

Applying FML and Fuzzy Ontologies to Malware Behavioural Analysis

Hsien-De Huang

National Center for High-
Performance Computing
NARL
Tainan, Taiwan
TonTon@ieee.org

Giovanni Acampora
Vincenzo Loia

Dept. Computer Science
University of Salerno
Italy
acampora@unisa.it

Chang-Shing Lee

Dept. Computer Science and
Information Engineering
NUTN
Tainan, Taiwan
leecs@mail.nutn.edu.tw

Hung-Yu Kao

Dept. Computer Science and
Information Engineering
NCKU
Tainan, Taiwan
hykuo@mail.ncku.edu.tw

Abstract—Antimalware applications represent one of the most important research topic in the area of information security threat. Indeed, most computer network issues have malwares as their underlying cause. As a consequence, enhanced systems for analyzing the behavior of malwares are needed in order to try to predict their malicious actions and minimize eventual computer damages. However, because the environments where malwares operate are characterized by high levels of imprecision and vagueness, the conventional data analysis tools lack to deal with these computer safety applications. This work tries to bridge this gap by integrating semantic technologies and computational intelligence methods, such as the Fuzzy Ontologies and Fuzzy Markup Language (FML), in order to propose an advanced semantic decision making system that, as shown by experimental results, achieves good performances in terms of malicious programs identification.

Keywords—malware behavioural analysis; ontology; fuzzy ontology; fuzzy markup language

I. INTRODUCTION

In recent years, the rapid growth of information technology has strongly enhanced computer systems both in terms of computational and networking capabilities. However, these novel distributed computing scenarios, other than enabling the development of efficient computer applications, have raised important issues about the security and safety of the interconnected hardware systems. Indeed, computer threats, such as viruses and backdoor programs have created different economical and technological disasters all around the world. Internet fraud activities [1], users private data stealing, malwares damaging computer systems without the owner's informed consent [2], are only some examples of computer threats each day sabotage personal, academic and industrial computer equipments. As a consequence, enhanced systems for analyzing the behavior of malwares are needed in order to try to predict their malicious actions and minimize eventual computer damages [3]. However, because the environments where malwares operate are characterized by high levels of imprecision and vagueness, the conventional data analysis tools lack to deal with these computer safety applications. This work tries to bridge this gap by integrating different computational intelligence methods, such as the Fuzzy Ontologies [4] and Fuzzy Markup Language (FML) [5].

Ontologies are structural frameworks for modeling

information by means of an explicit specification and a shared conceptualization [6]; they are mainly used as good knowledge representation and communication model for intelligent agents [4]. Nevertheless, in their conventional view, ontologies are not enough to deal with imprecise and vague knowledge. As a consequence, many researchers have explored the use of fuzzy ontologies, i.e., structural frameworks based on fuzzy logic used for modeling information characterized by high levels of imprecision and vagueness. For example, Lee *et al.* proposed a fuzzy ontology for designing an intelligent decision making system for summarization systems [4]; Quan *et al.* presented the automatic fuzzy ontology generation for semantic help desk support [7], and the automatic fuzzy ontology generation for semantic web [8]. Typically, the joint exploitation of Fuzzy Ontologies and FML is mainly used as support framework for designing intelligent fuzzy inference systems, one of the key topics in artificial and computational intelligence research [9].

The FML [10] is a novel computer language useful for modeling advanced fuzzy systems by means of a structural and taxonomic approach that improves fuzzy systems design from a twofold point of view: transparency and reliability (see section II.B). This paper presents an FML-based ontological agent for malware behavioral analysis. In particular, this agent uses a fuzzy ontology for modeling malware behavioral knowledge collected by domain experts and our homemade tool, Taiwan Malware Analysis Net (TWMAN) [1, 3]. An FML program is used in order to design an intelligent decision making system whose behavioral rules are used to detect viruses and malicious programs and, consequently, provide a better protection to computer equipments.

This paper contains five sections. Section II describes the literature survey about Web Ontology Language (OWL), FML and TWMAN. Section III describe the integration of a fuzzy ontology model and FML for modeling the knowledge about malware behaviors and propose an intelligent decision making system for detecting computer anomalies. Section IV discusses the experimental results of this study. Finally, some conclusive remarks are offered in Section V.

II. BASIC CONCEPTS AND BACKGROUND

Today, there is a growing interest in the development of systems based on knowledge representation by means of ontological approaches. Nevertheless, conventional ontology languages are not appropriate to deal with uncertainty [11],

vagueness or imprecision which strongly characterize the real world scenarios. It is not surprising that fuzzy set theory and fuzzy logic are handle these types of knowledge and, indeed, fuzzy ontologies are emerging as a useful methodology for knowledge representation in several semantic-oriented applications. In this paper, a decision making system for analyzing malware system behavior is implemented by integrating the following technologies: Web Ontology Language (OWL), Fuzzy Markup Language (FML) and Taiwan Malware Analysis Net (TWMAN).

