Prince Hussein Bin Abdullah College for Information Technology
Computer Science Department

Enhancing the performance of Intrusion Detection System by minimizing the false alarm detection using fuzzy logic

تحسين أداء أنظمة كشف التسلل وذلك بتقليص عدد الإنذارات الكاذبة بإستخدام المنطق الضبابي

By:
Mohammed O. Alshroqee
Supervisor:
Dr. KhaledBatiha

This Thesis was submittedin Partial Fulfillment of the Requirements For the Master's Degree of Science in Computer Science

Deanship of Graduate Studies

Al Al-Bayt University

2015

تفويض

انا محمد عمر عبدالغني الشروقي، أفوض جامعة آل البيت بتزويد نسخ من رسالتي للمكتبات او المؤسسات او الهيئات او الاشخاص عند طلبهم حسب التعليمات النافذة في جامعة آل البيت.

التوقيع:

التاريخ:

اقرار والتزام بقوانين جامعة آل البيت وأنظمتها وتعليماتها

أناالطالب :محمد عمر عبدالغني الشروقي الرقم الجامعي : 1220901006
التخصص :علم حاسوب الكلية :الأمير الحسين بن عبدالله لتكنولوجيا المعلومات

اعلن بانني قد التزمت بقوانين جامعة آل البيت وأنظمتها وتعليماتها السارية المفعول المتعلقة بإعداد رسائل الماجستير والدكتوراه عندما قمت شخصيا بأعداد رسالتي بعنـــــوان:

Enhancing the performance of Intrusion Detection System by minimizing the false alarm detection using fuzzy logic

وذلك بما ينسجم مع الأمانة العلمية المتعارف عليها في كتابة الرسائل والأطاريح العلمية. كما انني أعلن بان رسالتي هذه غير منقولة أو مستلة من رسائل أو أطاريح أو كتب أو ابحاث أو أي منشورات علمية تم نشرها أو تخزينها في أي وسيلة اعلامية، وتأسيساً على ما تقدم فانني أتحمل المسؤولية بأنواعها كافة فيما لو تبين غير ذلك بما فيه حق مجلس العمداء في جامعة آل البيت بإلغاء قرار منحي الدرجة العلمية التي حصلت عليها وسحب شهادة التخرج مني بعد صدورها دون أن يكون لي أي حق في التظلم أو الاعتراض أو الطعن بأي صورة كانت في القرار الصادر عن مجلس العمداء بهذا الصدد.

التوقيع:

التاريخ:

Enhancing the performance of Intrusion Detection System by minimizing the false alarm detection using fuzzy logic

By:

Mohammed O. Alshroqee

Supervisor

Dr. KhaledBatiha

This Thesis was Submitted in Partial Fulfillment of the

Requirements for the Master's Degree of Science in

Computer Science

Deanship of Graduate Studies

Al Al-Bayt University

2015

Dedication

This thesis is dedicated to my father and mother, my dear lovely wife (Alaa), my children (Omar and Masaa), my sisters, and my friends for their endless love, support and encouragement.

Acknowledgment

I would like to record my thanks to Prof. Khaled Batiha for his constant and honest supervision,assistance, and beneficial remarks which have made this study conceivable. I like also tothank the Department of Computer Science at Al al-Bayt University forthe valuable support and encouragement during my study for the MA degree.

My amiable thanks and appreciation go to the defense committee for their valuable comments after reading this study; namely, Prof. Khaled Batiha, Dr. Saad Bani-Mohammad, Dr. Jehad Alnihoud, and Dr. Belal Abo Ata.

Contents

1.	**Introduction**	**1**
	1.1 Motivation	4
	1.2 ProblemStatement	4
	1.3 Contribution	5
	1.4 Thesis Structure	5
2.	**Related Work**	**6**
3.	**Fuzzy logic and SSdeep Algorithm**	**11**
	3.1 Fuzzy Logic	11
	3.2 SSdeep Algorithm	11
	3.2.1 Piecewise Hashing	12
	3.2.2 The Rolling Hash	13
	3.2.3 Combining The Hash Algorithms	13
	3.3 Approximate Matching	14
	3.4 Custom Score	14
4.	**Methodology**	**16**
	4.1 Preprocessing Stage	22
	4.2 Processing Stage	23
5.	**Conclusion and Future works**	**32**
	References	**33**
	ملخص البحث	**37**

List of Figures

Figure 3.1	Sample piecewise MD5 hashes	13
Figure 4.1	Minimum percent of similarity for each folder	17
Figure 4.2	Minimum percent of similarity depend on file type	18
Figure 4.3	Preprocessingstage	22
Figure 4.4	processing stage	24
Figure 4.5	Number of alarms when number of file = 25	28
Figure 4.6	Number of alarms when number of file = 100	28
Figure 4.7	Number of alarms when number of file = 200	29
Figure 4.8	Number of alarms when number of file = 300	29
Figure 4.9	Number of alarms when number of file = 500	30

List of Tables

Table 4.1	Hash value in SHA1	20
Table 4.2	Hash value in MD5	21
Table 4.3	Hash value in SSdeep	21
Table 4.4	Folder priority	26
Table 4.5	Total alarms according to the number of changed files	27
Table 4.6	Results for each folder	31

List of Abbreviations

IDS	Intrusion Detection System
NIDS	Network Intrusion Detection Systems
HIDS	Host Intrusion Detection System
CTPH	Context Triggered Piecewise Hashes
SHA-1	Secure Hash Algorithm-1
MD5	Message Digest algorithm 5
SD Hash	Similarity Digest Hashing
GA	Genetic Algorithm

Abstract:

According to the information technology and regarding to the revolutions of the computer worlds, this world has got important information and files that have to be secured from different types of attacks that corrupt and distort them. Thus, many algorithms have turned up to increase the level of security and to detect all types of such attacks. Furthermore, many algorithms such as Message Digest algorithm 5 (MD5) and Secure Hash Algorithm 1 (SHA-1) tend to detect whether the file is attacked, corrupt and distorted or not. In addition, there should be more algorithms to detect the range of harm which the files are exposed to in order to make sure we can use these files after they have been affected by such attacks. To be clear, MD5 and SHA-1 consider the file corrupt once it is attacked; regardless the rate of change .Therefore, the aim of this thesisis to use an algorithm that allows certain rate of change according to the user which is SSdeep algorithm. Meanwhile, it gives the rates of change depending on the importance of each file. Moreover, each rate of change determines whether we can make use of the file or not. I made assumption in creating four folders, each contains multiple files with minimum predefined allowed of similarity. Then I created graphical user interfaceto utilize the SSdeep algorithm and to permit user to define the allowed similarity on each folder or file depending on impotency of it. After applying the algorithm, Igotresults showing the benefits of such algorithm to makeuse of these attacked or modified files.

Chapter One

1. Introduction

As computers are becoming increasingly used bylabor, security issues have posed a big problem within organizations. Firewalls, anti-virus software, password control are amongst the common steps that people take towards protecting their systems. However, these preventive measures are not perfect. Firewalls are vulnerable; they may be improperly configured or may not be able to prevent new types of attacks. Anti-virus software works only if the virus matches its signature. Passwords can be stolen and therefore, systems can be easily hacked into. Hackers can change the system on initial access and manipulate it so that their future access will not be detected. In these situations, Intrusion Detection Systems (IDS) come into play [Mallery, 2008].

Intrusion Detection System (IDS):

It is a software, device, or combination of them that monitors a network or system for suspicious activities, and informs the administrator by alerts that someone has done anomaly behavior in the network or system. Therefore, the IDS is to detect all these violations very quickly [Anderson, 1995]. Furthermore, the IDS is meant to determine whether the file or the packet isintruded or not. Depending upon the place of anomaly, the IDS can be classified into Network-based IDS or Host-based IDS [Scarfone, 2007].

So the IDS is classified into the following:

1. Network-based IDS (NIDS): Network Intrusion Detection Systems are systems placed at strategic points in the network in order to examine traffic

among all devices in the network and detect all types of activities. They tend to compare the passing packets with known attacksto identify any problem. Then an alert is sent to the administrator or user when a suspicious behavior is detected [Steven, 1991].

