International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-5, June 2019

Concept Relation Logic for Visualizing File Behavior in a Knowledge Representation Model

Praveena Rachel Kamala S, Justus S

Abstract*: Malware attack is a major problem in cyber security. Malware attack causes problems like slowing down the computer, system crash, continuous display of error messages, shutting down or restarting certain activities, gather personal information or data from the system, redirect the browser, infect the system, spend spam mail and can even be invisible. The badly affected sectors due to cybercrime are movie & video production, commercial banking, health & medical insurance and warehouse clubs & supercentres. The file behavior monitoring method is used to detect the early access stage of malware attack. In this paper, using the Concept Relation Knowledge Representation (CRKR) framework model the file behavior is represented and visualized for monitoring the nature of the file. The aim of this work is to provide effectiveness among complex multiple featured files with uncertain and insufficient data. In this knowledge representation model the proposed Concept Relation Logic enables the features to be correctly placed in the visualization graph which helps in efficiently identifying the malware affected file.

***Index Terms*: Cyber Security, Knowledge Representation, Logic, Malware Attack, Visualization.**

I. INTRODUCTION

Internet has become part of our life. Most of global population is connected through internet [1]. Internet is mainly used in banking, entertainment, shopping and communication activities. On one side Internet has made our lives easy, but on the other side it has exposed us to risk of security and privacy.

Millions of malware software and thousands of hacker gangs are now part of cybercrime plundering today's online world. The numbers of reported attacks increase each year. This activity started as a game to steal sensitive information from users, now grown to level of deploying intelligent hackers to formulate malware programs to fetch details and commit financial frauds.

Proliferation of malware takes place as a threat only after fetching the software from the internet. In order to protect the system from such software, malware detection has become most important problem in cyber security. With the increasing risk of such cyber crime, in this paper we are proposing an application for visualizing the behavior of malware software with our Concept Relation Knowledge Representation (CRKR) model.

We will discuss the basic concepts such as types of malware in cyber security and knowledge based system in the session II, an overview about our CRKR framework model and the definition of Concept Relation (CR) Logic in the session III, then how our CRKR works on cyber security application to visualize the malware file is discussed in the session IV, demonstration of the experimental results in session V, analysis and the visualization of the features related to behavior of the file is discussed in session VI and finally conclude the paper with future work in session VII.

Manuscript published on 30 June 2019.
***** Correspondence Author (s)
Praveena Rachel Kamala S*, Research Scholar, School of Computing Science and Engineering, VIT University, Chennai, India.
Justus S, Associate Professor, School of Computing Science and Engineering, VIT University, Chennai, India.

II. BASIC CONCEPTS

A. Cyber Security

A Malware is a set of instructions that run on a computer and make the system do something that a hacker wants it to do. The various activities performed by them are steal personal information, delete files, click fraud, steal software serial numbers and use computer as relay. The various methods of malware are Virus, Backdoor, Trojan horse, Rootkit, Scareware, Adware and Worm [2].

Virus is a program that can modify within it to infect other programs. (Fred Cohen 1983) [13]. Virus types are classified into two; they are 1) Polymorphic which uses a polymorphic engine to mutate while keeping the original algorithm intact and 2) Methamorphic which change after each infection. Trojan describes a type of malware that pretends to perform a desirable function but in fact performs undisclosed malicious functions that allow unauthorized access to the victim computer[5]. A *Root kit* is a component that uses stealth to maintain a constant and untraceable presence on the machine. A *computer worm* is a self multiplying computer program. It uses a network to send copies of itself to other nodes and do so without any user intervention.

Ransomware either encrypt all the user files or restrict the user from accessing their files. *Adware* shows specific advertisements to the infected host that helps the attacker to generate revenue. This may not harm the victim directly but there might be some effect which may be created by some attacker. *Spyware* are generally used to spy on users and aim at stealing private information from the infected host without getting noticed and then send this information to the attacker. A *Bot* is software that allows the attacker to remotely access the host machine. A collection of Bots controlled by a single server is called a Botnet.

Backdoors allow access to a system through a remote terminal. It is used for both malicious and non-malicious purposes.The malware infects the system as executable, interpreted file, Kernel, service, Master Boot Record and Hypervisor [6].

This modifies the behavior of the file by compressing, encryption, randomizing, adding junk value, anti-debugging, anti Virtual Machine (VM) and virtualization.

Malware analysis is a process of dissecting a binary file to understand its working and the devising methods to identify it and other similar files. This aims at figuring out the actions performed by the malware software and then developing methods to neutralize those actions and prevent further infections. This process is not only performed by anti-virus companies, security administrators of various organizations but also by the system administrators to figure out the extent of damage to the system, network administrators to find and fix the damage in network and by the academic researcher to understand the behavior of malware and improve the existing security infrastructure [7],[8].

Malware analysis is used for both detecting and classification of malware. Malware detection is the first step of malware analysis. In malware detection process labeling of executable characteristic is accomplished. After identification, the malware analysis classifies the malware. Later on figure out its family of the malware and determine the remedial procedure [9].

Malware analysis can take place in two ways static and dynamic. Static analysis refers to analyzing data without execution, while in dynamic analysis is determining its behavior on execution.