A. Web Ontology Language (OWL)

When a malware behavioral analysis tool owns the malwares source code at its disposal, the analysis step becomes quite simple. Indeed, understanding the malicious program is as simple as reading, interpreting and looking for code fragments that reveal key information on the attacker and the attack. However, our approach does not own malwares source codes to detect eventual viruses or similar programs, but it exploits an ontological view of malwares behavior in order to define a more general and efficient detection platform. In order to express the ontological knowledge about malware behavior, our proposal uses the OWL2 language, a standard tool for modeling knowledge in a semantic way. In particular, our framework uses OWL for describing an ontological model for malware behavioral (see Fig. 1) composed by the following information: *File Hash (FH)*, *Connect IP (CI)*, *Files Change (FC)* and *Similarity (SI)* (see section II.C) [1]. Precisely, File Hash is a malware information (computed by the *ssdeep* program [12]) useful to detect the presence of malwares on the system under protection by comparing the hash code of a suspicious file with a collection of malware hash codes previously gathered. Connect Ip allow us to count the number of TCP/IP connection opened by a given program and, as a consequence, to detect anomalous network behaviors; this information is computed by INetSim [13], a software suite for simulating common internet services for analyzing the network behavior of unknown malware samples; File change is a malware information used to evaluate the number of changes occurred in the file system due to some eventual malware actions; it is computed by the Advanced Intrusion Detection Environment (AIDE) [14]. The Similarity concept is a cumulative information used to evaluate the level of maliciousness of a program; it is computed starting from FH, CI and FC. However, even though OWL enables a suitable representation of malwares knowledge, it is not able to apply advanced inference mechanisms to derive additional imprecise and vague knowledge in the scenario of malwares detection. For this reason, we exploit FML, an innovative fuzzy inference language bridging the gap shown by other methodologies.

B. Fuzzy Markup Language (FML)

FML is a domain-specific language proposed by Acampora and Loia [10] whose main aim is to model fuzzy systems by directly dealing with fuzzy concepts, fuzzy rules and fuzzy inference engines [9]. In particular, it is used to model two well-known kind of fuzzy controllers: Mamdani and Takagi-Sugeno-Kang (TSK). This language is XML-based, so FML programs are coded through a collection of correlated semantic XML tags, following an opportune XML Schema, capable of modeling the

different components of a fuzzy controller. Some samples of FML tag are [15]:

- `<FUZZYCONTROL>` that models a whole fuzzy system;
- `<KNOWLEDGEBASE>` and `<RULEBASE>` that represent the main components of a fuzzy system: the knowledge base and the rule base.

However, since an FML program realizes only a static view of a fuzzy system, the so-called eXtensible Stylesheet Language Translator (XSLT) is provided to change this static view to a computable version. Therefore, in short, FML is essentially composed by three layers [9]:

- XML in order to create a new markup language for fuzzy logic control;
- a XML Schema Document (XSD) as grammar to define the legal building blocks;
- a XSLT in order to convert a fuzzy controller description into a general-purpose computer language.

Differently from other similar approaches used to describe fuzzy controllers such as Fuzzy Control Language or MATLAB Fuzzy Inference System (FIS) developed by the Math Works, FML allows fuzzy designers to simply code their ideas on heterogeneous hardware by exploiting abstraction benefits offered by XML tools. Indeed, thanks to FML it is possible to design a fuzzy controller in human-readable and hardware independent way by avoiding additional work and development phases. The exploitation of FML is resulted in a new fuzzy control design philosophy named *transparent fuzzy controller* based on the integration among fuzzy systems, labeled tree theory, and XML methodology. Fig. 2 shows the tree structure of FML.

C. Taiwan Malware Analysis Net (TWMAN)

TWMAN is an open source (GPL v2) project based on a real OS environment such as the Joe Stewart's Truman [1, 3, 16]. TWMAN is an automated behavioral malware analysis environment to analyze the malware and a client-server architecture. The server works on Linux, while the client is Microsoft Windows, which use PXE-Boot in a real environment. In a Windows environment, malware usually attempts to execute or to install other applications in the Windows. It also includes the WinSock-functions, which can be able to communicate with TCP/IP-networks, such as the Internet or an intranet system. Some important efforts are made to interpret the Winsock calls. It must detect HTTP, FTP, SMTP and IRC connections and extracts the important data (i.e. FTP and IRC login information). It is interesting to know what the malware can create or how it can modify files on the disk. Windows NT/2000/2003/XP stores configuration data in a registry. It is a central repository for data configuration which is stored in a hierarchical manner. Systems, users, applications, and hardware in Windows allow the registry to store their configuration especially when a registry run key has been modified by some programs to make itself auto-start. The Advanced Intrusion Detection Environment (AIDE) against the disk image after the malware is run and compare it to the clean image that was uploaded to the client before the process and

take note if files were added, modified, or deleted during the malware run.

In order to test and verify, we have completed analyze 4840 malwares and generated report by TWMAN (Please see Table I). Our approach can get correct malware behavioral information, which can be expanded to solve more complex, practical problems and exploited to malware behavioral by agents to infer additional information and provide advanced services to end users.

III. FUZZY ONTOLOGICAL MODEL FOR TWMAN

This section is devoted to integrate OWL and FML in order to design a semantic methodology for improving malware behavioral analysis. In particular, in this section, we will describe how to use fuzzy ontologies to model the knowledge related to malware behaviors and how to merge these fuzzy ontologies with FML in order to design the proposed intelligent system. The main tools used to achieve the proposed result are: OWL by Protégé and VisualFML.

Description Logic and Fuzzy set theory are well-known formalisms for representing complex knowledge related to different application domains. According to the FML, which can create a data-oriented markup language features, and combine fuzzy logic, then create a vague abstraction of the fuzzy controller. Fig. 3 shows structure of FML-based malware behavioral analysis, so we can see two parts, namely the left part of the Knowledge Base and the right half of the rule base, and Table III shows the part of example FML code.