2. Host-based IDS (HIDS): Host Intrusion Detection System works on either individual hosts or available devices on the network. HIDS detects and monitors packets, whether inbound or outbound within the device so it runs alerts to the user if any suspicious behavior is detected. The way it works is that it takes a snapshot for the existing file, compared to the original one. If any of the original files of the system is modified or deleted, an alert is raised and sent to the user for the purpose of investigating [Teresa, 1988].

The present study focused on HIDS. Therefore, whether the IDS is HIDS or NIDS, it should inform the administrator by any means that suspicious activities had happened and should trigger an alarm.

IDS alarm is the status or the situation that IDS detects suspicious activities or intrusions that occur to the system, warn the administrator about it as an alert, and these alerts can be classified as the following:

1. True Positive (Attack – Alert): A reasonable attack which urgesan IDS to produce an alarm.
2. False Positive (No attack – Alert): An event signaling an IDS to produce an alarm when no attack has happened.
3. True Negative (No Attack–No Alert): When no alarm is raised or produced, no attack has taken place.

4. False Negative (Attack – No Alert): When no alarm is raised when an attack has taken place [Nitin, 2008].

False Alarm Detection:

The so-called "False Alarm Detection" is a method that specializes in detecting anomalies in computer systems and networks, which is mainly dependent on fuzzy logic and artificial intelligence. The main purpose is to differentiate between normal behaviors and anomalous ones. What makes gaps and weaknesses is the false alarm rate mainly measured and counted by the false positives of normal behaviors [Pokrywka, 2008].

To clarify the idea, some anti-virus programs deal with programs and data as viruses which are directly stopped. This, in role, gives false alarms. For example, the so-called "Kaspersky" anti-virus program deals with the program "Net Support" as a virus which is directly stopped and cannot be installed. Because they are not viruses, we conclude that what happens is the so-called "False Alarm".

Fuzzy Logic:

It is an extension of Boolean logic that is used for computer-based complex decision making. On the one hand, in the old classical Boolean set, an element could be either a full member that indicates the (1) value, whereas the non-member indicates the (0) value. On the other hand, the membership of values in a fuzzy set deals with values within the interval (0, 1), thus it allows partial membership of elements in a set.

Highlighting both fuzzy logic and IDS,theIDS deals with two values (1, 0). Thus, this is considered a gap or weakness that can corrupt the process of detection. That is why the Fuzzy Logic has come to stand by the side of IDS in order to vary the values within

the interval of (0, 1). This is, in role, reinforces the level of detection and security because depending of two certain values may cause problems in a computer network [Zadeh, 1965].

1.1 Motivation

Many researchers have done researches in the field of computer and its network, and they have talked and discussed the threats that threaten them without talking about what is supposed to do in order to recognize the consequences and the damage by attacks and hackers. Therefore, this research is dedicated to measurethese attacks' damages. Regarding this matter, the Intrusion Detection System is used to detect these threats in systems. Therefore, in using the software, we developed the techniques we have created.Thus, the user canrecognizethe consequences and the damages and the attacks thathappened.Consequently, this research shows in details the problems, types of attacks, used algorithms and approaches to detect these illegal threats and its impact.

1.2 Problem Statement

When changes happens to some critical files, the IDS alerts thesystem or the administrator that these files are intruded but without specifying thesize of the damage and the effect of damage which may increase the false alarm. As a result, we need analgorithm that can investigate the size of the damage and its effects. In so doing, the damage can not be considered harmful and does not need to fire an alarm about it.

Consequently, the present study gives the user the control with the range of trueness of produced alarms that detecting intrusions.

1.3 Contribution

This research comes up with findings and solutions that are supposed to be followed when our computers and networks are exposed to face malicious attacks and hackers who tend to distort our main and important files. Besides, it showshow we can utilize the proposed algorithms to detect the anomalies in these files and take re-use of some of the attacked files.

1.4 Thesis Structure

This thesis is organized as the following:

Chapter 2 includes related works of previous studies.

Chapter 3 explains fuzzy logic and SSdeep algorithm and how the work correctly.

Chapter 4 clearly shows how we applied the methodology of SSdeep algorithm and what happens during this process.

Chapter 5concludes the main ideas of the present study and develops future achievements.

Chapter Two

Related Work

This chapter aims to present a brief description of some previous approaches related to SSdeep algorithm, Intrusion Detection System and fuzzy hash function.

Zadeh [Zadeh, 1965] started with the concept of fuzzy set theory which was meant to focus on the vagueness for dealing with it in several cases in the world. The function called membership explains the values of universe that lie between (0, 1). Each value has got an indication. The (0) value indicates that it is not a member in the fuzzy set, whereas the (1) value indicates that it is a member in the fuzzy set. So, the other values remainwithin this range.

A.Menezes et al [A.Menezes, 1997] According to this study, hash functions have got two essential functions ease of computation and compression. Compression means that the length of the input file is not paid attention to, whereas the output one has a limited and fixed one. Because of that, the fuzzy hashing is considered puzzling and similarity digest is more convenient. So, they use similarity digest, fuzzy hash function and similarity preserving hash function as synonyms.

Harbour[Harbour, 2002] pointed to similarity preserving hashing was the so-called block based hashing. The process is very easy: divide a random input in blocks of fixed

size, hash each block distinctly and concatenate all hash values. To be in control with this approach, it is adequate enough to add/remove one byte at the beginning. Thus the entire input changes and all hash values will differ in size.

Kornblum [Kornblum, 2006] found Context Triggered Piecewise Hashing (abbreviated CTPH) that divides an input dependent on its context. It was basically dependent on a spam detection algorithm of Tridgell. Since then several researches had been published which checked this method in details. For example, improvements related to efficiency and security had been proposed by F.Breitinger in "Performance Issues about Context-Triggered Piecewise Hashing", whereas a security analysis had shown that this method cannot resist an active opponent regarding whitelisting and blacklisting.

MahbodTavallaeeet al [Mahbod, 2009] Anomaly detection has caught many researchers' eyes to dominate the weakness and faults of signature based IDSs in catching unusual attacks and behaviors. KDDCUP'99 is the most widespread and used dataset for the progression of these systems. Because he had performed a statistical study on this data set, he found two essential issuesthat,to a certain extent, influenced the performance of considered system and the results in a very lacking evaluation of Anomaly Detection approaches. In addition to what is mentioned above, Anomaly Detection has grown up with many advantages and features that increase the level and quality of security. Moreover, it increases security because its main interest is to stand against all these obtrusive attacks and behaviors.

Roussev [Roussev, 2010] proposed an improved method called SDhash dependent on some previous studies. SDhash (similarity digest hashing) takes out "statistically improbable features" using an entropy calculation for each 64 byte sequence, i.e., bytes 0 to 63, 1 to 64, 2 to 65, ... when a 'characteristic feature' is recognized. Therefore, it is hashed using the cryptographic hash function SHA-1 and inserted into a Bloom filter. So, files are similar if they have common and similar features.

VassilRoussev [VassilRoussev, 2011] proposes a comparison between SDhash and SSdeep which demonstrates that an "approach significantly outperforms in terms of recall and accuracy in all tested scenarios and insists on active and scalable behavior".

Mostaque [Mostaque, 2013] has discovered approaches and a better fuzzy classifier using Genetic Algorithm. In addition, he proposed encounters in IDS. He recommended the firsthand definition of fuzzy set where he defined the fuzzy membership value and fuzzy membership function. These are two different functions for the reason that the external value is not always totaled gradually, from the first level. The most benefit of this method helps in decreasing the false alarm rate in Intrusion Detection.

B. Uppalaishet al [B. Uppalaish, 2012] proposed the GA and applied it to KDDCUP99 dataset to new rules for IDS in order to clarify and categorize the types of attacks and

irregular behaviors. This is to increase a high-level security so as to detect and catch these obtrusive violations.

