Malware Detection based on Cloud Computing integrating Intrusion Ontology representation

Cristian Adrián Martínez
Ecole supérieure d'Ingénieurs en Electronique et Electrotechnique
París – Francia
adrian.martinez@ieee.org

Gustavo Isaza Echeverri
Systems and Informatics Department
Caldas University
Manizales, Colombia
gustavo.isaza@ucaldas.edu.co

Andrés G. Castillo Sanz
Systems and Informatics Department
Pontifical University of Salamanca,
Madrid Spain
andres.castillo@upsam.net

Abstract—**This paper aims to present a model for malware detection, uCLAVS (University of Caldas' AntiVirus Service), a multiple engine service that follows the set of defined protocols and standards for Web Services technologies, in addition an Ontology for Malware and Intrusion Detection is described. uCLAVS is based on the idea that the files analysis commonly carried by applications residing on the client can improve their performance if they are moved to the network, where instead of running complex software on every host, it gives each process a receiving the light entering the system files, send them to the network to be analyzed by multiple engines, and then to decide whether or not they are executed according to the report of threat delivered. As a result of the tests with the prototype can be uCLAVS arguing, among other things, that offers the possibility of increasing the rate of the assertion characterization harmful files, allows the construction of thin clients, facilitates zero-day updates, and provides a forensic capabilities enhancement. (Abstract)**

Keywords- Cloud Computing, Malware Detection, Antivirus, Web Services, Cloud Security Intrusion Detection, Intrusion Prevention, Ontology for Malware Detection

I. INTRODUCTION

The Malware Detection is one of the most security challenges. The operation mode of this type of software is based primarily on the use of signatures and heuristics. A common phenomenon that supports this approach is the increased complexity in software designed to combat malware. These situations have increased the antivirus complexity and contributed indirectly to become a focus of vulnerabilities and attacks. The current knowledge representation standards for malware and intrusion are based on taxonomies; therefore these become insufficient to support optimal attacks identification, reasoning and predictive anomalous behaviour's definition. Ontologies allow to describe objects, concepts and relationships in a knowledge domain, for this case, malware signatures resources, detection rules, reaction and prevention processes need to be semantically described in order to facilitate uniformity in the knowledge bases of the systems underlying and the possibility to providing a reasoning framework, intelligence and inference from these models onto-semantic. This paper proposes an architecture for malware detection based on the concept of Web Services and Malware Intrusion Ontology, *uCLAVS (University of Caldas' AntiVirus Service)* a service deployed in the computing cloud that follows the set of protocols and standards defined for Web Services technologies, which can detect malicious content or behavior of an unknown file through the use of multiple engines that implement heterogeneous analysis strategies. Section 2 provides a brief review of the state of the art and the most relevant contributions; section 3 describes the architecture, models, specifications and services deployment, finally show some preliminary findings of the prototype model and its underlying design and section 4 presents the ontology defined for malware/intrusion detection and prevention. Results are evidenced in Section 5. Finally, the main conclusions of our research as well as the future work are highlighted in Section 6.

978-1-4244-7173-7/10/$26.00 ©2010 IEEE

II. PREVIOUS WORK

Malware detection in a Cloud Computing service was explicitly introduced in [1], but filter systems for e-mail and HTTP protocol is deployed in-cloud were becoming popular for some time ago. As shown in [2] is possible as a form of protection services. The tools listed in the ICAP protocol, such as antivirus service (running on the same machine) can serve as a multi-purpose scanner to facilitate the detection of viruses via e-mail, web and proxy servers.

Our proposal using Multi-engine protection heterogeneous systems provides a better assertion in characterizing harmful files having the implementation of multiple protection engines that use heterogeneous methods and techniques of analysis and detection.

The standards for representation and ontologies in Intrusion Detection system manifest a precarious effort, proposes as IDMEF (Intrusion Detection Message Exchange Format) and the CIDF (The Common Intrusion Detection Framework) [3] defined APIs and protocols for research projects in intrusion detection that can share information and resources, as well as components which can be refused by constructing a model of representation based on XML syntax. The research developed in [4] has defined a target centric ontology for intrusion detection, new approaches have proposed ontologies as a way to represent and understand the attacks domain knowledge, expressing the intrusion detection system much more in terms of their domain and performing intelligent reasoning. These projects aim to define an ontology DAML-OIL (DARPA Agent Markup Language + Ontology Interface) target centric based on the traditional taxonomy classification migrated to semantic model, the investigation done in [5] integrates ontology entities and interactions captured by means of an centric-attack ontology which provides agents with a common interpretation of the environment signatures which are matched through a data structure based on the internals of the Snort network intrusion detection tool. In [6] we have developed an ontology for intrusion detection and prevention, complementing, we propose to extend the domain of this ontology to integrate malware signatures based on the results generated by the multi-engine embedded in the uCLAVS architecture.