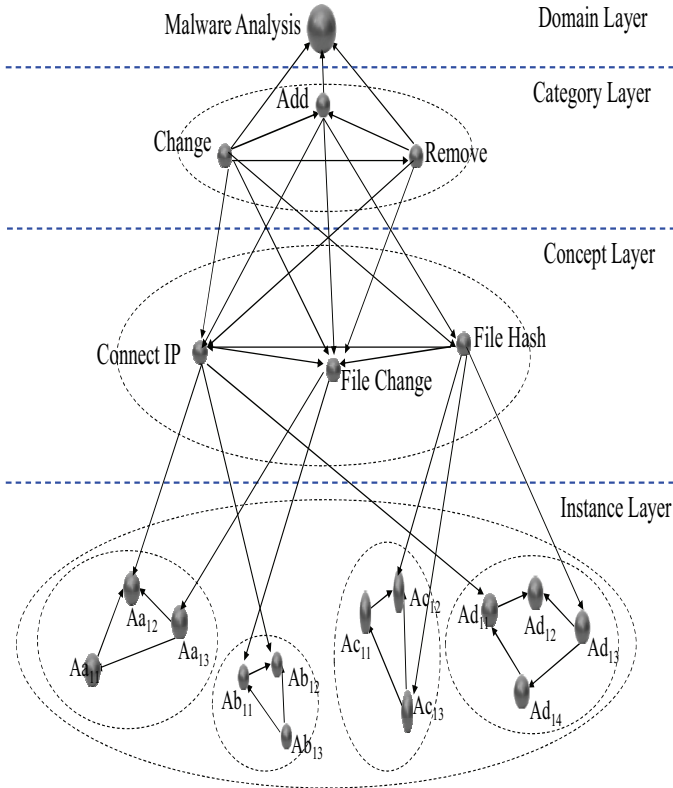


Figure 1. Structure of the malware behavioral ontology [1, 5].

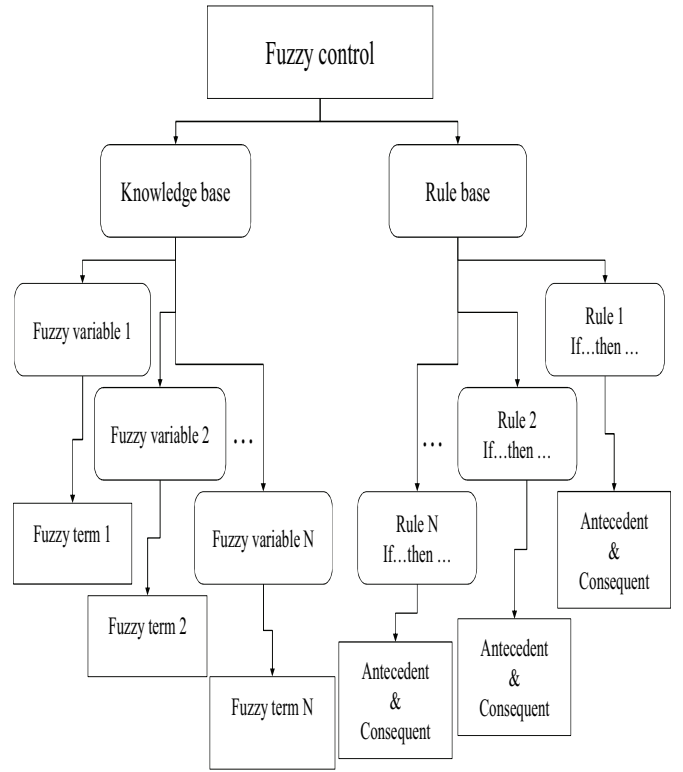


Figure 2. The tree structure of FML [10, 15].

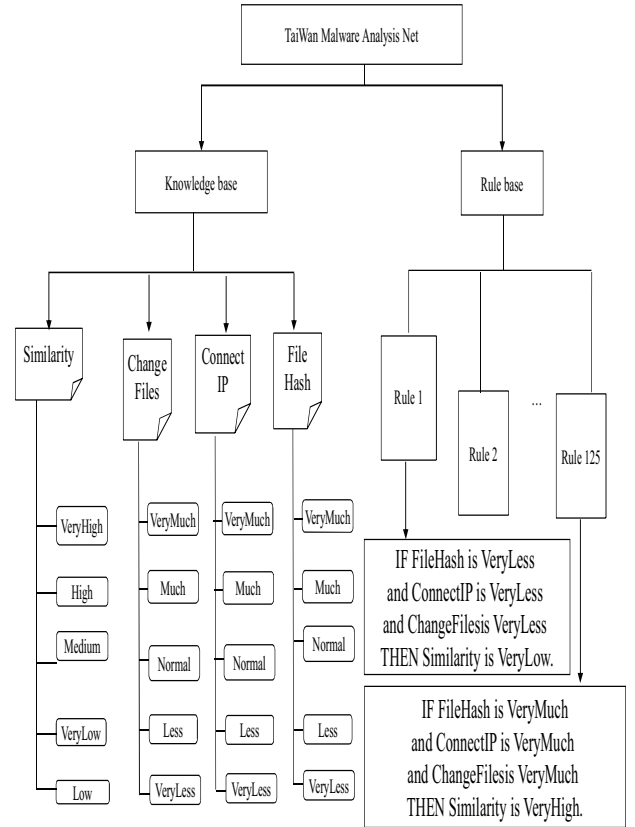


Figure 3. The structure of FML-based malware behavioral analysis.

