IJICTR International Journal of Information & Communication Technology Research ITRC

Volume 9-Number 3-Summer 2017(17-24)

Domain Ontology to Distinguish Different Types of Rootkits

Ahmad Salahi
(Corresponding Author)
Information Security Department
Research Institute for ICT
Tehran, Iran
salahi@itrc.ac.ir

Javad Enayatizadeh
Information Security Department
Research Institute for ICT
Tehran, Iran
j_enayati@comp.iust.ac.ir

Received: January 11, 2017 - Accepted: May 19, 2017

Abstract— **Rootkit is an auxiliary tool for sniffing, stealing and hiding, so it has become the key component in almost all successful attacks. Analysis of rootkits will provide system administrators and security software managers the ability to detect and prevent a computer being compromised. Ontology will provide detailed conceptualization to represent the rootkit concepts and its relationships to other security concepts in cyber-attack domain. In this paper we presented an ontology for rootkits which contains many concepts relating to security, cyber-attacks and operating systems. We divided rootkits according to four attributes, and expanded the ontology for rootkits accordingly. This ontology can be used to distinguish different types of rootkits**

Keywords: Ontology, Rootkit, Malware, Security

I. Introduction

Harm caused by malware is a serious problem in information system domain. Although there are a lot of security software to detect malware, but they can't guarantee a perfect detection and removal of malware. Malware authors make use of extremely sophisticated hiding techniques to prevent malware being detected According to [3], Malware is a malicious code that has potential to harm any machine which executes it or the network over which the machine communicates. Malwares include virus, worm, botnet, spyware, backdoor, Trojan horse, rootkit and exploits [4]. Today malware is used to steal business, financial and sensitive personal information for the benefit of others. We focus on rootkit, because once a malicious program is installed on a system, it is essential to remain hidden to avoid detection and be hidden from the user. The term rootkit, in the field of computer security is used to define a set of programs which are used by a cracker to conceal his/her activities on a compromised computer and make it possible to return undetected in future.

If we consider a rootkit as a “Trojan Horse” and according to [1], it can be divided into four categories, *Direct masquerades* **(**pretend to be normal programs**)**, *Simple masquerades***(**do not masquerade as existing programs, but masquerade as possible programs**)**, *Slip masquerades* (have names approximating existing names), *Environmental masquerades***(**already-running programs that not obvious for the user**)**. The rootkit can be direct masquerade and environmental masquerade because it tries to hide its existence on an infected computer by modifying program binaries or legitimate code and hooking call tables such as the Interrupt Descriptor Table (IDT) and the System Service Descriptor Table (SSDT) to hijacking the kernel's control flow [5]. We should notice that a cracker must already have root access before installing a rootkit because a rootkit just makes it easier for a cracker to gain root level access some other time.

The term ontology can be explained in many different ways, for our research activities ontology defines basic terms and relations compromising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary [2]. Therefore our ontology includes concepts, concept taxonomies, relationships between concepts, and properties that describe concepts.

As mentioned earlier, the malwares are important in information security. There are different malwares that cause different malwares by different concepts and relationships between them.

The detection of malwares is one of the security challenges. There are several methods for detection of these malwares that use signatures and heuristics.

The malwares can be combined that makes detection of malwares more complex.

Several knowledge representations are proposed for malwares that are based on taxonomies that almost all of them don’t support optimal attack detection for complex malwares.

"Ontology is a formal, explicit specification of a shared conceptualization that is characterized by high semantic expressiveness required for increased complexity."[27]

Ontology is a technology that can create objects, concepts and relation between them. Ontology is used for detection and prevention of different malwares it provides a knowledge presentation for malwares that produce a reasoning framework**.**

Protégé [26] is an open source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontology.

In this paper we use Protégé to create ontology for root kits that are used for detection we propose an ontology base behavior analysis for rootkits. We provide information about rootkits.

II. Related works

Malwares are serious problem for the security of networks that led to widespread investigations of malwares. The detection of malwares was done by different antivirus software.

These anti-viruses use signatures based methods that describe the probabilities of specific malicious behaviors. These signatures are static and can be obtained from using behavior of malwares with experts of information technology. Unfortunately, with a small change of malware it will be not be detectable by the same signature. Thus, this static software cannot be used for unknown malwares [14-15].