B. Knowledge Based System

Computer programs are coded with the specific aim of solving problems. One classic method for obtaining solution by tracing to its origin is done by knowledge based system. The world is flooded with both relevant and irrelevant information. Data is very rarely available in structured format and mostly it is non-structured. Knowledge based systems are expertise computer systems in a particular domain, where huge volume of data are analyzed and processed in very short time. The knowledge collected and stored is used for solving the problem in the particular domain they are deployed. Relevant knowledge can be extracted from the knowledge base provided the knowledge is represented properly.

We have developed a framework for knowledge representation. In this model, the data is acquired from a source containing both related and irrelevant data. Knowledge Representation Model will give clear knowledge about the scenario, streamline the doubts and clarify the mismatches.

III. PROPOSED METHODOLOGY - CONCEPT RELATION KNOWLEDGE REPRESENTATION (CRKR) MODEL

Our proposed system for Knowledge Representation has several stages, they are:

Stage 1 – Parsing

Stage 2 – Representation

Stage 3 – Concept Relation Logics

Stage 4 – Visualization of Concepts

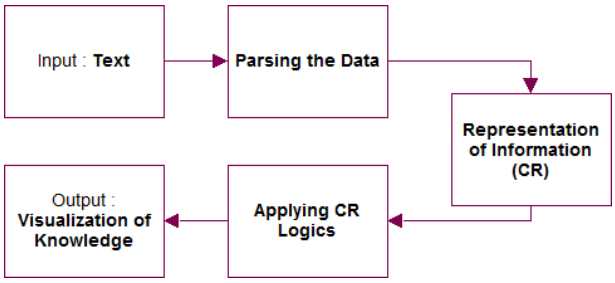

Fig. 1: Stages in Knowledge Representation Model

Parsing: In this stage the data is acquired from the source and filtering of the content is done as in Fig. 1. As we are dealing with unstructured data, the source of information will be a collection of related or unrelated text data. Hence the data has to be processed and segregated with at most care otherwise content based on the context can be misinterpreted.

In this approach we have proposed a content parsing algorithm, which accepts the text input of any size and analyses sentence after sentence by using a stop (.) indicator. This result helps us to identify the concepts and related concepts, enabling easy understanding of the context.

Representation: In this stage the parsed content are to be represented properly which in turn helps in clear visualization. Now the concepts are separated from the relations and are indicated using different symbols. The usage of symbols is due to easy reorganization of the content. The concepts are represented by oval boxes or '()' brackets and relations are represented by square boxes or '[]' brackets.

In this Concept Relation (CR) Representation, each concept has to be mapped to a relation than to another concept. The representation cannot begin or end with a relation (i.e.) concept is mapped to another concept through a relation.

Concept Relation Logic:

Every concept is linked with another concept through a relation. In other words, every concept is related to another concept. Identifying the concepts and their relation is a challenge. In conceptual graph, the function of quantifier are represented as boxes called concepts and arcs marked with arrows show the connections of arguments to circles called conceptual relations [4].

Definition of Basic Elements

Concept: The concept of entities, attributes are denoted as concepts. The representation of concept is depicted as ( ).

Relation: These denote the occurrence of the concept they are linked to. The representation of relation is depicted by [ ].

Algorithm: The parsed sentence contains both concepts and relation words. The representation algorithm identifies and classifies the word as either concept or relation.

CR_Logic({Concept$_1$,..,Concept$_n$},{relation$_1$,….,relation$_m$})

Input: Sentence (Concept$_1$, relation$_1$.. relation$_m$,Concept$_n$),
Content {Concept$_1$,..,Concept$_n$} and relations {relation$_1$,….,relation$_m$}

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

Output: Entire content converted into CR_Logic
$(Concept_1)$-[:$relation_1$]->$(Concept_2)$-…. $(Concept_{n-1})$-[:$relation_m$]->$(Concept_n)$
/* Relation Table (RT) contains a set of relations */
/* Concept Table (CT) contains clauses, which are existing in the graph */
/* Sentence Table (ST) contains the parsed sentences */

Procedure:

DETERMINE the length of the content (words) in the sentence
INITIALIZE count=0, word, $crword_l$
word = $crword_1$
RETURN ((word)-)
count++
IF count < length-1 of the content
word = $crword_{2..l-1}$
CHECK IF the word is in RT
RETURN ([:word]->)
count++
ELSE
CHECK IF the word is present in CT
RETURN ((word)-)
count++
crword++
ELSE
RETURN ((word))
END

As a pre-requisite, the passage or the input text undergoes the splitting process using the parsing algorithm. Further, the concepts and relations are identified and stored in the concept table (CT) and relation table (RT) respectively. Parsed sentences are stored in the sentence table (ST). Each concept and relation is stored with a unique identification code in the tables.

Now, to convert the contents from the representation model to a logical model we use the Concept Relation (CR) Logic. In the CR logic generation algorithm, each word is taken and defined according to their occurrence in the sentence.

Initially the first word in the sentence is taken as the concept and represented with a definition of concept. Then the second word till the last but one word in the sentence are taken one by one and compared with their occurrence as concept or relation from concept table (CT) and relation table (RT) respectively. If the word is present in the CT then the word is formulated with the definition of concept, otherwise defined as relation. Last word in the sentence is defined as the concept.