TABLE I. EXAMPLE REPORT PART OF TWMAN [1, 3, 16].

```

Taiwan Malware Analysis Net, TWMAN - Analysis Report
>> Summary report for 7d99b0e9108065ad5700a899a1fe3441 created at Jan
12 14:06:37 CST 2011 <<
>> Host file changes <<
>> Registry Run Key changes<<
+++ /forensics/7d99b0e9108065ad5700a899a1fe3441/run-rip.txt 2010-07-
12 05:05:06.000000000 +0800
+Cryptographic Service\C\WINDOWS\system32\enjqtk.exe
>> Registry Service Key changes <<
+{33F3B709-064F-4FF7-95BD-434D50D67CCC}
+{DAACA8DA-DD27-4F5F-8163-1AE2BF76188D}
>> ssdeep info ( Fuzzy Hashing )<<
192:5E0HVFnXIKyFMOJd1BPSXPe2T7GCKE80XuPH:KwznVKyqCvPSry
iZu/"7d99b0e9108065ad5700a899a1fe3441"
7d99b0e9108065ad5700a899a1fe3441 matches
/forensics/ssdeep.db:96924b5583a0fa34a66d81257939a5d6 (65)
7d99b0e9108065ad5700a899a1fe3441 matches
matches /forensics/ssdeep.db:33d8d205001b83f96a9480ce51d54b60 (60)
>> Connect IP <<
IP 192.168.0.110 > 131.80.82.30: ICMP echo request, id 512, seq 5376,
IP 192.168.0.110 > 131.80.120.13: ICMP echo request, id 512, seq 56
IP 192.168.0.110 > 201.196.11.244: ICMP echo request, id 512, length 41
IP 192.168.0.110 > 201.196.171.67: ICMP echo request, id 512, length 41
IP 192.168.0.110.123 > 207.46.197.32.123: UDP, length 48
IP 192.168.0.110 > 201.196.207.12: ICMP echo request, id 512, length 41
IP 192.168.0.110 > 201.196.202.32: ICMP echo request, id 512, length 41
IP 192.168.0.110 > 201.196.13.108: ICMP echo request, id 512, length 41
IP 192.168.0.110 > 201.196.85.157: ICMP echo request, id 512, length 41
IP 192.168.0.110 > 201.196.108.43: ICMP echo request, id 512, length 41
IP 192.168.0.110 > 131.80.82.30: ICMP echo request, id 512, seq 5376,
IP 192.168.0.110 > 131.80.120.13: ICMP echo request, id 512, seq 5632,
IP 192.168.0.110 > 131.80.120.13: ICMP echo request, id 512, seq 5632,
IP 192.168.0.110 > 131.80.57.246: ICMP echo request, id 512, seq 5888,
IP 192.168.0.110 > 131.80.57.246: ICMP echo request, id 512, seq 5888,
IP 192.168.0.110 > 131.80.120.13: ICMP echo request, id 512, seq 56
IP 192.168.0.110 > 131.80.136.224: ICMP echo request, id 512, seq 6656,
IP 192.168.0.110 > 131.80.136.224: ICMP echo request, id 512, seq 6656
IP 192.168.0.110 > 201.196.85.157: ICMP echo request, id 512, length 41
IP 192.168.0.110 > 201.196.108.43: ICMP echo request, id 512, length 41
IP 192.168.0.110 > 131.80.82.30: ICMP echo request, id 512, seq 5376,
IP 192.168.0.110 > 131.80.120.13: ICMP echo request, id 512, seq 5632,
IP 192.168.0.110 > 131.80.120.13: ICMP echo request, id 512, seq 5632
>> NPAScan v1.7 <<
-----Start Scan-----
掃描完成!!未偵測到相關惡意程式!
-----End Scan-----
>> CWSandBox VirusScan Report <<
VSCAN Version:3.2.1861.2 (Feb 22 2009 19:30:04);run at: Jul 11 12:54:40
2010
defs version: 5444 (2009-10-12T17:47:12)
command line: c:\SBScanV3\vsan /l c:\virus.txt /def c:\SBDefsV3
C:\WINDOWS\system32\sandnet.exe
[ 53], Malicious ,CRC8_BL , Trojan.Win32.Generic!BT,
4150696,C:\WINDOWS\system32\sandnet.exe
1 objects processed in 0 secs, 0 fps
1 threats detected, 0 suspicious files

>> Advanced Intrusion Detection Environment-檔案異動偵測 <<
Summary:
Total number of files: 28292
Added files: 333
Removed files: 664
Changed files: 51
-----
Taiwan Malware Analysis Net, TWMAN - Analysis Report
Last Update 2011-01-01" E-Mail : TWMAN@nchc.narl.org.tw
Developed By TonTon | http://TWMAN.nchc.org.tw "

```

A. OWL and FML Model

There are several knowledge representation language, DLs and Fuzzy DLs have existed, e.g., Semantic Web Rule Language (SWRL), Java Environment Expert Systems (Jess), FUZZYDL and DELOREAN. Which provide a shared vocabulary of an application domain. Indeed, OWL can describe the relation between concept with instance and inference new instance, but the limitation of the first one must integrate SWRL language and Jess in Protégé 3, and the second one have to integrate Fuzzy OWL in Protégé 4. Table II shows the part of example OWL code. Therefore, we will introduce FML to model malware behavioral knowledge, are the following ones.

FML parser was used to extract the knowledge from the FML-based definition. The input is FML-based definition of the fuzzy controller which denoted by CFML. The output is composed of a set of $Fin = \{FileHashVeryLess, FileHashLess, FileHashNormal, FileHashMuch, FileHashVeryMuch, ConnectIPVeryLess, ConnectIPLess, ConnectIPNormal, ConnectIPMuch, ConnectIPVeryMuch, FileChangeLess, FileChangeVeryLess, FileChangeNormal, FileChangeMuch, FileChangeVeryMuch\}$, a set of $Fout = \{SimilarityVeryLow, SimilarityLow, SimilarityMedium, SimilarityHigh, SimilarityVeryHigh\}$ and fuzzy inference rules set $FS = \{Rule1, Rule2, \dots, Rule125\}$.