J. Gomez & D. Dasgupta [J. Gomez & D. Dasgupta, 2002] the authors presented the method of fuzzy logic to decrease the false alarm rate to detect obtrusive behaviors and attacks. This set stands for detecting the usual and unusual behavior in networks of computers. The authors presented a method to use and apply fuzzy rules that are to detect intrusive activities and particular intrusions. This approach presented an innovative approach in order to catch these attacks which decrease the level of security and fill it with many gaps and weak points. Besides, it is to classify these types to build a firewall to protect and secure our networks.

Anderson,[Anderson, 1980] defines the IDS as an expert system that aims at detecting false alarms caused by programs that tend to masquerade as another program. He also conducts experiments using the statistical rules to detect the changes and the abnormal behaviors.

Gassata, [Gassata, 1998] follows an approach that tends to find the types of attacks in the audit data so genetic algorithm is used to perform this process efficiently. During this process, it has many activities that vary, which allow the attacks to prevail and start attacking.

Jose Nazario, [Jose Nazario, 2004] the author in this research describes the Intrusion Detection as a perfect method to detect the activities of worms. In addition, the number of alerts increases when the connection is lost. These worms attack networks and quickly spread unless this method does not detect them as quick as possible. So, this method is meant to do this process in enough time to strike security and safety.

Despite the importance of the above list of sources, this study presents a different perspective. As mentioned earlier, the previous studies highlight and detect the damage on the network only. However, this study detects the damage on the host. In fact, it tends to propose an algorithm that measures the size of the damage and the effects of the damage when changes influence some critical files. Moreover, it comes up with solutions to re-use some of the attacked files.

Chapter Three

Fuzzy Logic and SSdeep

3.1 Fuzzy Logic

Fuzzy logic is a method of computing dependent on degrees of truth, not the normal "false or true" (0 or 1) which is the so-called "Boolean logic" on which the new computer and its network are based. The idea of fuzzy logic was firstly found by Dr. Lotfi Zadeh in 1960s. Dr. Zadeh was working on the gaps of computer understanding of a natural language. Natural language is not easily translated into the absolute values of 1 and 0 [Zadeh, 1965].

Fuzzy logic consists of 1 and 0 as logical cases of truth but also contains the various cases of truth in between so the result of a comparison between two things could be not "short" or "tall" because we could have values that lie between the two logical values [Rouse, 2006].

3.2 SSdeep Algorithm

It is an algorithm that descends from context triggered piecewise hashing to recognize certain files that have got data inserted, changed or deleted.

Firstly, it does a test for the cryptographic hashes used by forensic users and what gaps exist with these hashes. Secondly, a process called piecewise hashing starts activating. At last, the rolling hash algorithm produces a pseudo-random output that basically depends on the original context of an input. The rolling hash is mainly used to set the limits of the original piecewise hash. These hashes could be used to recognize the

similar sequences between the unknown inputs and current files even if the unknown file is a modified copy of the current file. So, all these sequences lead to the concept of SSdeep algorithm [Kornblum, 2006].

SSdeep algorithm consists of three stages:

1. Breaking up the file into pieces using the result of a rolling hash functions.

2. Using another hash function to a produce a (small) hash for each piece.

3. Concatenating the results to produce the hash signature for the whole file.

SSdeep is classified into the following processes:

1. Piecewise Hashing.

2. The rolling hash.

3. Combining the hash algorithm.

3.2.1 Piecewise Hashing

It was developed by Nicholas Harbour in 2002. He focuses on piecewise hashing that are used as haphazard hashing algorithm in order to create checksums for the whole file, instead of one. In addition, it creates discrete fixed-size segments for the file, not just one. This process generates, for example, the first 512 bytes of the input file while the other hash is for the next 512 bytes and the same process repeats. See fig. 3.1 for a set of sample piecewise hashing. What is special about it is that it was developed to reduce the

gaps during the forensic imaging. What happens here is that once we have an error or a gap, only one of these hashes is corrupt [Kornblum, 2006].

```
0 - 512: 24a56dad0a536ed2efa6ac39b3d30a0f
512 - 1024: 0bf33972c4ea2ffd92fd38be14743b85
1024 - 1536: dbbf2ac2760af62bd3df3384e04e8e07
```

Figure 3.1: Sample piecewise MD5 hashes

3.2.2 The Rolling Hash

This algorithm, relying on the current context of the input file,differentiates a pseudo-random value. In so doing, the created and the rolling hash preserve a state that relies on the last few bytes of the input file. Consequently, each byte is added to the state after it is replaced and removed from the current state for the other byte to process [Kornblum, 2006].

3.2.3 Combining the Hash Algorithms

The rolling hashis used when the current piecewise hashing programs used fixed offsets to conclude when to begin and to stop the traditional hash algorithm. Yet, when the rolling hash's output is specific, the traditional hash is triggered. While processing the input file, one must compute not only, the traditional hash for the file but also, the rolling hash in order to record the value of the traditional hash in the CTPH signature, paving the way for the traditional hash to rest. As a result, any change in the input is recorded and seen in localized changes only in the CTPH signature, maintaining the majority of the CTPH signature. Therefore, the modified file is associated with CTPH signature of known files.

3.3 Approximate matching

Using the SHA-1 or MD5 only gives two simple answers (yes or no), so two matched files might match or might not.However, according to SSdeepalgorithm, itgives a probable answer within the interval of numbers (0-1) once two files are compared. Here we have got two types of scores as the following:

1. Confidence score that indicates a low score when there is a small amount of similar content in the two files.
2. High score when the ratio of similarity of content is high [Roussev, 2013].

3.4 Custom Score

We proposed another type of score called custom score. It is mainly dependent on the importance of certain files according to the user point of view. This allows the user the ability to choose the files he wants to be secured and how much he allows a percentage of changes.

To be clear on this matter, we suggested that the user has important files which kept in a folder with minimum 95 percent of importance, and the less important files are kept in a folder that has got minimum 85 percent of importance. The files which have got minimum 75 percent of importance are also saved in a third folder, and the less important files are saved in a folder that has got minimum 65 percent of importance. Moreover, the custom score comes in the interval (0, 1) because (0) means that the file is totally different, whereas (1) indicates that the file is totally identical. What comes in between is dependent on the user's custom score.

Empirical tests stated that more than 65% of similarity leads to the recognition of the same files as they seem to be identical [Chyssler, 2004].

At last, we conclude that the files are classified into many percentages of importance according to the user's desire. This also shows if the input file can be used or distorted.

Chapter Four

Methodology

The attacks have differentimpactson the files, some of these attacks are considered innocuous depending on the data types. Also, we have set an assumption stating that determining the allowed rate of change belongs to the user's request of choosing the type of file and the location of it, and the files are compared by File Finger Print (Hash). Some algorithms such as SHA-1 or MD5 consider that any attack on the data affects it and not useful. But it cannot determine the size of the damage on these data, whereas the SSdeep algorithm can determine the size of damage on these data. Depending on the above assumption, some data are very important. Thus, if the attack happened over the allowed rate, they are considered useless or unbeneficial. On the other hand, if some of the files are attacked within the allowed rate, they can be used and considered safe. Therefore, the figure (4.1) demonstrates our assumption, depending on the location and the extension of file. In so doing, we can determine that the alarm, produced by the attack, is true or not.