Today, there are a lot of zero day malwares that sniff, steal and change the information. These malwares cannot be detected by antiviruses.

In recent years, several works for the behavior detection of zero days’ malwares have been done. These works study the behavior of different malwares.

Rootkit is a kind of malware that uses stealth methods to hide itself from being discovered by system administrators. E. Lacombe and F. Raynal [15] define rootkits as “a set of modifications that allow an attacker to maintain along the time a fraudulent control of the information system". First Rootkits were introduced at the end of 80’s. Rootkits are very hard to detect by usual anti-viruses. Jianxiong Wang introduces a rule-based approach for the rootkit detection because a rootkit can change some data structures of a system by hiding itself[16].Woei-Jiunn Tsaur and Yuh-Chen surveyed the weaknesses of current detectors, and also discussed possible remedies and solution for detecting the proposed subtle rootkits[17]. Shu Zhou and Chenglong Cao suggested a rootkit detection mechanism based on the hidden registry information, and designed a Windows rootkit detection method based on cross-view [18]. Endong Wang, Long Xin, Zhongyuan Wu, Weiqing Dong and Xiaoshe Dong proposed a method of Root kit detection based on KVM (Kernel-based Virtual Machine) by using virtualization technology [19]. Hai Bi suggested a method of integrity detection and restoration based on kernel file, which is proved to ensure correct implementation of the kernel function [20]. Watters, P. and Xinwen Wu proposed a new rootkit classification system and tested their system on a sample of rootkits that use inline function hooking [21].

Yu-Jie Hao, Yan Zhang, Zhi-Peng Lu and Rui Zhang, according to the analysis of hiding technology of malicious programs proposed a new idea of detecting malware based on the raw data [22].

The ontology is a new theory in network security that can be used to detect relation between different attacks. Andrew Simmons [6] has defined ontology for network security attacks and reviewed threats, vulnerabilities and failure modes. Kim, Luo and Kang [7] introduced ontology that describes type of security information including algorithms, protocols, mechanisms, objectives and credentials. John D. Howard [9] designed a common language that includes terms and taxonomies for gathering, exchanging and comparing different computer security incidents. Denker, Nguyen, and Ton [8] express security related information for all types of resources. Hsiu-Sen Chiang, Woei-Jiunn Tsaur [10] proposed ontology about mobile malware behavior for organizations and end users to increase their knowledge about mobile malware. Tala Tafazzoli and Seyed Hadi Sadjadi [4] used fuzzy logic to present relationship between concepts of malware.

Jun Han described WS security threats and stated that they have to be analysed and classified systematically in order to allow the development of better distributed defensive mechanisms for WS using F/IDS [22]. Modern rootkits do not elevate access, but rather are used to make another software payload undetectable by adding stealth capabilities.

In the next section, we describe the proposed ontology for rootkit.

III. Key concepts of proposed rootkit ontology

In this section, by using ontology, we classify different rootkits and discuss different structures of rootkits.

A. *Rootkit types*

In most information systems there are mechanisms to protect data and functionality from malicious behavior and fault. Operating systems provide different level of access to resources. These levels are called rings. The inner most ring is called Ring 0 and this level is the most protected which interacts most directly with the physical hardware such as the CPU and memory. Linux and windows only use two rings, kernel level and user level. According to this categorization as shown in figure [1] there are generally two main types of rootkits: user-mode and kernel-mode. User-mode rootkits run within the environment and security context of a user on the system and kernel-mode rootkits operate within the operating system at the same level as drivers for hardware. There are some different types of rootkits such as:

Hybrid rootkit: It combines the easiness characteristics of the user-mode and stability characteristics of the kernel-mode. This allows a rootkit which has access to all procedures that have access to the user-mode and all data structures in the kernel-mode. FU is a hybrid rootkit which has components operating in the kernel mode and the user mode and utilizes Direct Kernel Object Manipulation (DKOM) to hide processes, device drivers, and ports and alter process properties. The FU rootkit can hide processes, elevate process privileges, fake out the Windows Event Viewer so that forensics is impossible, and even hide device drivers. It does all this by Direct Kernel Object Manipulation. (25)