VisualFML Tool is a development environment for fuzzy-inference-based systems developed by Acampora and Loia [4]. The environment has been completely programmed in Java, so it can be executed on any platform with JRE (Java Runtime Environment) installed. Its functionalities cover the different stages of the fuzzy system design process, from their initial description to the final implementation [4]. It is possible to import a *.fml* file to build a fuzzy controller, and also to export a system created by the tool in a *.fml* file. The main functionalities the tool provides are the following ones: Fuzzy system description: define the system creating linguistic variables and rules; Verification: monitor the system behavior showing outputs corresponding to different inserted inputs; Tuning: apply algorithms based on neural network; Synthesis: generate high-level languages descriptions. Fig. 4 shows the fuzzy input variable editor in VisualFML and Fig. 5 shows the conceptual framework for the fuzzy controller.

B. Fuzzy Inference Mechanism

In order to design the our proposal of malware behavioral analysis tool, the main fuzzy components composing the system have to be defined. Fig. 6 show the membership functions of the fuzzy variables: File Hash (FH), Connect IP (CI), File Change (FC), and Similarity (SI) respectively. Fig. 7 shows the fuzzy variable control surface viewer. Table IV lists the parameters of the membership functions. Table V lists part of the fuzzy rules, which are constructed by the domain experts. Starting from this initial information, the proposed decision making system will be able to take opportune decisions about the malicious program detection. Next section is devoted to introduce a case study where our intelligent framework has been tested and validated.

TABLE II. THE PART OF EXAMPLE OWL CODE.

```

<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="file:D:/TWMAN/TWMAN.owl#"
  xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
  xml:base="file:D:/TWMAN/TWMAN.owl">
  <owl:Ontology rdf:about="file:D:/TWMAN/TWMAN.owl">
  <owl:imports
rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
  <owl:imports
rdf:resource="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl"/>
  ...
    </swrl:argument1>
    <swrl:propertyPredicate
rdf:resource="file:D:/TWMAN/TWMAN.owl#Has_Change"/>
    </swrl:IndividualPropertyAtom>
  </rdf:first>
  </swrl:AtomList>
  </rdf:rest>
  </swrl:AtomList>
  </rdf:rest>
  </swrl:AtomList>
  </swrl:body>
</swrl:Imp>
<MS_Win_XP rdf:ID="Win_XP-SP2"/>
<America rdf:ID="IP_199.7.51.190.80"/>
<System32 rdf:ID="tmp.edb"/>
<MS_Win_XP rdf:ID="Win_XP"/>
<Service_Key rdf:ID="xmlprov"/>
<MS_Win_Vista rdf:ID="Win_Vista"/>
<MS_Win_XP rdf:ID="Win_XP-SP1"/>
</rdf:RDF>
<!-- Created with Protege (with OWL Plugin 3.4.2, Build 562)
http://protege.stanford.edu -->

```

TABLE III. PARAMETERS OF THE MEMBERSHIP FUNCTIONS..

Symbol	Fuzzy Term	Membership Function
FH	Very Less	[0, 0, 851, 1276]
	Less	[851, 1276, 1702, 2128]
	Normal	[1702, 2128, 2553, 2979]
	Much	[2553, 2979, 3405, 3830]
	Very Much	[3405, 3830, 4682, 4682]
FC	Very Less	[0, 0, 1818, 2727]
	Less	[1818, 2727, 3636, 4545]
	Normal	[3636, 4545, 5454, 6363]
	Much	[5454, 6363, 7272, 8181]
	Very Much	[7272, 8181, 10000, 10000]
CI	Very Less	[0, 0, 1818, 2727]
	Less	[1818, 2727, 3636, 4545]
	Normal	[3636, 4545, 5454, 6363]
	Much	[5454, 6363, 7272, 8181]
	Very Much	[7272, 8181, 10000, 10000]
SI	Very Low	[0, 0.09, 0.18, 0.27]
	Low	[0.18, 0.27, 0.36, 0.45]
	Medium	[0.36, 0.45, 0.54, 0.63]
	High	[0.54, 0.63, 0.72, 0.81]
	Very High	[0.72, 0.81, 0.90, 1]

TABLE IV. THE PART OF EXAMPLE FML CODE.

```

<?xml version="1.0" encoding="UTF-8"?>
<FuzzyController name="TWMAN-3i5v-1o5v" ip="140.110.96.100">
  <KnowledgeBase>
    <FuzzyVariable name="ConnectIP" domainleft="0.0"
domainright="10000.0" scale="" type="input">
      <FuzzyTerm name="Less" complement="false">
        <TrapezoidShape Param1="1818.181818"
Param2="2727.272727" Param3="3636.363636"
Param4="4545.454545"/>
      </FuzzyTerm>
      <FuzzyTerm name="Much" complement="false">
        <TrapezoidShape Param1="5454.545455"
Param2="6363.636364" Param3="7272.727273"
Param4="8181.818182"/>
      </FuzzyTerm>
      <FuzzyTerm name="Normal" complement="false">
        <TrapezoidShape Param1="3636.363636"
Param2="4545.454545" Param3="5454.545455"
Param4="6363.636364"/>
      </FuzzyTerm>
      <FuzzyTerm name="VeryLess" complement="false">
        <TrapezoidShape Param1="0.0" Param2="0.0"
Param3="1818.181818" Param4="2727.272727"/>
      </FuzzyTerm>
      <FuzzyTerm name="VeryMuch" complement="false">
        <TrapezoidShape Param1="7272.727273"
Param2="8181.818182" Param3="10000.0" Param4="10000.0"/>
      </FuzzyTerm>
    </FuzzyVariable>
    <FuzzyVariable name="FileChange" domainleft="0.0"
domainright="10000.0" scale="" type="input">
      <FuzzyTerm name="Less" complement="false">
        <TrapezoidShape Param1="1818.181818"
Param2="2727.272727" Param3="3636.363636"
Param4="4545.454545"/>
      </FuzzyTerm>
      <FuzzyTerm name="Much" complement="false">
        <TrapezoidShape Param1="5454.545455"
Param2="6363.636364" Param3="7272.727273"
Param4="8181.818182"/>
      </FuzzyTerm>
    ...
  </Consequent>
</Rule>
<Rule name="RULE99" connector="and" operator="MIN"
weight="1.0">
  <Antecedent>
    <Clause>
      <Variable>FileHash</Variable>
      <Term>Less</Term>
    </Clause>
    <Clause>
      <Variable>FileChange</Variable>
      <Term>VeryLess</Term>
    </Clause>
  </Antecedent>
  <Consequent>
    <Clause>
      <Variable>Similarity</Variable>
      <Term>VeryLow</Term>
    </Clause>
  </Consequent>
</Rule>
</RuleBase>
</FuzzyController>