Generated CR Logic:

(Concept1)-[:relation1]->(concept2)-[:relation2]->(concept3)-[:relation3]->(concept4)-[:relation4]->(concept5)

Inference: Each Concept is mapped with its relation and relation in turn is connected to a concept, this illustrates their relationship between concepts. The CR logic is formulated by beginning and ending with a concept. (i.e.) starting word and the last word should be concepts (C) – [:r]-> (C).

The selected concept will be the root node and the related concepts to the selected node will be the connected surrounding nodes with level 1. Further, when the nodes associated with the related concept are retrieved, the level increases by 1. This process can be continued till all the related nodes are retrieved.

Visualizing: In this step the identified and represented concepts along with their related concepts are visualized in graphical structure. The signs for the concepts along with their relations are formulated using the Concept Relation (CR) Logic Notation Algorithm. Thus the graph for viewing the content is produced.

The entire content from the source can be easily viewed by the user for simple understanding. For specific concept to be seen, a Retrieval Algorithm [12] for concept searching is deployed. In this algorithm the searched concept if available will be displayed and further the associated concepts can be retrieved by extending the search.

IV. CRKR ON CYBER SECURITY

Our framework for knowledge representation using Concept Relation (CR) Logic is used for placing the parameters and visualizing the abnormality of the file characteristic features. As a primary procedure the source data from the Portable Executable (PE) file, assembly file and kernel are recorded for monitoring purpose.

The various ways for extracting the source data are as follows: The Application Program Interface (API) comprises of a set of commands, functions and protocols which is used to build software for a particular operating system. API's are defined functions to build new software. The data pertaining to features of the files are collected as byte sequences. The difference in frequency of byte code is analyzed for malicious code. Every assembly file contains a list with opcode and each opcode sequence frequency is analyzed for difference. The features from kernel are collected as system calls (i.e.) files, registries, process and threads, networking and memory sections. The PE files consist of number of headers and sections which are organized as a stream of linear data. The different features are DLL's, API function calls, frequency of the instruction code, API call sequences, text based search technique, hashing, attribute certificate, date/time stamp, file pointer, linker information, CPU type, PE logical structure (section alignment, code size, debug flags) etc. [2]

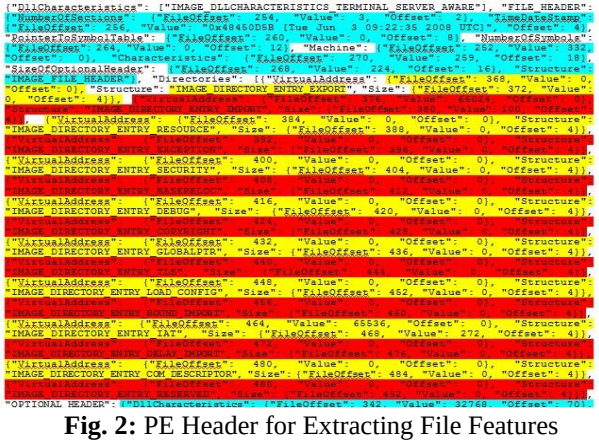

Fig. 2: PE Header for Extracting File Features

We extract the features from PE header and sections. The above Fig. 2 shows the DLL characteristics. From all the features extracted only seven features are considered for analyzing the behavior of malware[3].

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

The features are Debug Size, Image Version, IatRVA, Export Size, Resource Size, VirtualSize2 and Number of Sections.

Debug Size: Indicates the size of debug-directory table. Normally the executable files have debug directory and clean programs mostly have non-zero value for Debug Size.

IatRVA: Indicates the relative-virtual address of the import-address table. The value of this feature is 4096 for clean programs and 0 or very large for virus files. Mostly, import functions may not be used by malware or might obfuscate their import tables.

Export Size: Indicates the size of the export table. Non executable programs like DLLs have export table. Hence the value will be non-zero for clean files and 0 for virus files.

Image Version: Indicates the version of the file. It is not associated to the function of the program. Majority of the clean programs have many versions and a bigger image-version set, whereas for malware software the value of Image Version is 0.

Resource Size: Indicates the size of the resource section. Some of the virus files may have no resources and clean files have larger resources.

Virtual Size2: Indicates the size of the second section. Virus files mostly have only one section and this field value will be 0 for them.

Number of Sections: Indicates the number of sections. This feature value varies for both virus and clean files. This feature does not give a clear indication for separating malware and clean files.

These features are selected based on the occurrence of the values by comparing thousands of samples of both clean and malware affected files. The accuracy values of these seven features using Feature Selection Algorithm are as in the Table I [3].

Features	Accuracy
Debug Size	0.9234
IatRVA	0.8249
Export Size	0.8146
Image Version	0.8898
Resource Size	0.8025
Virtual Size2	0.7839
Number of Sections	0.5438

Table I – The Accuracy Values of the seven Features based on Feature Selection Algorithm

The Accuracy values of file features using KNN are as in the Table II [11].