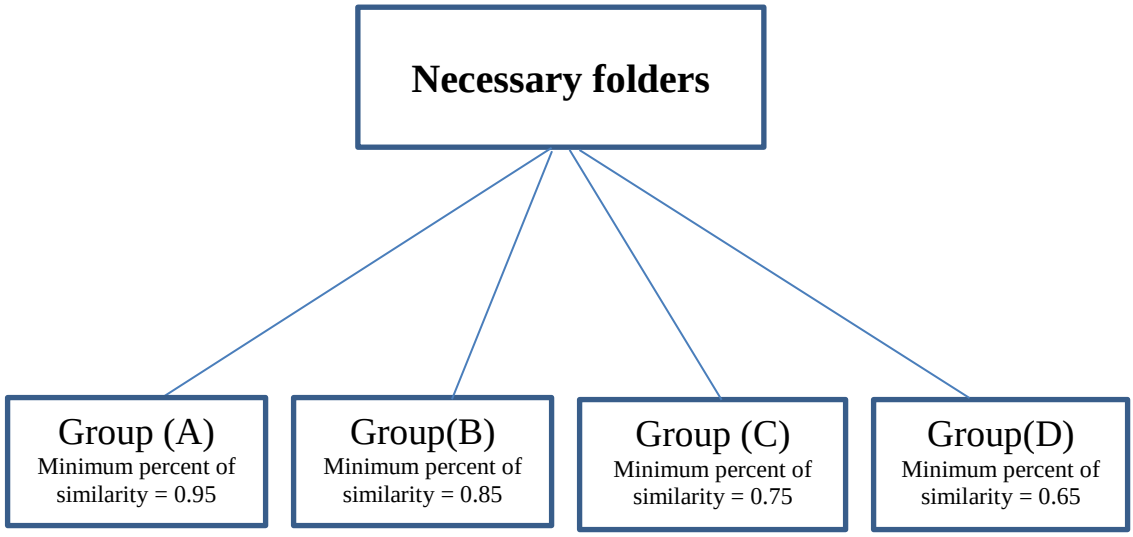


Figure 4.1: Minimum percent of similarity for each folder

The folder that is called “Necessary folders” is classified in to four main folders according to minimum percent of allowed similarity, and the following folders show this classification:

Folder "A":

This folder is the most important one that includes sensitive files.

Folder "B":

This folder is less important than folder "A", which includes files which can be exposed to few changes.

Folder "C":

Folder "C" includes files which can face more changes than folder "B".

Folder "D":

This one includes files that can face a lot of changes in comparison with the previous folders.

In addition to this, we allowed the user to set the importance of minimum of similarity according to the type of the file (extension) in order to increase the security. Here, the proposed assumption of the minimum of similarity depends on the user himself and his job. For example, the most important files for secretary are Ms-word files. Therefore, the minimum of allowed similarity on these files is very high and these files are critical, while the pdf files are less important with less minimum of similarity. However, according to a programmer, the most important files are the DLL files and the minimum allowed of similarity isvery high, while the Ms-Excel files are less important withless minimum allowed of similarity.

As we can see in Figure (4.2), we proposed generally the minimum allowed of similarity of some files, but the user can set his own minimum allowed percent of similarity depending on his job and the location he puts the files into. Also, we made assumption states that the default percentage of any other files is 65 percent.

Minimum percent of similarity is 0.67.

Minimum percent of similarity is 0.75.

Minimum percent of similarity is 0.80.

Minimum percent of similarity is 0.90.

Minimum percent of similarityis 0.95

Figure 4.2: Minimum percent of similarity depend on file type

Consequently, the comparison is made according to the type of the file (extension) and where it exists, based on the highest rate, as demonstrated in the following equation:

α: file acceptable percentage (extension for file).

β: folder acceptable percentage.

μ: result percentage.

$$\mu = \begin{cases} \alpha \geq \beta & \alpha \\ \text{otherwise} & \beta \end{cases}$$

μ: is it the highest percentage

Comparison between SHA-1, MD5 and SSdeep algorithms:

The following example, depending on applying these algorithms on a paragraph on page (20), clarifies the difference among the three algorithms. In the first two algorithms, SHA-1 and MD-5, it is clear that the new hash is different from the original one when any change happens to the text as shown in table 4.1 and 4.2 respectively. Actually, we do a process of calculation for the text, and we notice that the target text is quite different from the original one.

If we remove the first character from the beginning, five characters from the middle, and the last character from the end, we find out that the hash of the original text differs quietly from the original hash of the original text as shown in the previous mentioned tables.

In comparison with these two algorithms, we come to describe the SSdeep algorithm that depends on the file size. Regarding this algorithm, we do the same process of calculation for the text in order to notice what happens after that. Here we remove the

same characters from the three positions and compare the original text hash to the new text one. Even though the text is changed, this file is a bit changed. In addition, it shows that the target hash still exists and there is a clear percentage of hash similarity as shown in table 4.3.

According to the previous results, the first two algorithms negate the existence of the file because they are not dependent on the original hash, whereas the SSdeep algorithm confirms the existence of the file because it allows a certain percentage of change to the file.

"Tobacco smoke is enormously harmful to your health. There's no safe way to smoke. Replacing your cigarette with a cigar, pipe, or hookah won't help you avoid the health risks associated with tobacco products. Cigarettes contain about 600 ingredients. When they burn, they generate more than 7,000 chemicals, according to the American Lung Association. Many of those chemicals are poisonous and at least 69 of them can cause cancer. Many of the same ingredients are found in cigars and in tobacco used in pipes and hookahs. According to the National Cancer Institute, cigars have a higher level of carcinogens, toxins, and tar than cigarettes."

Table 4.1: Hash value in SHA1

	Hash in SHA1
Original text	E3D4F4EC354815E376A2EF32025B2985EA2926D5
First characters removed.	1A013F779731FC0875F37C746BB9DFA21A1DD6C7
From middle paragraph 5 characters are removed.	078C55BE5B2D7A62F918737AE0CCBC9D0F19EE39
Last character is removed.	92339FC32E9BB409C801AD980E30A85F4013ADFC

Table 4.2: Hash value in MD5

	Hash in MD5
Original text	904F57483F6B1B194262ACE22E35B881
First characters removed.	EACAEA3BD5D1CD61920752BCEA816B60
From middle paragraph 5 characters are removed.	B80D307C48650D90BCB895D393BE7DD4
Last character is removed.	05517FEB46980300421123BB08C5578C

Table 4.3: Hash in SSdeep

	Hash in SSdeep
Original text	KVuReUuIuGA+LF0MQwy9IM+N5VexyYweffWguRmH+4KIAXiooF qqFlL4n
First characters removed.	AVuReUuIuGA+LF0MQwy9IM+N5VexyYweffWguRmH+4KIAXiooF qqFlL4n
From middle paragraph 5 characters are removed.	KVuReUuIuGA+LF0MQwy9IM+N5VexyYHvlfosDKlNSvnipm4rrs7Io XvKM
Last characters removed.	KVuReUuIuGA+LF0MQwy9IM+N5VexyYweffWguRmH+4KIAXiooF qqFlL4u

Stages of our proposed work:

4.1 Preprocessing Stage:

Our system deals with a huge set of files that cannot process them shortly so we do pre-processing for these files and calculate the hash for each one. Then, we store them in the data base to be used later.

In this stage, the system scans all the selected files by the user and passes them to "SSdeep algorithm" in order to produce a hash code which consists of alphabets and symbols for each file. After that, it stores their hash codes, sizes, and paths in the data base as shown in the figure 4.3.

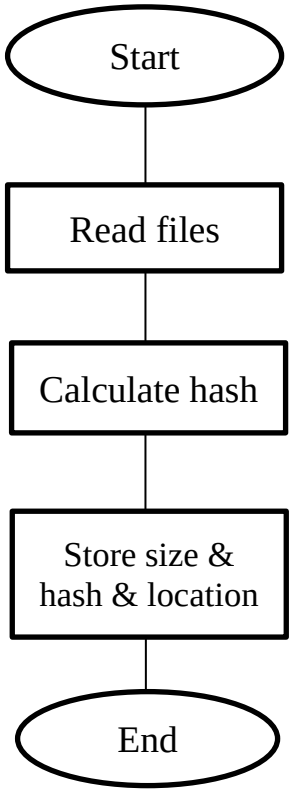


Figure 4.3: Preprocessing stage

4.2 Processing stage:

In this stage, we tend to divide this process into the following steps:

I. Input file:

Here we choose the file we need to check whether it is attacked or not.

II. Calculating the file hash (using "SSdeep algorithm"):

The system passes the file to SSdeep algorithm to produce a hash code that consists of alphabetical characters and symbols.