Firmware rootkit: Uses platform firmware or devices to create a persistent malware image in hardware, such as the system bios, a network card or hard drive. Therefore the rootkit can hide itself in firmware and reinstall itself when the computer restarts. The most interesting feature is that even if security software identifies and removes it; it can install itself again, when the computer is switched on. For example in March 2009, researchers Alfredo Ortega and Anibal Sacco published details of a BIOS-level Windows rootkit that was able to survive disk replacement and operating system re-installation. [24]

Virtual rootkit: The early works of Goldberg and Popek have defined some of the hardware requirements to be able to run a hypervisor, i.e. the software that controls different physical systems and virtual machines. The capability to host a hypervisor, also known as virtual machine monitors (VMM). This kind of rootkit is almost invisible and prevents being detected by security software through hiding rootkit software in virtual machine environments. There are two rootkit architectures based on virtual machines, namely full virtualization and partial virtualization.

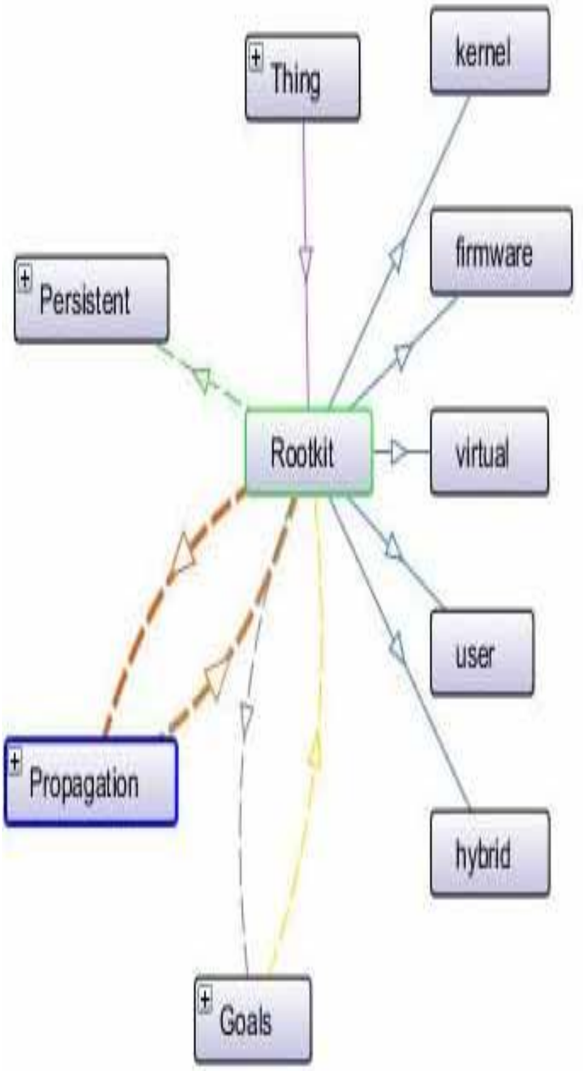

Fig. 1. Different types of rootkits.

B. *Persistent rootkits*

There are two types of rootkits: hard resident and memory resident.
Hard resident: In order to remain in host after a reboot, a rootkit must physically alter the data of the hard drive to automatically start itself up. For example by adding auto start entry to the registry, it can be loaded into memory and executed automatically.

Memory resident: It just exists in memory and is not capable of automatically running again after the system has been restarted. Therefore it makes rootkits a lot harder to detect because they have no physical trace of their existence on the hard drive.

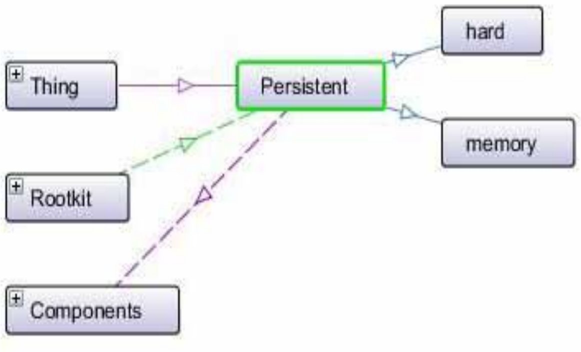

Fig. 2. Different types of persistent rootkits

C. Propagation

Kernel-level access is usually installed by a dropper component that may come to the system from different sources. Rootkits usually employ attacks against platforms and applications such as Microsoft Windows, Linux and Mac OS. We should notice that rootkits can't propagate by themselves. Indeed, rootkits are just one component of a blended threat that consists of a dropper, loader and rootkit. The dropper is the code that gets the rootkit's installation to start and the loader loads the rootkit into memory. In the following sections, we discuss how rootkits can reach to a victim system. Fig [3] shows the relation between propagation and other objects in this ontology.