Features	Accuracy
Debug Size	91.18
IatRVA	91.59
Image Version	88.68
Resource Size	90.40
Number of Sections	91.18
Virtual Size2	89.92

Table II – The Accuracy Values of the seven Features based on KNN Classifier

Comparing the accuracy values of both Feature Selection Algorithm and KNN Classifier, the Fig. 3 shows their exactness is choosing the features for identifying clean or infected file. Debug size and image version plays an important role in the process of identification.

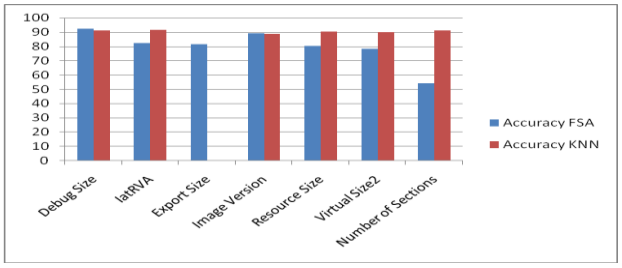

Fig. 3: Comparison of Feature Accuracy FSA vs KNN

The Tables III and IV are a few sample values obtained for the seven features from both clean and malware files respectively. These values are used in our model representation and logic generation.

FileName	DebgSz	IatRVA	ExpoSz	ImgVer	ResoSz	VirtSz2	No. of Sect
autochk.exe	28	375904	0	501000	183528	21440	4
autodisc.dll	28	26584	254	501000	52472	272	4
calc.exe	28	76672	0	501000	64000	4124	3
hhsetup.dll	28	29880	6918	502000	944	1132	4
msoffice 2007 service pack 2.exe	0	163840	0	600000	17624	140	8
fontext.dll	28	171396	164	501000	190600	27260	4
comdlg32.dll	56	192628	689	501000	66056	13344	4
dsound.dll	56	346040	435	501000	1296	7976	4
dllhost.exe	28	5808	0	501000	29696	60	3
dvdplay.exe	28	5180	0	501000	80384	40	4

Table III – The Values Extracted from PE Header for Clean Files

Malware name	DebgSz	IatRVA	ExpoSz	ImgVer	ResoSz	VirtSz2	No. of Sect
Adware	0	5976	0	0	68280	69632	2
Backdoor	0	5976	0	0	68280	69632	2
Backdoor	0	200704	0	0	1952	170384	5
Ramsomeware	0	1314816	2700	100000	744	3760	11
Trojan	0	13308	64	0	155136	1888	5
Unknown	0	163840	0	600000	17624	140	7
Sdbot	0	122968	0	0	32768	14106	5
Spam	0	81504	0	0	1203044	5202	4

Table IV – The Values Extracted from PE Header for Malware Files

The values are converted as text for each and every file. The data is given as input for our Concept Relation Knowledge Representation (CRKR) Model. The parsing algorithm is deployed to split the given passage into smaller units. The concepts and relations are identified for representation. In this case the file type, file / malware name and values are identified as concept and the features are indicated as relation. The concept relation mapping is done. Any modification and deletion of concepts can be performed by using our modification algorithm.

All the values are updated in the concept table and relation table along with the sequence number which indicates their order of occurrence. Using this information our Concept Relation (CR) Logic is generated. The first word in the sentence is taken as the concept based on the understanding that every sentence starts with a concept followed by a relation to another concept. Hence, the first word and the last word in the sentence will be concepts.

Each concept is connected to another concept through a relation. Therefore, the next word after a concept is checked in the relation table (RT), if not found then the word is verified in the concept table (CT). Depending upon their presence the logical syntax is assigned. If they are continuous concepts then they are connected without using a defined relation, assuming that they are interconnected concepts. Thus the process continues till the last but one word and the last word in the sentence is taken as a concept.

This is the procedure by which the specified features from the clean and malware affected files are taken, represented using the CRKR model and from the mapping, the CR logic is generated for reasoning and visualizing the knowledge from the created knowledge base.

V. EXPERIMENTAL RESULTS

A. Representation of a single uninfected file: (autodisc.dll)

A single uninfected file (autodisc.dll) is taken and the values of the selected features are available in the table 2 from above. As the first step the information is given to the system as a text input as in Table V. The identification of concepts and relations takes place and are mapped using the CRKR algorithm as in Fig. 3. The flow of information from the text input is understood from the representation.

Further by using proposed CR logic algorithm, the content is converted into CR logic in Table VI. The generated logic is converted into cyber query language for visualization as in Fig. 4.

FileName autodisc.dll debugSize 28. autodisc.dll IatRVA 26584. autodisc.dll ExportSize 254. autodisc.dll ImageVersion 501000. autodisc.dll ResourceSize 52472. autodisc.dll VirtualSize 272. autodisc.dll NumberOfSections 4.