III. Hash existence check:

The system searches in the database for the hash code and compare it to the input file hash code. If the hash code of input file is similar to the current file hash code, we make sure that the file is not attacked because there is, at least, one file with the same size and hash, so no alarm is on this file. Otherwise, it calculates the file size and stores it in (T). Then (T) equals " file size *0.35". Afterwards, it creates a file probability range that tends to add T to the input file size and subtract T from the input file size. In addition, it does a small procedure that creates a range of files (zero or more files), based on the target file size. After that, it starts searching in the result list (zero or more files). If it doesn't find any file within the specified files size range (+T,-T), the system will send an alarm to the user. But if it finds files, it calculates the percentage of similarity depending on extension and location.

If it has, at least, one file in thedatabasewhich has an acceptable percentage of similarity; it will alert the user that the file has been changed without any alarm. Otherwise, it will send an alarm to the user because the file has been changed over the predefined allowed percentage of similarity as shown in figure 4.4.

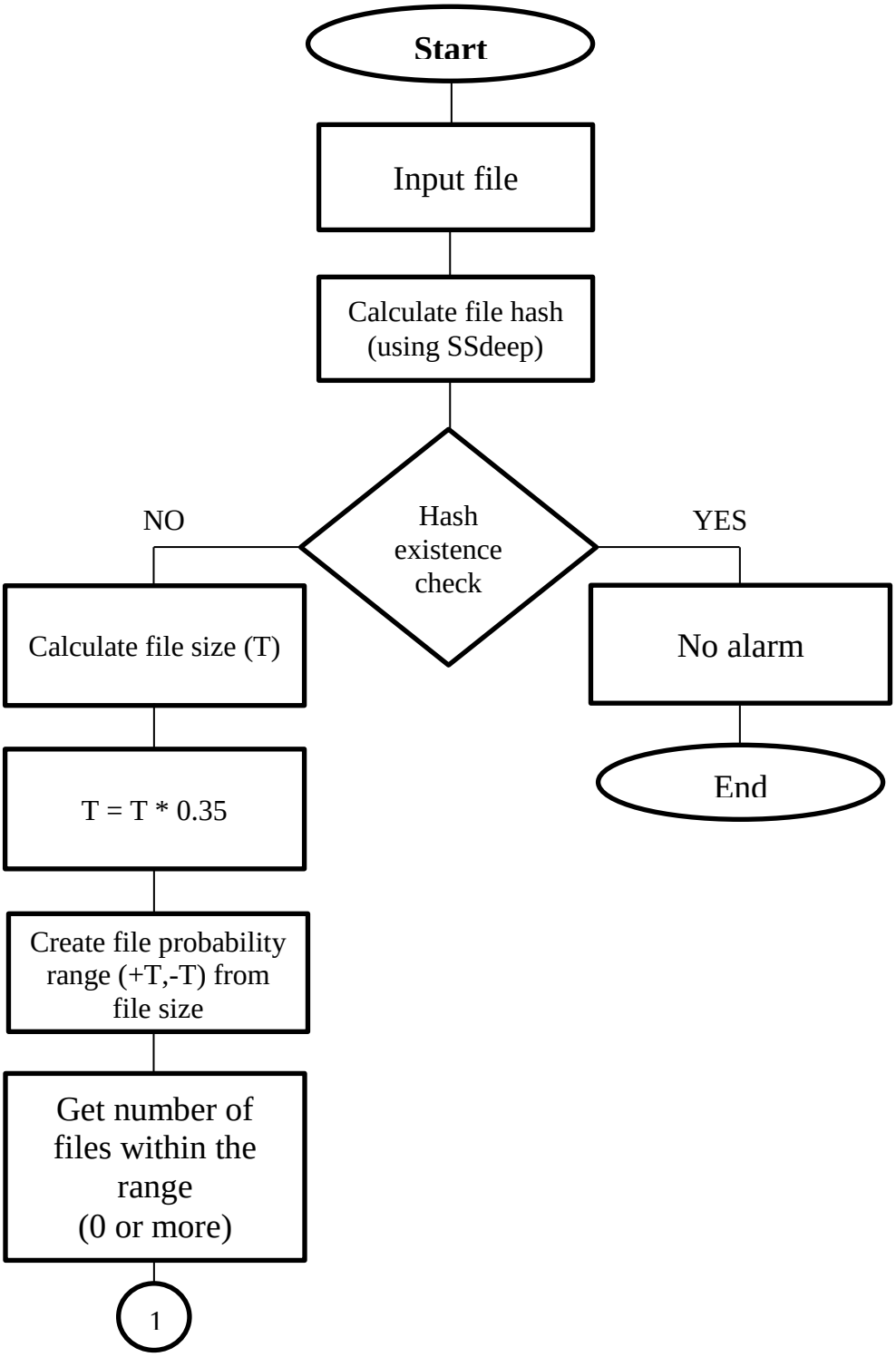


Figure 4.4: Processing stage

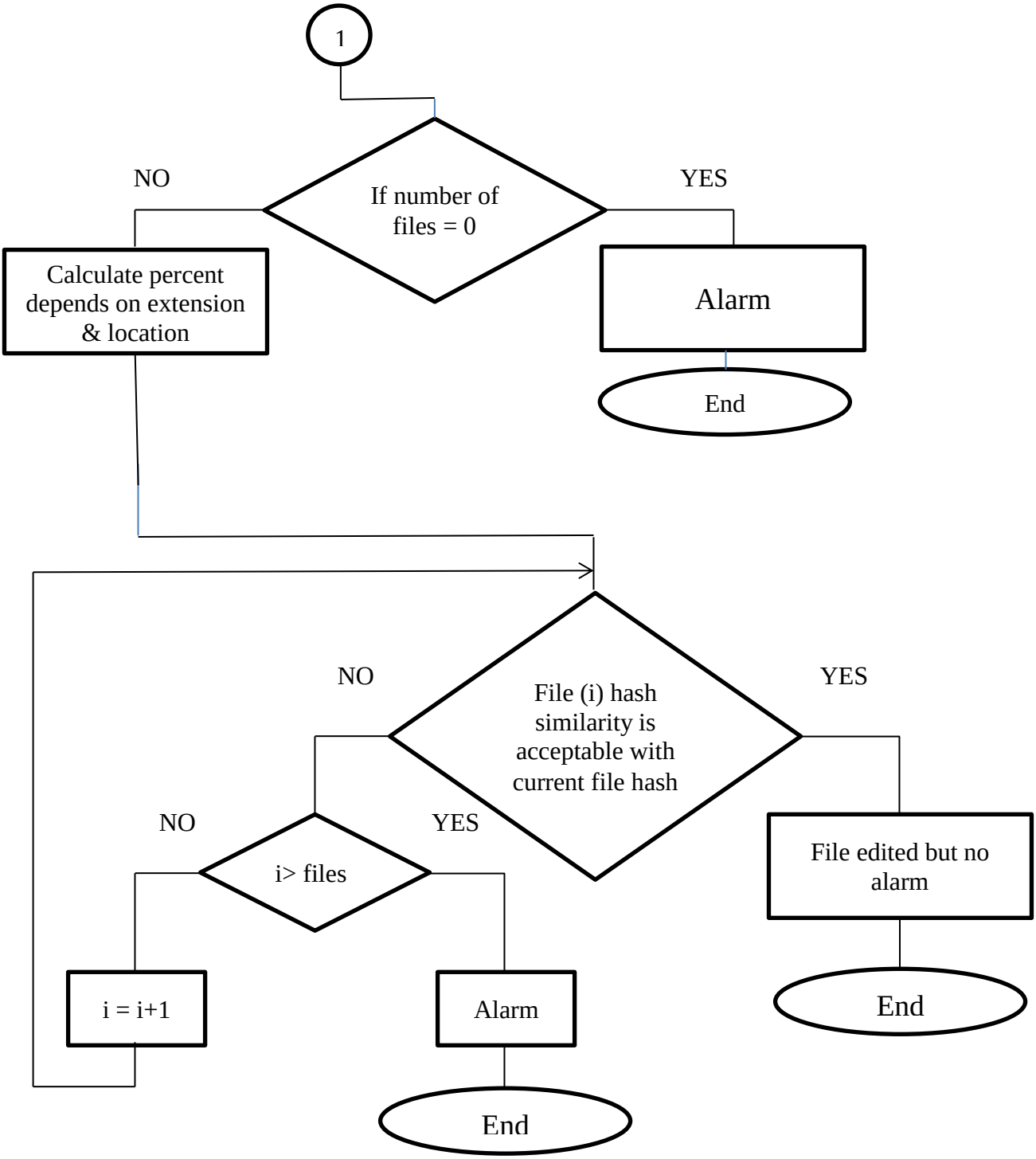


Figure 4.4: Processing stage (Cont.)