```

TABLE V. PART OF THE FUZZY RULES

Rule No	Fuzzy Variable			
	Input			Output
	FH	FC	CI	SI
1	Normal	Normal	Normal	Medium
2	Normal	Normal	Very Much	High
3	Normal	Normal	Much	Medium
4	Normal	Normal	Less	Low
5	Normal	Normal	Very Less	Very Low
6	Normal	Very Much	Normal	Medium
7	Normal	Very Much	Very Much	High
8	Normal	Very Much	Much	High
9	Normal	Very Much	Less	Medium
10	Normal	Very Much	Very Less	Medium
11	Normal	Much	Normal	Medium
12	Normal	Much	Very Much	High
13	Normal	Much	Much	High
...				
114	Very Less	Normal	Less	Low
115	Very Less	Normal	Very Less	Low
116	Very Less	Less	Very Much	Low
117	Very Less	Less	Much	Medium
118	Very Less	Less	Normal	Low
119	Very Less	Less	Less	Low
120	Very Less	Less	Very Less	Very Low
121	Very Less	Very Less	Very Much	Low
122	Very Less	Very Less	Much	Medium
123	Very Less	Very Less	Normal	Medium
124	Very Less	Very Less	Less	Low
125	Very Less	Very Less	Very Less	Very Low

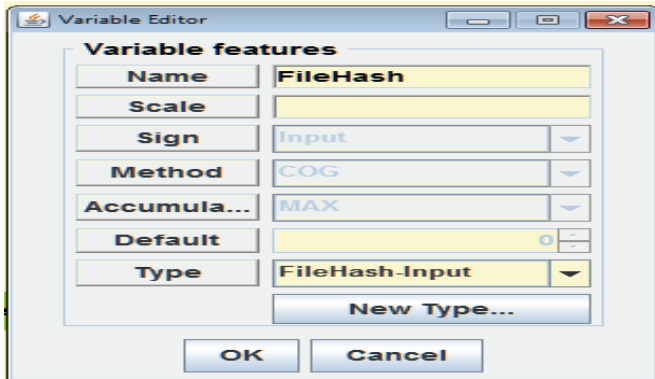


Figure 4. The fuzzy input variable editor in VisualFML .

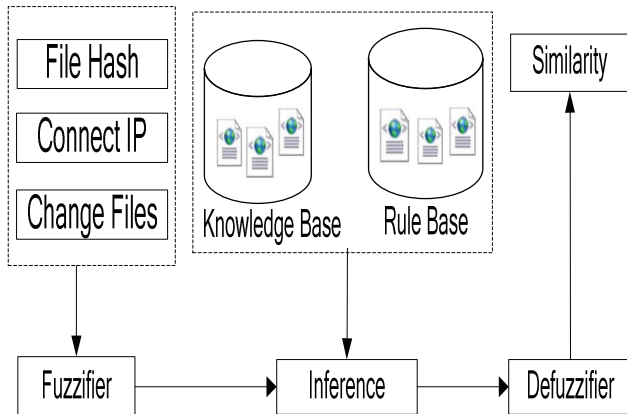


Figure 5. conceptual framework for the fuzzy controller

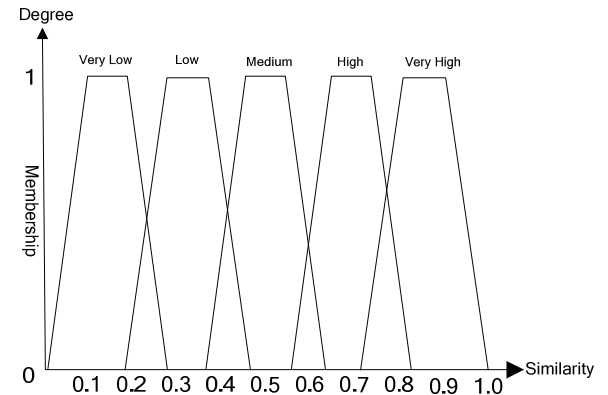
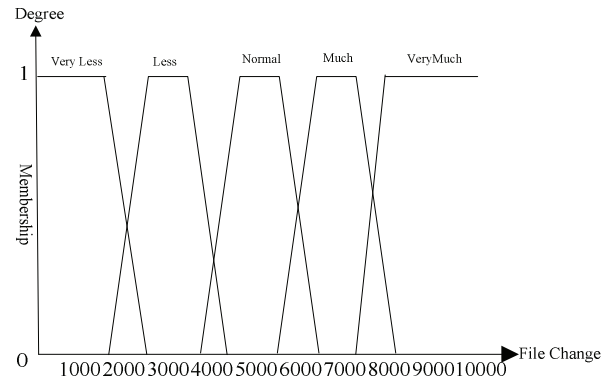
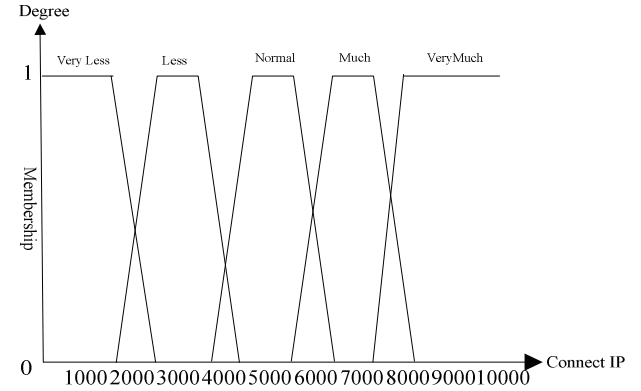
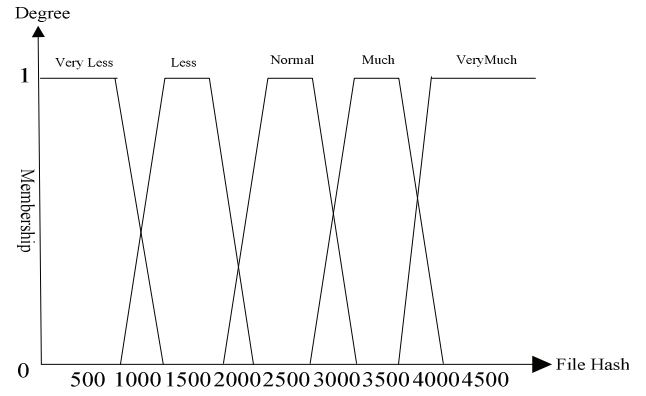