Social engineering: The oldest and most effective method for propagation of rootkits across a network is trust relationship. Social engineering is a term that describes a non-technical kind of intrusion that relies on human interaction to trick people to break computer security procedures. Crackers use this technique thorough email attachments, website, peer-to-peer network and phishing to install a rootkit on victim systems.

File Execution: This is the most straightforward method for rootkit infection. Today, crackers compromised systems through social engineering techniques to make users click an infected file that maybe renames or embedded within another file, such as Microsoft Office Documents, PDFs, Zips and other popular file types.

DLL Injection: DLL injection refers to a method for attackers to manipulate programs and processes to execute another program. DLL injection provides a manner for attributing the malicious .dll to running processes. Processes are tasks that are being handled by the operating system. DLLs are Dynamic Link Libraries, i.e. they are shared code that may be executed by a running process. There are two kinds of injection: static and dynamic injection. Static injection occurs prior to program execution. Dynamic injection occurs when processes are loaded into memory. It provides a way to piggy back the malicious code onto a process. This gives attacker two advantages: secrecy and trust. By DLL injection, trusted applications can be exploited. Rootkits use DLL injection to inject code into a process that has some privileges.

D. Goals

A rootkit is designed to enable continued privileged access to computer and hide its process and programs from normal methods of detection. Therefore we divided these goals into two categories: Data theft and Concealment. Fig [4] shows all aspects of goals.

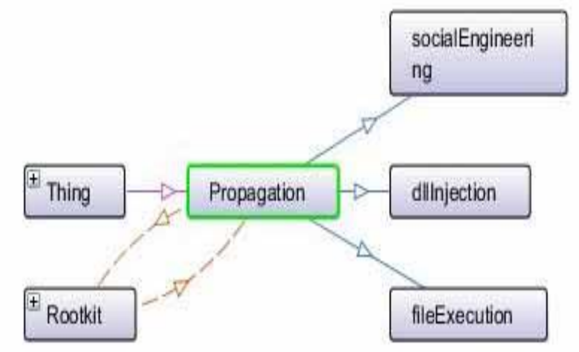

Fig. 3. Different types of rootkit propagation

Data theft: A rootkit is used to steal information from a host such as identity, financial information and click fraud. Keylogger is immensely used in order to steal information and broadcast recorded data from the host. Software keyloggers capture keystrokes by running procedures and can be further categorized into three types: kernel based, hook based, and user space method.

1-Kernel based: This type of keylogger is at the kernel level and receives data directly from the input device (typically, a keyboard). Codes are written in the kernel to directly intercept key events from hardware. It can be programmed to be virtually undetectable by taking advantage of the fact that it is executed on boot, before any user-level applications start. Since the program runs at the kernel level, one disadvantage to this approach is that it fails to capture auto complete passwords, as this information is passed to the application layer

2-Hook based: The program has access to kernel calls and captures keystrokes by subscribing to keyboard events detected by OS. This type of logging is accomplished by using the Windows function SetWindowsHookEx() that monitors all keystrokes. The spyware will typically come packaged as an executable file that initiates the hook function, plus a DLL file to handle the logging functions. An application that calls SetWindowsHookEx() is capable of capturing even autocomplete passwords. It is impossible for Anti-Virus software to remove kernel-based and hook-based keyloggers because they reside in/close to kernel and enjoy direct access to keyboard resources.

3-User space: This logs key by calling system API that allows checking key state. GetAsyncKeyState is an example of such API. In MSDN, GetAsyncKeyState function determines whether a key is up or down at the time the function is called, and whether the key was pressed after a previous call to GetAsyncKeyState.

Concealment: Concealment is the most important part of a rootkit such as concealment of process injection, device driver's entries, registry key, file, communication link and process. Rootkit uses hooking or modify some system binary files to conceal its activity.

There are two types of hooks, (1) API hooking and (2) System tables hooking.

