESET LLC* **2010**, *6*, 1–85.
7. Ploszek, R.; Švec, P.; Debnár, P. Analysis of encryption schemes in modern ransomware. *Rad Hazu Maematičke Znanosti* **2021**, *25*, 1–13.
8. Cannarile, A.; Carrera, F.; Galantucci, S.; Iannacone, A.; Pirlo, G. A study on malware detection and classification using the analysis of API calls sequences through shallow learning and recurrent neural networks. In Proceedings of the TASEC'22: Italian Conference on Cybersecurity, Rome, Italy, 20–23 June 2022; Volume 3260, pp. 1–11.
9. Amer, E.; Zelinka, I. A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence. *Comput. Secur.* **2020**, *92*, 1–15. [CrossRef]
10. Hu, W.; Tang, Y. Black-box attacks against RNN based malware detection algorithms. In Proceedings of the AAAI Workshops, New Orleans, LA, USA, 2–7 February 2018; pp. 245–251.
11. He, K. Malware Detection with Malware Images using Deep Learning Techniques. Bachelor's Thesis, University of Canterbury, Canterbury, UK, 2018.
12. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In *VizSec '11: Proceedings of the 8th International Symposium on Visualization for Cyber Security*; ACM: Pittsburg, PA, USA, 2011; pp. 1–7.
13. Iglesias Perez, S.; Criado, R. Increasing the effectiveness of network intrusion detection systems (NIDSs) by using multiplex networks and visibility graphs. *Mathematics* **2023**, *11*, 107. [CrossRef]
14. Kumar, J.; Subbiah, G. Zero-day malware detection and effective malware analysis using shapley ensemble boosting and bagging approach. *Sensors* **2022**, *22*, 2798. [CrossRef]
15. Kaspersky Enterprise Cybersecurity. Machine Learning for Malware Detection. 2017. Available online: media.kaspersky.com (accessed on 7 June 2023).
16. Tayyab, U.-E.-H.; Khan, F.B.; Durad, M.H.; Khan, A.; Lee, Y.S. A Survey of the Recent Trends in Deep Learning Based Malware Detection. *J. Cybersecur. Priv.* **2022**, *2*, 800–829. [CrossRef]
17. Aslan, Ö.A.; Samet, R. A comprehensive review on malware detection approaches. *IEEE Access* **2020**, *8*, 1–23. [CrossRef]
18. Webster, M.; Malcom, G. Detection of metamorphic and virtualization-based malware using algebraic specification. *J. Comp. Virol.* **2009**, *5*, 221–245. [CrossRef]
19. Zavadskij, A.G. On Two Point Differentiation and its Generalization. *Algebr. Struct. Their Represent. AMS Contemp. Math. Ser.* **2005**, *376*, 413–436.
20. Zavadskij, A.G. Tame equipped posets. *Linear Algebra Appl.* **2003**, *365*, 389–465. [CrossRef]
21. Cañadas, A.M.; Gaviria, I.D.M. Categorical Properties of Some Algorithms of Differentiation for Equipped Posets. *Algebra Discret. Math.* **2022**, *33*, 38–86.
22. Cañadas, A.M.; Vargas, V.C. On the apparatus of differentiation DI-DV for posets. *São Paulo J. Math. Sci.* **2019**, *9*, 249–286. [CrossRef]
23. Mantovani, A.; Aonzo, S.; Ugarte-Pedrero, X.; Merlo, A.; Balzarotti, D. Prevalence and impact of low-entropy packing schemes in the malware ecosystem. In *Network and Distributed Systems Security (NDSS) Symposium*; NDSS: San Diego, CA, USA, 2020; pp. 1–15.