Figure 6. the membership functions of the fuzzy variables.

IV. EXPERIMENTAL RESULTS

In order to evaluate the effectiveness of the proposed approach, we adopt 30 attendance records retrieved from the malware repository of National Center for High Performance and Computing (NCHC) as the experimental data in Table VI. Fig. 7 shows a three-dimensional graphical representation of the system behavior and results of fuzzy inference process, which the values of the different output variables for a given set of input values.

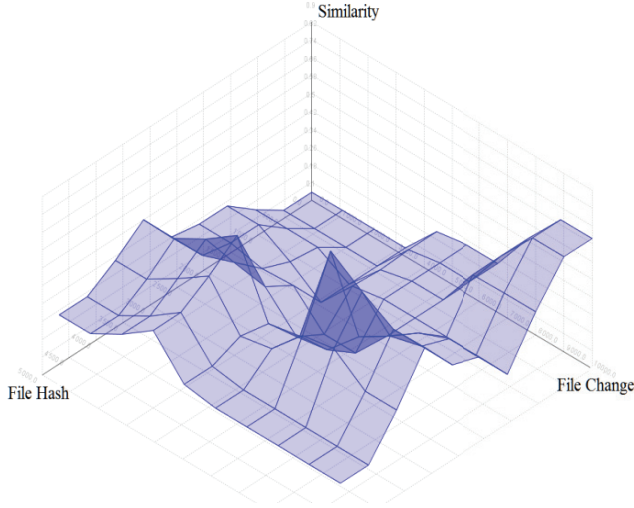


Figure 7. Fuzzy variable control surface viewer.

TABLE VI. EXPERIMENTAL DATA INPUT.

Unit	Input			Output	
	[0,4762]	[0,10000]	[0,10000]	[0,1]	
No	FH	CI	FC	SI	
				Desired	Inferred
01	40	0	998	0.977	0.136
02	0	8480	316	1	0.5
03	0	7332	194	0	0.5
04	0	10000	563	0.976	0.681
05	0	10000	2675	0	0.512
06	0	10000	2677	0	0.512
07	8	115	42	0	0.136
08	0	5726	553	0.951	0.375
09	12	10000	2586	0.951	0.5
10	6	10000	6198	0.951	0.681
11	0	10000	3375	0	0.5
12	195	0	1048	0.829	0.136
13	69	0	16	0.95	0.136
14	654	0	17	0.95	0.136
15	3522	7850	43	0.976	0.681
16	3140	7850	6450	0.976	0.681
17	3980	10000	10000	0.951	0.863
18	3540	10000	4357	0.951	0.727
19	2654	60	1423	0.976	0.364
20	3474	254	4750	0.95	0.318
21	941	10000	350	0.976	0.542
22	4112	10000	3456	1	0.863
23	50	930	132	0.976	0.136
24	359	10000	3246	0.976	0.5
25	298	339	6756	1	0.318
26	3728	0	2585	0.976	0.514
27	4	0	28	0.825	0.136
28	4613	947	6541	0.951	0.318
29	293	0	2831	0.951	0.136
30	4174	9543	7585	0.878	0.747

TABLE VII. VARIOUS PARAMETERS USED IN EQS. (1), (2) AND (3).

Actual results	Prediction results	
	YES	NO
YES	TP (true positive)	FN (false negative)
NO	FP (false positive)	TN (true negative)

$$Accuracy = (TN+TP)/(TP+TN+FP+FN) \times 100\% \quad (1)$$

$$Precision = TP/(TP+FP) \times 100\% \quad (2)$$

$$Recall = TP/(TP+FN) \times 100\% \quad (3)$$

The accuracy, precision and recall functions are shown by Eqs. (1), (2) and (3), respectively. The parameters of TP, FP, FN and TN are defined and listed in Table VII. The distribution of the accuracy, precision and recall are shown in Fig. 8.