(1) API hooking: the term hooking covers a range of techniques used to alter or augment the behavior of an operating system, of applications, or of other software components by intercepting function calls or messages or events passed between software components. Programs in user-mode communicate with kernel through an application programming interface (API). Most rootkits modify the address of APIs in the important address table (IAT) of user process in order to make sure the operating system returns only filtered results. For example, it may hook the APIs that are used by Windows Explorer to display files and folders or the APIs that Task Manager uses to shows its list of active processes.

(2) System Tables hooking: Kernel mode rootkits involve system hooking or modification in kernel space. Kernel space is generally off-limits to standard authorized (or unauthorized) users. One must have the appropriate rights in order to view or modify kernel memory. The kernel is an ideal place for system hooking because it is at the lowest level and thus, is the most reliable and robust method of system hooking. The system call path through the kernel passes through a variety of hook points.

As a system call's execution path leaves user mode and enters kernel mode, it must pass through a gate. The purpose of the gate is to ensure user mode code does not have access to kernel mode space, protecting the kernel space. This gate must be able to recognize the purpose of the incoming system call and initiate the execution of code inside the kernel space and then return results back to the incoming user mode system call. The gate is effectively a proxy between user mode and kernel mode.

A popular hook point is to modify the System Service Descriptor Table (SSDT) which is a function pointer table in kernel memory that holds all the addresses of the system call functions in kernel memory. A system call is a function supplied straight by the kernel and usable by all user-mode processes. For example by modifying this table, the rootkit can redirect execution to its code instead of the original system call.

Some rootkits may modify system binary files to change their functionality. For example, rootkit changes ps utility (short for "process status") which displays the current process running on a system to hide the attacker's activity from the system administrator.

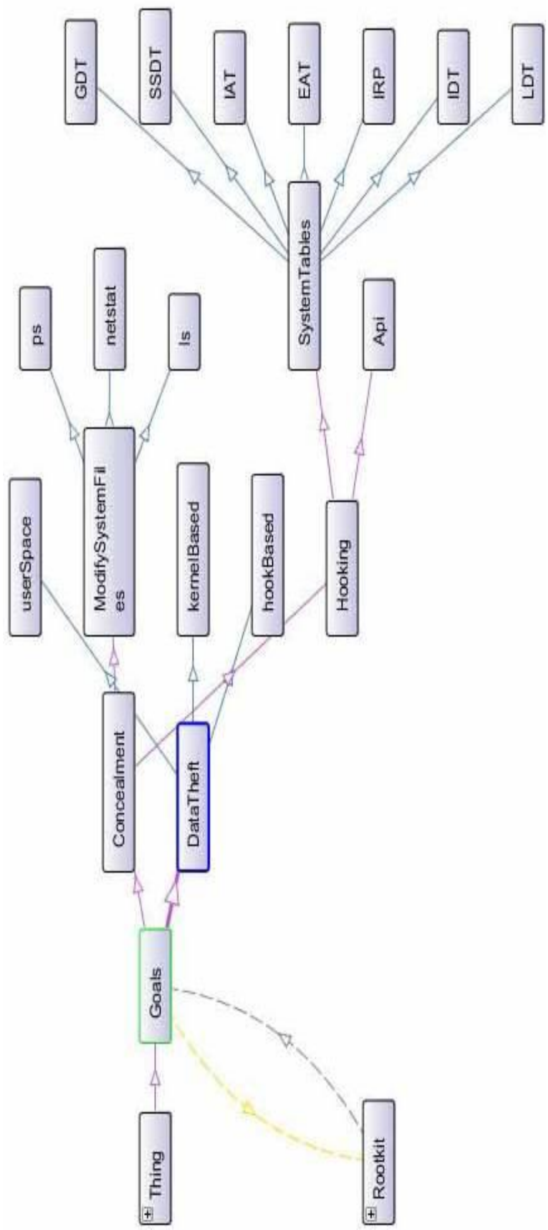

Fig. 4. Rootkit's goals

IV. FAMOUSE ROOTKITS

In this section, we discuss some new famous rootkits introduced in recent years and describe them according to proposed ontology.