Results:

Assuming we have four folders each has minimum percent of similarity as shown in table 4.4.

Table 4.4: Folder priority

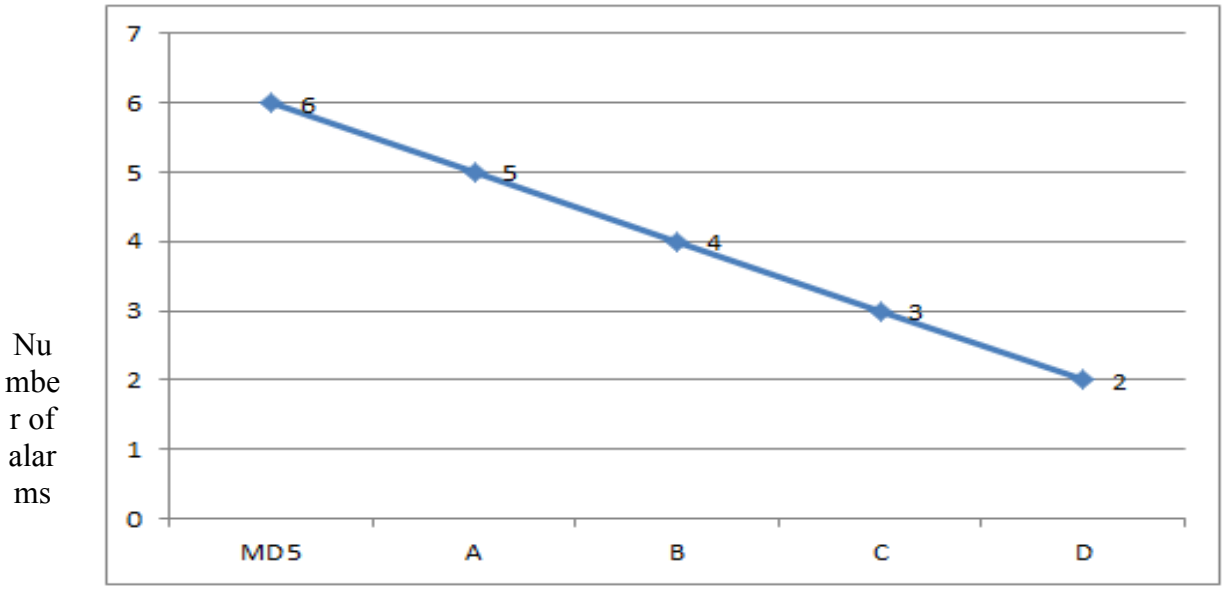
Folder name	Score
A (Critical)	>= 0.95
B (Very high)	>= 0.85 and <0.95
C (High)	>= 0.75 and <0.85
D (Low)	>= 0.65 and <0.75

Here five experiments are made, in experiment one we took twenty five files and we let the system go through the preprocessing stage to store the size and the hashes, then we change six files which have been intruded, then we checked the whole files with MD5 algorithm to check whether these files have been hacked or not, and how many files have been hacked. As a result, six files have been hacked. Then when we move the files to folder (A) and check them with SSdeep algorithm, the system finds five files have been hacked. After that, we move them to folder (B) and check them again with SSdeep algorithm. As a result, four files have been hacked. Then we repeat that with folders (C and D) and the results are similar to the ones shown in table 4.5 and figure 4.5. In the second experiment, we increased the number of files to one hundred, and change fifteen files. Then we checked them with MD5 algorithm and then with SSdeep algorithm while being in folder (A) then folder (B) until folder (D) and the results are similar to the ones as shown in table 4.5 and figure 4.6. Then, we repeat the whole procedures with the third

experiment till five, taking into consideration, changing the total number of files and the number of the changed files as shown in table 4.5 and figures (4.7, 4.8 and 4.9).

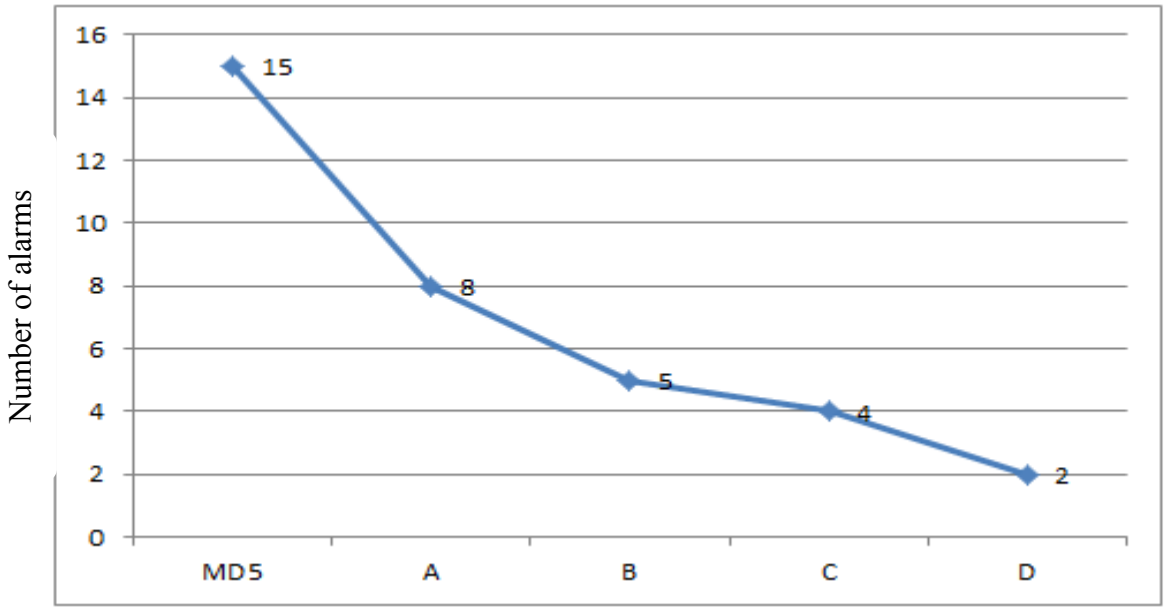
Table 4.5: Total alarms according to the number of changed files

Experiment number	Total files	Number of changed files	Total alarms of MD5 in all folders	Total assumed alarms in folder "A"	Total assumed alarms in folder "B"	Total assumed alarms in folder "C"	Total assumed alarms in folder "D"
1	25	6	6	5	4	3	2
2	100	15	15	8	5	4	2
3	200	25	25	12	10	7	4
4	300	40	40	20	17	16	13
5	500	60	60	31	28	20	16