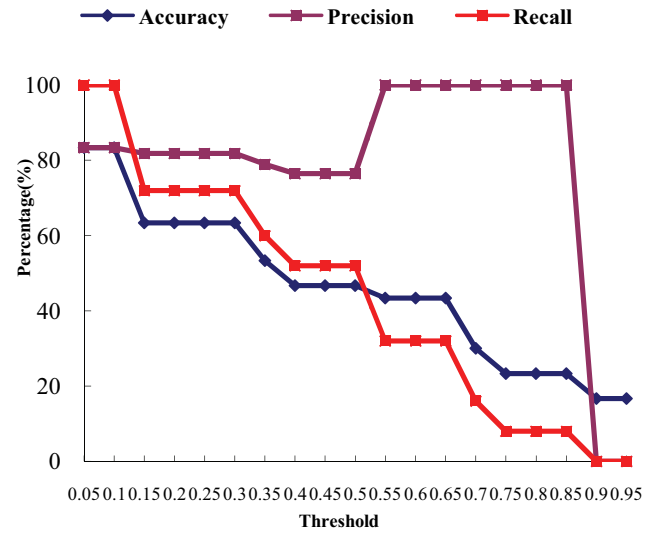


Figure 8. Results of the accuracy, precision and recall.

V. CONCLUSION

In this study, fuzzy ontology for malware behavioral analysis scheme based on FML has been presented. It is composed of a fuzzy inference and semantic decision-making mechanism to describe the changes of the registry key value, network connection and file to build the ontology and behavioral rules. The experimental results show that our approach will feasible and usable for malware behavioral analysis system. It can utilize them to determine the behaviors of virus or malicious program to provide better protection and services to end users.

In the future, there are more vague knowledge of OS platform with experimental data will be collected, and type-1 FML is not enough to deal with. Therefore, some upgradable technologies, like genetic algorithm and type-2 fuzzy set, will be update to be exercised for comparing with the current technology. We also compare the performance with other classification methods. In order to take prompt and appropriate measures, TWMAN can integrate with a human thinking semantic model. Therefore, we consider that TWMAN could be a powerful tool to improve the network safety.

ACKNOWLEDGMENT

The authors would like to thank the National Center for High Performance Center of Taiwan for financially supporting this open source research project. In addition, the authors would like to thank National Science Council of Taiwan for financially supporting this research under the Grant NSC 99-2923-E-024-003-MY3, NSC 98-2221-E-024-009-MY3, NSC 99-2911-I-024-004 and NSC 99-2218-E-492-006 .

REFERENCES

- [1] H. D. Huang, T. Y. Chuang, Y. L. Tsai, and C. S. Lee, "Ontology-based Intelligent System for Malware Behavioral Analysis," in *Proceeding of the 2010 IEEE World Congress on Computational Intelligence (WCCI2010)*, Barcelona, Spain, Jul. 18-23, 2010, pp. 1-6.
- [2] H. Zhao, M. Xu, N. Zheng, J. Yao, and Q. Ho, "Malicious Executables Classification Based on Behavioral Factor Analysis," *2010 International Conference on e-Education, e-Business, e-Management and e-Learning*, Sanya, China, 2010.
- [3] H. D. Huang, C. S. Lee, H. Y. Kao, Y. L. Tsai, and J. G. Chang, "Malware Behavioral Analysis System: TWMAN," in *Proceeding of the 2011 IEEE Symposium on Computational Intelligence for Intelligent Agent (IEEE SSCI 2011)*, Paris, France, Apr. 11-15, 2011, pp. in press.
- [4] C. S. Lee, Z. W. Jian, and L. K. Huang, "A fuzzy ontology and its application to news summarization," *IEEE Transactions on Systems, Man and Cybernetics Part B*, vol. 35, no. 5, pp. 859-880, 2005.
- [5] M. H. Wang, C. S. Lee, K. L. Hsieh, C. Y. Hsu, G. Acampora, and C. C. Chang, "Ontology-based multi-agents for intelligent healthcare applications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, no. 2, pp. 111-131, Jun. 2010.
- [6] F. G. Sanchez, R. M. Bejar, L. Contreras, J. T. F. Breis, and D. C. Nieves, "An ontology-based intelligent system for recruitment," *Expert Systems with Applications*, vol. 31, no. 2, pp. 248-263, Aug. 2006.
- [7] T. T. Quan, S. C. Hui, and A. C. M. Fong, "Automatic fuzzy ontology generation for semantic help-desk support," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 3, pp. 155-164, Aug. 2006.
- [8] T. T. Quan, S. C. Hui, A. C. M. Fong, and T. H. Cao, "Automatic fuzzy ontology generation for semantic web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 6, pp. 842 - 856, Jun. 2006.
- [9] S. H. Ho, C. L. Yang, C. Y. Chen, C. Y. Hsu, and Y. K. Chang, "An Intelligent-Mamdani Inference Scheme for Healthcare Applications Based on Fuzzy Markup Language," in *Proceeding of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN2009)*, Kaohsiung, Taiwan, Dec. 14-16, 2009, pp. 400-404.
- [10] G. Acampora and V. Loia, "Using fml and fuzzy technology in ambient intelligent environments," *International Journal of Computational Intelligence Research*, vol. 1, no. 2, pp. 171-182, 2005.
- [11] F. Bobillo and U. Straccia, "Representing fuzzy ontologies in OWL 2," in *Proceeding of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010)*, 18-23 July, 2010, pp. 1-6.
- [12] J. Kornblum. (2010). *ssdeep*. Available: <http://ssdeep.sourceforge.net>
- [13] T. Hungenberg and M. Eckert. (2010). *INetSim*. Available: <http://www.inetsim.org>
- [14] R. Lehti and P. Virolainen. (2011). *AIDE*. Available: <http://aide.sourceforge.net>
- [15] G. Acampora and V. Loia, "Fuzzy control interoperability and scalability for adaptive domotic framework," *IEEE Transactions Industrial Informatics*, vol. 1, no. 2, pp. 97-111, May 2005.
- [16] J. Stewart, "Behavioural malware analysis using Sandnets," *Computer Fraud & Security*, vol. 2006, no. 12, pp. 4-6, Dec. 2006.