Stuxnet: is a computer worm discovered in June 2010, this worm can steal code and design projects and also hide itself using a classic Windows rootkit. Stuxnet has the ability to take advantage of the programming software and also upload its own code to the PLC (Programmable logic controller) in an industrial control system that is typically monitored by SCADA systems (Supervisory control and data acquisition). In addition, Stuxnet then hides these code blocks, so when a programmer using an infected machine tries to view all of the code blocks on a PLC, they will not see the code injected by Stuxnet. Stuxnet hooks the programming software, which means that when someone uses the software to view code blocks on the PLC, the injected blocks are nowhere to be found.

Duqu: is a computer worm discovered on 1, September, 2011. It is built on relatively old technology but infections can lead to confidential information theft, loss of intellectual property and other risks associated with the presence of a keylogger. Duqu rootkit protects a keylogger component that gathers information from the infected computers.

Flame: is a modular computer malware discovered in 2012. Flame can spread to other systems over a local network (LAN) or via USB stick. It can record audio, screenshots, keyboard activity and network traffic. The malware determines what antivirus software is installed, then customizes its own behavior to reduce the probability of detection by that software [11]. Additional indicators of compromise include mutex (mutex is a synchronization mechanism for enforcing limits on access to a resource in an environment where there are many threads of execution) and registry activity, such as installation of a fake audio driver which the malware uses to maintain persistence on the compromised system [12].

V. EVALUTION OF THE ONTOLOGY

In this paper, we use OntoQA [28], an approach that analyzes ontology schemas and their populations and describes them through a well-defined set of metrics. OntoQA, is a tool that evaluates ontologies related to a certain set of terms and then ranks them according a set of metrics that captures different aspects of ontologies. Since there are no global criteria defining how a good ontology should be, OntoQA allows users to tune the ranking towards certain features of ontologies to suit the need of their applications. We use OntoQA to evaluate the quality of proposed ontology on the different dimensions mentioned in OntoQA.

The OntoQA framework is one of the metric based approaches as well. OntoQA defines the quality of a populated ontology as a set of five schema quality features and nine knowledgebase (or instance-base) quality features.

The quality of ontology classified to two groups: schema and knowledgebase. The first category evaluates ontology design and knowledge presentation and the second category evaluates instance data within the ontology and the effective utilization of the knowledge modeled in the schema [5]. In this section, we describe the different metrics that can be used in two groups.

A. Schema Metrics

Schema Metrics describe the design of the proposed ontology. These metrics are not used to correct the proposed ontology. It can be used to measure the richness, width, depth, and inheritance of an ontology schema design.

There are three important metrics in schema metrics.

A.1 Relationship Richness

The relationship richness shows variant kinds of relations in the ontology. The ontology with more types of sets of relationship has more information in comparison to inheritance relationships.

The relationship richness is shown as the percentage of the non-inheritance relationships (P) between classes compared to all of the possible connections that can include inheritance and non-inheritance relationships (H).

$$RR = \frac{|P|}{|P| + |H|}$$

A.2 Inheritance Richness

The inheritance richness of the schema (IR)is defined as the average number of subclasses per class.

$$IR = \frac{|H|}{|C|}$$

A.3 Attribute Richness

The number of attributes that are defined for each class can indicate both the quality of ontology design and the amount of information pertaining to instance data. The attribute richness (AR) is defined as the average number of attributes (slots) per class. It is computed as the number of attributes for all classes (att) divided by the number of classes (C).

$$AR = \frac{|att|}{|C|}$$

B. Knowledgebase Metrics

The way data is placed within ontology is also a very important measure of ontology quality because it can indicate the effectiveness of the ontology design and the amount of real-world knowledge represented by the ontology. Instance metrics include metrics that describe the KB (Knowledgebase) as a whole, and metrics that describe the way each schema class is being utilized in the KB.

The results of metrics calculation for ontology of rootkit shown in Fig.5 are given in Table 1.

Table 1. Different metrics and their values for rootkit ontology

Metric	Value
Relationship Richness (RR)	0.33
Inheritance Richness (IR)	0.9
Attribute Richness (AR)	0.5
Axioms (Triples)	20
Concepts	12
Object Properties	6
Data Properties	2

VI. CONCLUSION

Rootkits are dangerous malwares that can steal, modify and sniff the information of a system without knowledge of administrator.

In this paper, we have presented ontology for rootkit which shows the relationship between diverse concepts, with the conceptualization drawn in figure 5. The next step, after getting feedback and refining this proposal, we are going to customize our rootkit ontology which can detect user-mode and kernel-mode rootkits.