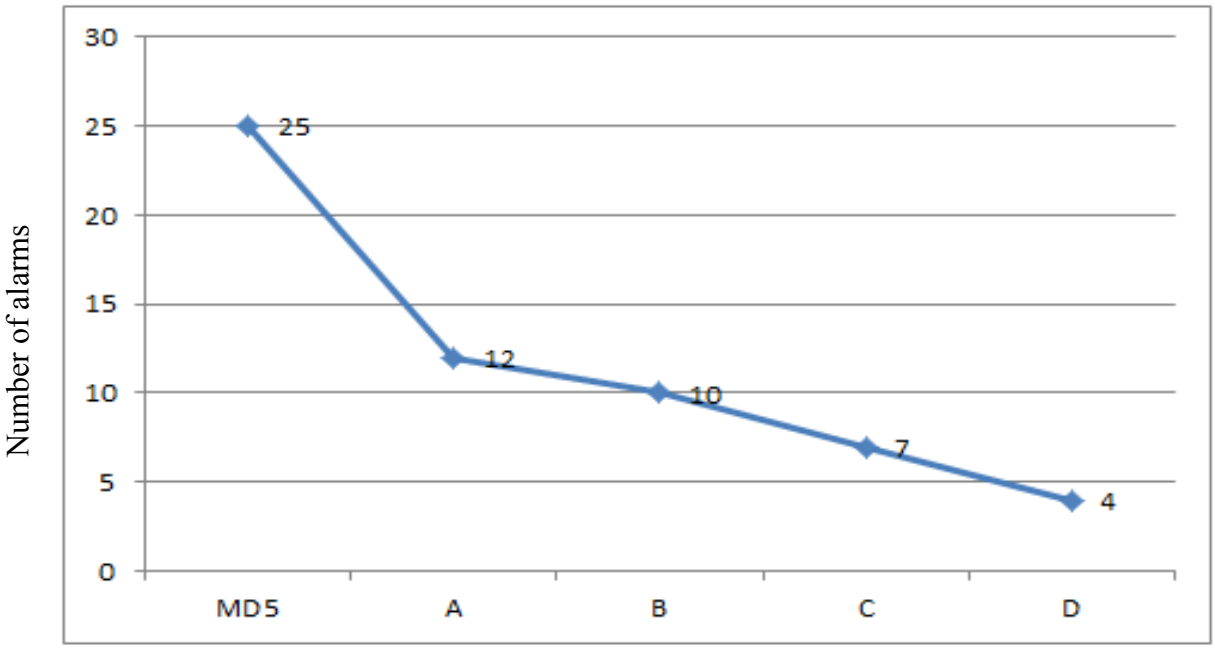
Number of file = 25

Figure 4.5: Number of alarms when number of file = 25



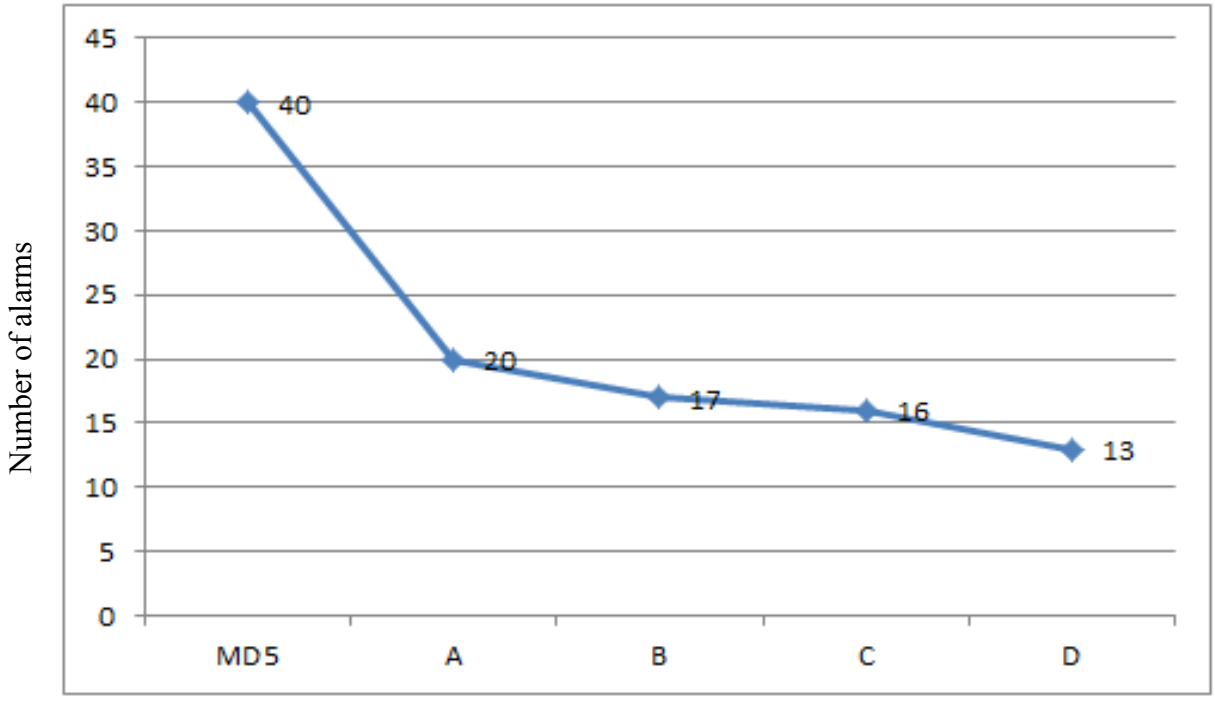
Number of file = 100

Figure 4.6: Number of alarms when number of file = 100



Number of file = 200

Figure 4.7: Number of alarms when number of file = 200



Number of file = 300

Figure 4.8: Number of alarms when number of file = 300

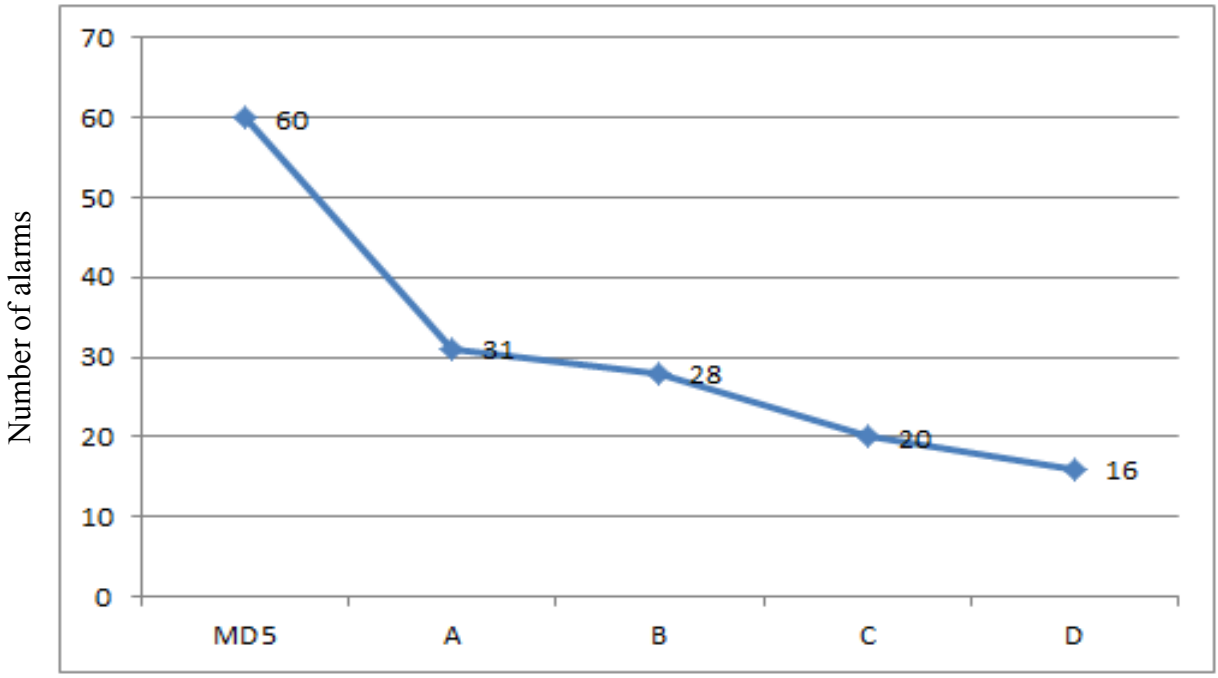


Figure 4.9: Number of alarms when number of file = 500

Also, we can see in table 4.6 all the accepted files in folder (A) will be also accepted in the other folders (B,C and D). Accordingly, all the files accepted in folder (B) will be accepted in folders (C and D) but they will not be accepted in folder (A). While the files accepted in folder (C) will be accepted in folder (D), but they will not be accepted in folders (A and B). Finally, all the files accepted in folder (D), will not be accepted in Folders (B, C and D), Depending on the folder minimum of similarity.