24. Lyda, R.; Hamrock, J. Using entropy analysis to find encrypted and packed malware. *IEEE Secur. Priv.* **2007**, *5*, 40–45. [CrossRef]
25. Lee, K.; Lee, S.-Y.; Yim, K. Machine learning based file entropy Analysis for ransomware detection in backup systems. *IEEE Access* **2019**, *7*, 110205–110215. [CrossRef]

26. Perdisci, R.; Lanzi, A.; Lee, W. Classification of packed executables for accurate computer virus detection. *Pattern Recognit. Lett.* **2008**, *29*, 1941–1946. [CrossRef]
27. Ugarte-Pedrero, X.; Santos, I.; Sanz, B.; Laorden, C.; Bringas, P.G. Countering entropy measure attacks on packed software detection. In Proceedings of the Consumer Communications and Networking Conference (CCNC) Las Vegas, NV, USA, 14–17 January 2012; pp. 164–168.
28. Raphel, J.; Vinod, P. Information theoretic method for classification of packed and encoded files. In Proceedings of the 8th International Conference on Security of Information and Networks, SIN'15, Sochi, Russia, 8–10 September 2015; ACM: New York, NY, USA, 2015; pp. 296–303.
29. Lim, C.; Ramli, K.; Cheng, W.; Kotualubun, Y.S. Mal-flux: Rendering hidden code of packed binary executable. *Digit. Investig.* **2019**, *28*, 83–95. [CrossRef]
30. Menéndez, H.D.; Bhattacharya, S.; Clark, D.; Barr, E.T. The arms race: Adversarial search defeats entropy used to detect malware. *Expert Syst. Appl.* **2019**, *118*, 246–260. [CrossRef]
31. Menéndez, H.D.; Llorente, J.L. Mimicking anti-viruses with machine learning and entropy profiles. *Entropy* **2019**, *21*, 513. [CrossRef] [PubMed]
32. Chen, S.-W.; Chuang, T.-H.; Tien, C.-W.; Chen, C.-W. An experience in enhancing machine learning classifier against low-entropy packed malwares. *Comput. Sci. Inf. Technol.* **2021**, *11*, 4.
33. Cheng, W.; Guilley, S.; Carlet, C.; Danger, J.L.; Mesnager, S. Leakages in code-based masking: A unified quantification approach. *Iacr Trans. Cryptogr. Hardw. Embed. Syst.* **2021**, *2021*, 465–495. [CrossRef]
34. Li, Y.; Liu, S.; Guilley, S.; Tang, M. Analysis of multiplicative low entropy masking schemes against correlation power attack. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4466–4481. [CrossRef]
35. Zhang, Z.; Ding, A.A.; Fei, Y. A guessing entropy-based framework for deep learning-assisted side-channel analysis. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3018–3030. [CrossRef]
36. Grosso, V.; Standaert, FX.; Prouff, E. Low entropy masking schemes, Revisited. In *Smart Card Research and Advanced Applications*; CARDIS 2013; Lecture Notes in Computer Science; Fr, A., Rohatgi, P., Eds.; Springer: Cham, Switzerland, 2014; Volume 8419.
37. Ye, X.; Eisenbarth, T. On the vulnerability of low entropy masking schemes. In Proceedings of the Smart Card Research and Advanced Application Conference, Berlin, Germany, 27–29 November 2013.
38. Zhang, Z.; Dofe, J.; Yu, Q. Improving power analysis attack resistance using intrinsic noise in 3D ICs. *Integration* **2020**, *73*, 30–42. [CrossRef]
39. Hua, J.; Zhou, Z.; Zhong, S. Flow misleading: Worm-hole attack in software-defined networking via building in-band covert channel. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 1029–1043. [CrossRef]
40. Adesso, P.; Cirillo, M.; Di Mauro, M.; Matta, V. ADVoIP: Adversarial detection of encrypted and concealed VoIP. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 943–958. [CrossRef]
41. Yilmaz, B.B.; Callan, R.L.; Prvulović, M.; Zajić, A.G. Capacity of the EM covert/side-channel created by the execution of instructions in a processor. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 605–620. [CrossRef]