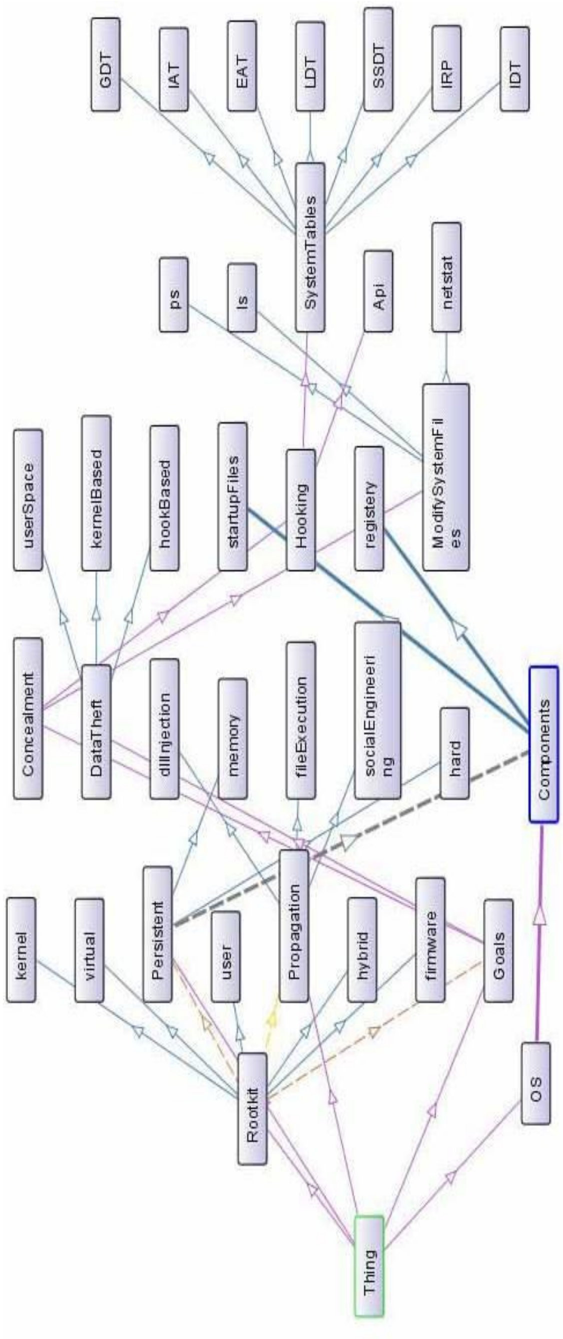

Fig. 5. Ontology of rootkit

REFERENCES

[1] H. Thimbleby, S. Anderson, P. Cairns, “A Framework for Modeling Trojans and Computer Virus Infections,” The Computer Journal, vol. 41, no.7 pp. 444-458, 1998.

[2] A. G´ omez-P´ erez, M. Fern´ andez-L´ opez, and O. Corcho, ntological Engineering, 1st ed. London: Springer, 2004.

[3] Gruber, T., Towards Principles for the Design of Ontologies used for Knowledge Sharing. International Journal of Human -Computer Studies, 1995. 43(5/6): p.907 -928.

[4] Tala Tafazzoli and Seyed Hadi Sadjadi. Malware fuzzy ontology for semantic web. IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.7, July 2008.

[5] Manuel Corregedor and Sebastiaan Von Solms, Implementing Rootkits to Address Operating System Vulnerabilities

[6] Andrew Simmonds, Peter Sandilands, and Louis van Ekert, An ontology for network security attacks, RAID 2003, LCNS 2820, Springer-Verlag,2003.

[7] Kim, A, Luo, J & Kang, M 2005 ‘Security Ontology for Annotating Resources’, paper presented to the 4th International Conference on Ontologies, Databases, and Applications of Semantics, ODBASE 2005.

[8] Denker, G, Nguyen, S & Ton, A 2004 ‘OWL-S Semantics of Security Web Services: a Case Study’, paper presented to SRI International, Menlo Park, California, USA.

[9] John D. Howard, Thomas A. Longstaff, A common language for computer security incidents, Sandia National Laboratories, Sandia Report, 1998.