Table V – Input for Single Clean File

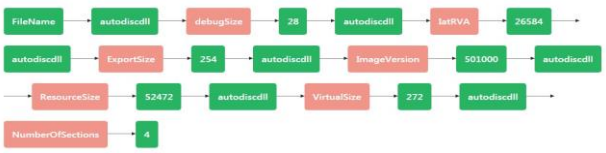

Fig. 3: CRKR Text Representation for Single Clean File

(filename)-[:_]->(autodiscdll)-[:debugSize]->(28) , (autodiscdll)-[:IatRVA]->(26584), (autodiscdll)-[:ExportSize]->(254), (autodiscdll)-[:ImageVersion]->(501000), (autodiscdll)-[:ResourceSize]->(52472), (autodiscdll)-[:VirtualSize]->(272),
(autodiscdll)-[:NumberOfSections]->(4)

Table VI – CR Logic for Single Clean File

Fig. 4: CRKR Graphical Visualization of Single Clean File

This visualization of the content related to autodisc.dll, enables easy understanding of the information in a graphical structure. The concepts filename and their values are related by their feature, which we have selected for analysis. Since it is a clean file, all the values are distinct and non zero.

B. Representation of multiple infected files: (Trojan and Backdoor)

In this case more than one infected file is taken and represented. The values of the selected parameter are passed as input as in Table VII. The mapping of concept is done using CRKR model in Fig. 5 and generate the logic using CR logic in Table VIII.

malwareName Trojan debugSize 0. Trojan IatRVA 13308. Trojan ExportSize 64. Trojan ImageVersion 0. Trojan ResourceSize 155136. Trojan VirtualSize 1888. Trojan NumberOfSections 5. malwareName Backdoor debugSize 0. Backdoor IatRVA 200704. Backdoor ExportSize 0. Backdoor ImageVersion 0. Backdoor ResourceSize 1952. Backdoor VirtualSize 170384. Backdoor NumberOfSections 5.

Table VII – Input for Two Malware Affected Files

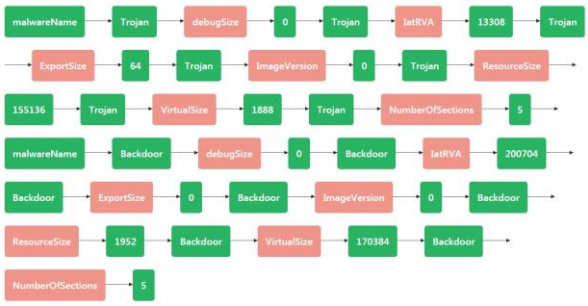

Fig. 5: CRKR Text Representation for Two Malware Affected Files

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

(malwarename)-[:_]->(trojan)-[:debugSize]->(0),
(trojan)-[:IatRVA]->(13308),
(trojan)-[:ExportSize]->(64),
(trojan)-[:ImageVersion]->(0),
(trojan)-[:ResourceSize]->(155136),
(trojan)-[:VirtualSize]->(1888),
(trojan)-[:NumberOfSections]->(5),

(malwarename)-[:_]->(backdoor)-[:debugSize]->(0)
,
(backdoor)-[:IatRVA]->(200704),
(backdoor)-[:ExportSize]->(0),
(backdoor)-[:ImageVersion]->(0),
(backdoor)-[:ResourceSize]->(1952),
(backdoor)-[:VirtualSize]->(170384),
(backdoor)-[:NumberOfSections]->(5)

Table VIII – CR Logic for Two Malware Affected Files

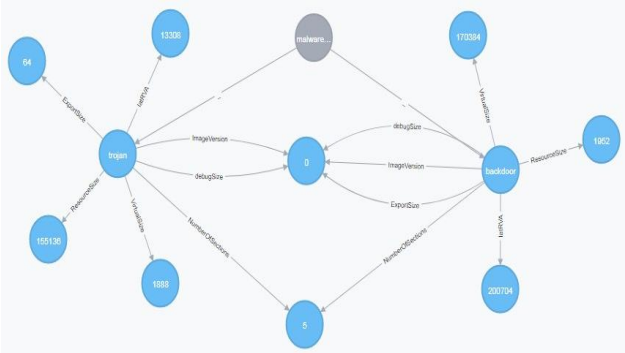

Fig. 6: CRKR Graphical Visualization of Two Malware Affected Files

Fig. 6 enables the visualizing process to take part effectively. In this case, the debug size, and image version values are zero. In addition the export size for backdoor malware software is also zero. Based on the analysis made [3] [11] for selecting the malware affected files, this visualization model yields its best for detecting the files.

VI. Analysis and Discussion

There are many types of files installed to the system at various instance of time. The nature of the file is determined by the parametric values. Their behavior is well assessed based on the performance metrics. Several studies have been taking place in various forms to detect and prevent the system from getting infected. Our Concept Relation Knowledge Representation Model enables the analyzer to visualize the information and to monitor the behavior of the file efficiently. The various stages are the text input based on the values is given, the identification of the concepts is done, a mapping of concepts along with their relation is performed, and based on the identified concepts and their relations, and the CR logic is generated using the Concept Relation (CR) Logic Algorithm.

Text Input: (comdlg32_dll, dsound_dll, dllhost_exe, dvdplay_exe, Sdbot and Spam) as in Table IX.

As input, the values pertaining six different files are given as a text. The CRKR identifies the concepts and map them with their relationship.

FileName comdlg32_dll debugSize 56.
comdlg32_dll IatRVA 192628.
comdlg32_dll ExportSize 689.
comdlg32_dll ImageVersion 501000.
comdlg32_dll ResourceSize 66056.
comdlg32_dll VirtualSize 13344.
comdlg32_dll NumberOfSections 4.