Table 4.6: results for each folder

Location / μ	A	B	C	D
Critical	Acceptance	Acceptance	Acceptance	Acceptance
Very high	Avoidance	Acceptance	Acceptance	Acceptance
High	Avoidance	Avoidance	Acceptance	Acceptance
Low	Avoidance	Avoidance	Avoidance	Acceptance

Chapter five

Conclusion and future work:

We can notice from the results, in the previous chapter, that as we increased the number of total files and the number of changed files, the MD5 algorithm will always give the same number of changed files, whereas the detected number of changed files will decrease, depending on minimum allowed percentage of similarity. As we can see I developed a simple graphical user interface for a program utilizing the SSdeep algorithm to allow a user manually to check if his files have been attacked or not with allowable setting for the minimum allowed percent of similarity for the files types or its locations. As a result the user can re-use some of the attacked files depends on the minimum of similarity.

In using such hashing, the examiners will be able to associate the previous lost files. In this way, the examiners, in investigating the homologous but not the identical files, find easily relevant materials in other investigations.

Regarding the matter of computer world and its revolution, this conclusion indicates the mentioned improvements and solutions discovered and found to detect the unallowable access done by several types of attackers and hackers. As a result, the attacked data do not seem to be useless, but the rate of change shows that the data can be reused or not. Therefore, I think by inventing new hashes or modifying the existed hashing algorithms, we can determine the exact portion of hacked file or increase the number of restored files.

References:

1. Anderson, James P .1980. "Computer Security Threat Monitoring and Surveillance," Washing, PA, James P. Anderson Co.
2. Anderson, D. 1995. T. Frivold, and A. Valdes. Next-generation Intrusion Detection Expert System (NIDES) A Summary. Technical Report SRI-CSL-9 D. Anderson, T. F. Lunt, H. S. Javitz, A. Tamaru, and A. Valdes. Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES). Technical Report SRI-CSL-95-06, SRI Computer Science Laboratory, May 1995.5-07, SRI Computer Science Laboratory.
3. A.Menezes, P. Oorschot and S. Vanstone 1997. *Handbook of Applied Cryptography*. CRC Press.
4. B. Uppalaish, AnandNarsimha Swaraj, and T. Bharat, 2012. "Genetic algorithm approach to intrusion detection system," International Journal of Computer Science and Telecommunications, vol. 3.
5. Chyssler, Tobias, Stefan Burschka, Michael Semling, and Tomas Lingvall, KalleBurbeck, 2004. "Alarm Reduction and Correlation in Intrusion Detection Systems". In Detection of Intrusions and Malware & Vulnerability Assessment". Ulrich Flegel, Michael Meier (Eds) Dept. of Computer and Information Science Linköping University, S-581 83 Linköping , Sweden.
6. F.Breitinger and H. Baier, 2011, "Performance Issues about Context-Triggered Piecewise Hashing," in 3rd ICST Conference on Digital Forensics & Cyber Crime (ICDF2C), vol. 3.

7. Gassata, Me, L. 1998, a genetic algorithm as an alternative tool for security audit trail analysis. In "First International Workshop on the Recent Advances in Intrusion Detection".
8. Harbour, Nicholas 2002. Dcfldd. Defense Computer Forensics Lab. Available from: http://dcfldd.sourceforge.net. Last visit Feb 2015.
9. J. Gomez & D. Dasgupta, 2002. "Evolving Fuzzy classifiers for Intrusion Detection, Proceedings of 2002 IEEE Workshop in Information Assurance", USA NY.
10. Kornblum, J. 2006, "Identifying almost identical files using context triggered piecewise hashing," Digital Forensic Research Workshop (DFRWS), vol. 3S, pp. 91–97.
11. Mahbod,Tavallaee, EbrahimBagheri, Wei Lu, and Ali A. Ghorbani, 2009 "A Detailed Analysis of the KDD CUP 99 Data Set proceeding on".
12. Mallery, John 2008. "Network Intrusion Detection". Security Technology & Design. 44 – 47.
13. Mostaque, MD and M. Hassan, 2013. "Current studies on intrusiondetection system, genetic and fuzzy logic," International Journal ofDistributed and Parallel Systems (IJDPS), vol. 4.
14. Nazario, Jose. 2004, "Defense and Detection Strategies Against Internet Worms," Artech House, London.
15. Nitin,Mattord 2008. "Principles of Information Security.Course Technology". pp. 290–301. ISBN 978-1-4239-0177-8.3

16. Pokrywka, 2008 Rafał. pokrywkahttp://link.springer.com/chapter/10.1007%2F978-3-540-69384-0_45. Last visit Jan 2015.
17. Rouse, Margaret, 2006. http://searchmidmarketsecurity.techtarget.com/definition/intrusion-detection. LastseenFeb2015.
18. Roussev, Vassil 2013 Simson Garfinkel, Frank Breitinger, John Delaroderie, Barbara Guttman, John Kelsey, Jesse Kornblum, Mary Laamanen, Michael McCarrin, Clay Shields, Douglas White, John Tebbutt, and Joel Young. The NIST Definition of Approximate Matching.Technical report, National Institute of Standards and Technologies.
19. Roussev, Vassil, 2011. *An evaluation of forensic similarity hashes*. Digital Forensic Research Workshop, 8:34–41.
20. Roussev, Vassil. 2010. Data Fingerprinting with Similarity Digests. Internation federation for Information Processing, 337/2010:207–226.
21. Scarfone, Karen. 2007, "Guide to Intrusion Detection and Prevention Systems (IDPS)". Computer Security Resource Center (National Institute of Standards and Technology) (800–94). Retrieved 1 January 2010.
22. Steven, R. Snapp .1991, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Chelin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and DougMansur. *DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype*. In

Proceedings of the 14th National Computer Security Conference, pages 167-176, Washington, DC.

23. Teresa, F. Lunt. 1988, R. Jagannathan, Rosanna Lee, Sherry Listgarten, David L. Edwards, Peter G. Neumann, Harold S. Javitz, and Alfonso Valdes. IDES: The enhanced prototype -a real-time intrusion-detection expert system. Technical Report SRI-CSL-88-12, SRI International, 333 Ravenswood Avenue, Menlo Park, CA.
24. Zadeh, L A .1965. “Fuzzy Sets”, Information and Control, Vol.8, pp. 338-353.

ملخص البحث:

وفقا لتكنولوجيا المعلومات والإهتمام بثورات عالم الحاسوب، اصبح لهذا العالم ملفات ومعلومات وجب حمايتها من الهجمات المختلفة بأنواعها و التي تتسبب بإفسادها و تشويهها. لذلك، ظهرت العديد من الخوارزميات لزيادة مستوى الحماية و لتكشف عن جميع أنواع الهجمات. وعلاوة على ذلك، تقوم العديد من الخوارزميات مثل خوارزمية MD5 و خوارزمية SHA-1 بالكشف فيما اذا كان الملف قد تم مهاجمته أو إفساده أو تشويهه ام لا. إضافة إلى ذلك، يجب أن يكون هنالك خوارزميات أخرى من أجل الكشف عن مدى الضرر على الملفات المعرضة لهذا الخطر من أجل التأكد من إمكانية استخدام هذه الملفات بعد تأثرها بمثل هذه الهجمات. تعتبر كلاً من الخوارزميتين MD5 و SHA-1 الملف فاسد عندما يتم مهاجمته بغض النظر عن نسبة التغيير؛تم في هذا البحث استخدام خوارزمية SSdeep لانها تسمح بنسبة تغيير معينة بناء على طلب المستخدم. تعطي خوارزمية SSdeep نسبة تغيير حسب أهمية كل ملف، علاوة على ذلك كل نسبة تغيير تحدد إن كنا قادرين على إستخدام الملف ام لا. لذلك، تعد الخوارزمية SSdeep الطريقة التي تمنح المستخدم القدرة على الإستفادة من الملف الذي قد تم تغييره حسب نسبة التغيير؛ لذلك فمن الافضل استخدام خوارزمية SSdeep من أجل الإستفادة من الملف المعدل إن أمكن ذلك.