[10] Hsiu-Sen Chiang, Woei-Jiunn Tsaur, Ontology-based Mobile Malware Behavioral Analysis

[11] A Complex Malware for Targeted Attacks". Budapest University of Technology and Economics. 28 May 2012. Archived from the original on 30 May 2012. Retrieved 29 May 2012.

[12] Flamer/sKyWIper Malware: Analysis. FireEye. Archived from the original on 31 May 2012. Retrieved 31 May 2012.

[13] [12] M. Christodorescu, S. Jha, S.A. Seshia, D. Song, and R.E. Bryant,“Semantics-aware malware detection,” in Proc. the IEEE Symposium on Security and Privacy, Oakland, California, pp. 32-46, May 2005.

[14] [13] J. A. Morales, P. J. Clarke, Y. Deng, and B. M. Golam Kibria,“Testing and evaluating virus detectors for handheld devices”,Journal in Computer Virology, vol. 2, no. 2, pp. 135-147, 2006.

[15] E. Lacombe, F. Raynal, V. Nicomette, Rootkit modeling and experiments under Linux, Journal in Computer Virology, vol. 4, no. 2, 2008, pp:137-157.

[16] Jianxiong Wang,” A Rule-based Approach for Rootkit Detection”,The 2nd IEEE International Conference on Information Management and Engineering (ICIME), Pp**.** 405 – 408,2010.

[17] Detectors' Vulnerabilities Using a New Woei-Jiunn Tsaur and Yuh-Chen,” Exploring Rootkit Windows Hidden Driver Based Rootkit”, IEEE Second International Conference on Social Computing (SocialCom), pp.842-848,2010.

[18] Shu Zhou and Chenglong,” A Windows Rootkit Detection Method Based on Cross-View”, *International Conference on E-Product E-Service and E-Entertainment (ICEEE)*,pp.1-3,2010.

[19] Endong Wang, Long Xin, Zhongyuan Wu, Weiqing Dong and Xiaoshe Dong,” KVM-based Detection of Rootkit Attacks “,International Conference on Intelligent Networking and Collaborative Systems (INCoS), PP**.** 703 – 708,2011.

[20] Hai Bi, “Anti-rootkit Technology of Kernel Integrity Detection and Restoration”, International Conference on Network Computing and Information Security (NCIS), Pp**.** 276 – 278,2011.

[21] Watters, P. ; Xinwen Wu ,” RBACS: Rootkit Behavioral Analysis and Classification System”, *Third International Conference on Knowledge Discovery and Data Mining*,pp.78-80,2010

[22] Yu-Jie Hao, Yan Zhang, Zhi-Peng Lu and Rui Zhang ,”A New Malware Detection Method based on Raw Information”, International Conference on Computing and Intelligence Analysis, pp- 307 - 310,2008.

[23] Jun Han ,”Security Attack Ontology for Web Services”, *Second International Conference on Semantics, Knowledge and Grid*,pp.42-50,2008. Sacco, Anibal; Ortéga, Alfredo (2009-06-01). "Persistent BIOS Infection: The Early Bird Catches the Worm". Phrack. 66 (7). Retrieved 2010-11-13.

[24] Sacco, Anibal; Ortéga, Alfredo (2009-06-01). "Persistent BIOS Infection: The Early Bird Catches the Worm". Phrack. 66 (7). Retrieved 2010-11-13.

[25] http://studylib.net/doc/9005710/lab-5-rootkits--backdoors--and-trojans

[26] Protégé https://protege.stanford.edu/

[27] Feilmayr, Christina; Wöß, Wolfram (2016). "An analysis of ontologies and their success factors for application to business". Data & Knowledge Engineering: 1–23. Retrieved 23 May 2017

[28] Ontology Evaluation and Ranking using OntoQA - IEEE Xplore ..., ieeexplore.ieee.org/document/4338348/

Ahmad Salahi received his B.Sc. degree from Tehran university in 1970,M.Sc. from Kansas University, Lawrence Kansas in 1974, and his Ph.D. from Purdue uiversity, West Lafayette,Indiana,U.S.A. in 1979 all in electrical engineering. He is currently an associate professor in Iranian Research Institute for ICT (ex. ITRC). His research interests are network security, switching and routing.

Javad Enayatizadeh received his M.Sc. degree in Information Technology from Iran university of science & technology in 2010. His main interest is software programming with focuses on network.