FileName dsound_dll debugSize 56.
dsound_dll IatRVA 346040.
dsound_dll ExportSize 435.
dsound_dll ImageVersion 501000.
dsound_dll ResourceSize 1296.
dsound_dll VirtualSize 7976.
dsound_dll NumberOfSections 4.

FileName dllhost_exe debugSize 28.
dllhost_exe IatRVA 5808.
dllhost_exe ExportSize 0.
dllhost_exe ImageVersion 501000.
dllhost_exe ResourceSize 29696.
dllhost_exe VirtualSize 60.
dllhost_exe NumberOfSections 3.

FileName dvdplay_exe debugSize 28.
dvdplay_exe IatRVA 5180.
dvdplay_exe ExportSize 0.
dvdplay_exe ImageVersion 501000.
dvdplay_exe ResourceSize 80384.
dvdplay_exe VirtualSize 40.
dvdplay_exe NumberOfSections 4.

MalwareName Sdbot debugSize 0.
Sdbot IatRVA 122968.
Sdbot ExportSize 0.
Sdbot ImageVersion 0.
Sdbot ResourceSize 32768.
Sdbot VirtualSize 14106.
Sdbot NumberOfSections 5.

MalwareName Spam debugSize 0.
Spam IatRVA 81504.
Spam ExportSize 0.
Spam ImageVersion 0.
Spam ResourceSize 1203044.
Spam VirtualSize 5202.
Spam NumberOfSections 4.

Table IX – Text Input for multiple file (both Clean and infected files)

Logic:

The CR Logic is generated based on the identified concepts. Each concept is related to another concept through a relation. This generation of logic is based on the algorithm proposed. Each concept is taken as a node and relation connecting each node as the edge between the nodes. This enables the analyst to visualize the selected parametric value clearly. The generated CR logic is as in Table X.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

(filename)-[:_]->(comdlg32dll)-[:debugSize]->(56),
(comdlg32dll)-[:IatRVA]->(192628),
(comdlg32dll)-[:ExportSize]->(689),
(comdlg32dll)-[:ImageVersion]->(501000),
(comdlg32dll)-[:ResourceSize]->(66056),
(comdlg32dll)-[:VirtualSize]->(13344),
(comdlg32dll)-[:NumberOfSections]->(4),

(filename)-[:_]->(dsounddll)-[:debugSize]->(56),
(dsounddll)-[:IatRVA]->(346040),
(dsounddll)-[:ExportSize]->(435),
(dsounddll)-[:ImageVersion]->(501000),
(dsounddll)-[:ResourceSize]->(1296),
(dsounddll)-[:VirtualSize]->(7976),
(dsounddll)-[:NumberOfSections]->(4),

(filename)-[:_]->(dllhostexe)-[:debugSize]->(28),
(dllhostexe)-[:IatRVA]->(5808),
(dllhostexe)-[:ExportSize]->(0),
(dllhostexe)-[:ImageVersion]->(501000),
(dllhostexe)-[:ResourceSize]->(29696),
(dllhostexe)-[:VirtualSize]->(60),
(dllhostexe)-[:NumberOfSections]->(3),

(filename)-[:_]->(dvdplayexe)-[:debugSize]->(28),
(dvdplayexe)-[:IatRVA]->(5180),
(dvdplayexe)-[:ExportSize]->(0),
(dvdplayexe)-[:ImageVersion]->(501000),
(dvdplayexe)-[:ResourceSize]->(80384),
(dvdplayexe)-[:VirtualSize]->(40),
(dvdplayexe)-[:NumberOfSections]->(4),

(malwarename)-[:_]->(sdbot)-[:debugSize]->(0),
(sdbot)-[:IatRVA]->(122968),
(sdbot)-[:ExportSize]->(0),
(sdbot)-[:ImageVersion]->(0),
(sdbot)-[:ResourceSize]->(32768),
(sdbot)-[:VirtualSize]->(14106),
(sdbot)-[:NumberOfSections]->(5),

(malwarename)-[:_]->(spam)-[:debugSize]->(0),
(spam)-[:IatRVA]->(81504),
(spam)-[:ExportSize]->(0),
(spam)-[:ImageVersion]->(0),
(spam)-[:ResourceSize]->(1203044),
(spam)-[:VirtualSize]->(5202),
(spam)-[:NumberOfSections]->(4)

Table X – Generated CR Logic for Multiple files

Visualize:

These are different versions of viewing the content given as input.

a. *Display the roots*: The name of the various files depending upon their behavioral nature is grouped as Filename and Malwarename for clean and infected files respectively. This information is given as input. Hence, these names are taken as the root node and are displayed as the root in Fig. 7.

Fig. 7: Visualizing only the root nodes

b. *Expand the root nodes*: To view the files added under each root, the respective root nodes are expanded.

1. *Normal File Nodes*: As an initial case, the root node with name Filename is expanded which yields all the files added with the caption as Filename as in Fig. 8.1.

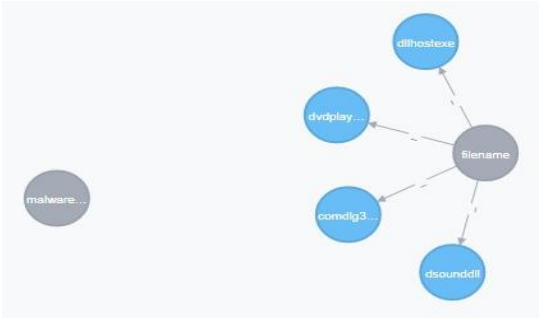

Fig. 8.1: Visualizing the Clean File Nodes

2. *Malware Nodes*: In this case, all the nodes corresponding to Malewarename are displayed as in Fig 8.2.

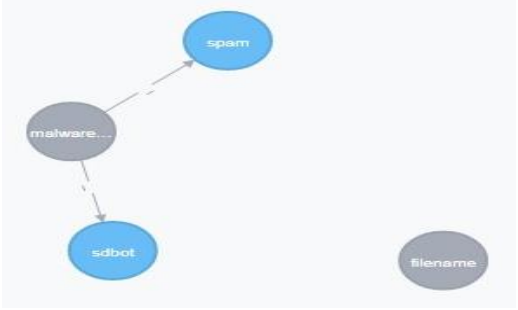

Fig. 8.2: Visualizing the Infected File Nodes

3. *Both nodes expanded*: Here, both the root nodes are expanded exhibiting all the related nodes of the next level. This view gives an overall picture of all the files added and also signifies the category under which each file is grouped (i.e.) root node shows the type of file as in Fig. 8.3.

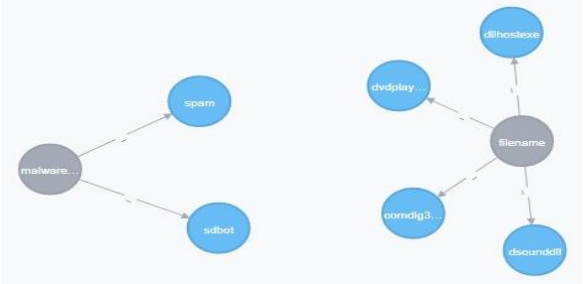

Fig. 8.3: Visualizing both Clean and Infected Filenames

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

c. *Expanding the sub-root nodes*:

1. *Expanding 'Spam' node*: On expanding the node named 'Spam', all the other related nodes are displayed. The nodes sharing the same property are also exhibited. Thus all the related concepts are viewed as in Fig. 9.1.

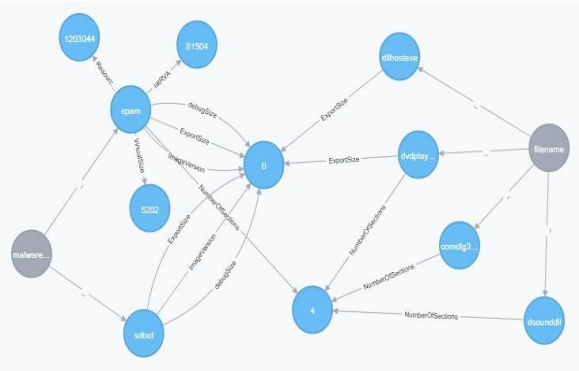

Fig. 9.1: Visualizing Infected Filenamed 'Spam'

2. *Expanding 'Dvdplay' node*: Further, on expanding another node 'Dvdplay' the various related nodes are presented. Thus making the graph look denser as on the figure below with the related concepts as in Fig. 9.2.

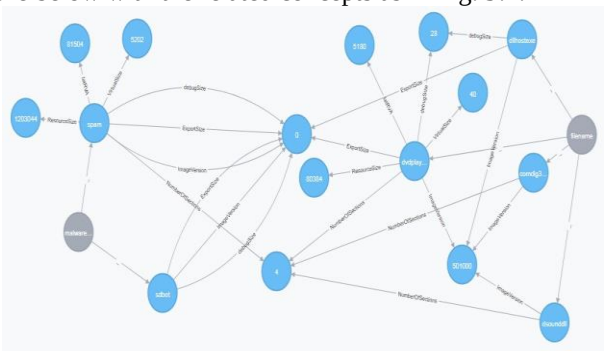

Fig. 9.2: Visualizing a Clean Filenamed 'Dvdplay'

d. *Complete graph on expanding all the nodes*: On expanding all the nodes, the graphical structure demonstrates the various concepts which are feed into the system mapping with their relations to each other as in Fig. 10.

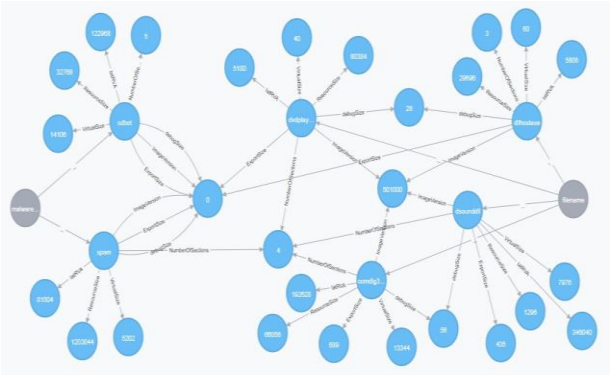

Fig. 10: Visualizing the Complete Graphical Structure

Observation: From the complete graph, the behavior of the files is assessed. Based on the previous study the parametric values indicate the nature of infected files by containing zero values. The non-infected files consist of non-zero values. From the accuracy comparison, the study signifies that debug size and image version plays an important role in infected files. Thus enabling the system to monitor and detect the presence of malware attacked file in the system.

This monitoring mechanism helps to visualize the values of all the files in a graphical structure. The CRKR model indentifies the concept along with their relations and with the support of CR logic displays the complete structure in a graphical form.

VII. CONCLUSION

In this paper, to visualize the behavior of infected file our proposed CR logic is used. Initially the selected parameters for each file are taken for representation using CRKR model. At this stage, the concepts along with their relation are identified and represented. Later using CR logic, each scenario is specifically defined based on their input and relation with other concepts.

This generated CR logic is interpreted into a graphical structure, making the monitoring process simpler and easier for the analyzer to visualize. Each concept node is mapped to another concept node using their relation edge. This enables the analyzer to clearly view the related concepts through the graphical structure. Resulting to analysis their similarities and indentify the existence or non-existence of infected file.

This CR knowledge representation model segregates the concepts from the given text input and maps along with related concepts by using the CR logic. This model requires no pre-training of the situation. For any given input by the user, the representation of information is done. Based on the information feed by the user the knowledge is created by relating each concept and defining explicitly.

As future development, the abnormal values can be indicated in the graphical structure making the monitoring process easier and optimizing the input text content for clearly representing the knowledge.

REFERENCES

1. ICT(2016). ICT: Facts and Figures. http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2016.pdf.
2. Ishita Basu et al, "Malware Detection Based on Source Data using Data Mining : A Survey", American Journal of Advanced Computing, Vol. III (1), pp.18-37, January 2016.
3. Karthik Raman, "Selecting Features to Classify Malware", © 2012 Adobe Systems Incorporated.
4. John F. Sowa "Syntax, Semantics, and Pragmatics of Contexts", AAAI Technical Report FS-95-02, 1995.
5. Asaf Shabtai, Eitan Menahem and Yuval Elovici, "F-Sign: Automatic, Function – Based Signature Generation for Malware", IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews, Vol. 41, No. 4, pp. 494 - 508, July 2011.
6. Mehryar Rahmatian, Hessam Kooti, Ian G. Harris and Elaheh Bozorgzadeh, "Hardware-Assisted Detection of Malicious Software in Embedded Systems", IEEE Embedded Systems Letters, Vol. 4, No. 4, pp. 94 – 97, December 2012.
7. Zhiyong Shan and Xin Wang, "Growing Grapes in Your Computer to Defend Against Malware", IEEE Transactions on Information Forensics and Security, Vol. 9, No. 2, pp. 196 – 207, February 2014.
8. Andrea Saracino, Daniele Sgandurra, Gianluca Dini and Fabio Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention", IEEE Transactions on Dependable and Secure Computing, Vol. 15, No.1, pp. 83 – 97, January/February 2018.
9. Taejoo Cho, Hyunki Kim and Jeong Hyun Yi, "Security Assessment of Code Obfuscation Based on Dynamic Monitoring in Android Things", IEEE Special Section on Security and Privacy in Applications and Services for Future Internet of Things, Volume 5, pp. 6361 – 6371, 2017.

10. S. Praveena Rachel Kamala and Dr. S. Justus, "Towards MORK: Modle for Representing Knowledge", I. J. Modern Education and Computer Science. 2016; pp. 45-53.
11. Udaykumar N et al, "Malware Category Prediction Using KNN and SVM Classifiers", International Journal of Mechanical Engineering and Technology. 2019; pp. 787-797.
12. S. Praveena Rachel Kamala and Justus Selwyn, "Concept-Relation Constructs for Knowledge Representation", International Journal of Control Theory and Applications, Volume 9, Number 52, 2016, pp. 463-473.
13. https://www.cisco.com/c/en/us/about/security-center/virus-differences.html

AUTHORS PROFILE

Praveena Rachel Kamala S is a full time Ph.D. Research Scholar at the School of Computer Science and Engineering, VIT University, Chennai, TN, India. Her areas of research include Knowledge Engineering, Knowledge Representation, Knowledge Storage and Retrieval.

She has received her undergraduate and postgraduate degrees in Computer Science & Engineering from Madras University and Anna University, Chennai. She has worked as lecturer in several Engineering Educational Institutions in India.

Dr. S. Justus received his doctorate degree from Madurai Kamaraj University, Madurai, India. His research specializations include Object-relational data modeling, knowledge engineering and Big Data. He has been into academic research and has published several of his research work results in International Journals and Conferences – including SwSTE in Israel and DASMA in Germany. He has also practitioner's experience while working with Software development companies. He is a member of IEEE, ISTE, IAENG professional associations. He has served as research & project coordinator for PG studies at Engineering Institutes. Presently he is working as Associate Professor at VIT University, Chennai.