III. MALWARE DETECTION USING CLOUD COMPUTING

A malware analyzer aims to determine how a software code has the ability to make an attack on a computer system [7]. To accomplish this, the current solutions such as antivirus programs use mainly static analysis based on signatures and heuristics [8], recent techniques have been adopted dynamic analysis and other similar defense strategies. A common scenario in the detection of malicious code is a resident in the host application which implements specific algorithms to discover as much malware as possible. The activity of these tools are most often focuses on the analysis of executable files from outside and takes place mainly in the terms of access time and on demand.

The security systems must be scalable to accommodate a significant number of clients. A multi-engine based file analysis service is a remote system that can identify the content or conduct of an unknown file through the analysis of multiple engines (antivirus) that implement heterogeneous strategies. *uCLAVS*, is a multi-engine based file analysis service deployed in the cloud computing through the set of protocols and standards for web services. The functionality of the service aims to be simple and robust: Identify a file with malicious code through the remote analysis performed by multiple engines.

The WWW Consortium W3C defines a Web Service as "... *a software application identified by a URI, whose interface (and use) can be defined, described and discovered by XML artifacts, and support direct interactions with other applications defining a software library operations remotely accessible using XML messages using standard Internet protocols*. "(W3C 2007)

A complete Web Service represents a service that complies with the following characteristics:

- Is available from the Web.
- Use the XML standard for sending messages.
- Not tied to the programming language or operating system.
- It is able to describe by itself.

Web Services are service-oriented architecture (SOA) implementation, which proposes a dynamic implementation, low coupling and distributed applications. SOA consists of three main roles: a service provider, a consumer and a broker. Similarly divide the major functions and components used in the description of the architecture [9].

Fig. 1 shows a general context diagram for the our proposed model based on Web Services.

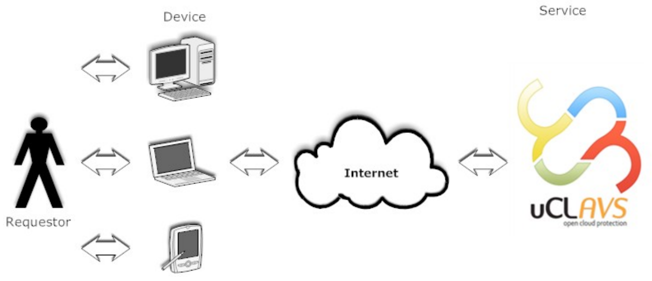


Figure 1. Context Diagram for uCLAVS

The main CLAVS utilities are:

- uploadSample (file)
- doScannerAnalysis (hash)
- getScannerAnalysis (hash)

These functions were designed to decouple the function of the service provider, which is responsible for detection, and the customer that based on the result obtained is in charge of making a decision.

uCLAVS provides the same capabilities for detecting anti-malware products based on the client, this implementation will have to follow some additional provisions specific to its nature of service in the cloud. For antimalware services, it must meet some requirements such as:

- *Support for multiple analysis engines*. The facility must allow the use of multiple parallel protection engines using heterogeneous methods and techniques for detecting malware.
- *Notice*. When a file is detected as potentially dangerous, the service must provide the user with information necessary to know the incident and take the right decision.
- *Collection of information*. All service activity should be gathered and accessible for analysis and management purposes.
- *Service Management.* Must provide mechanisms to configure and manage the service.

The two key aspects that make *uCLAVS* a complementary alternative to improve the automatic detection of malware are the nature of Web Services and the ability to analyze a file using multiple engines for protection under a paradigm called n-protection.

Multi-Engine

One of the main focus of this implementation is the ability to use multiple security engines that use heterogeneous methods and techniques for detecting malware, this model is called N-version protection [1] is based on the concept of N-version programming, which proposes the creation of multiple versions of an application in order to compare their outputs and thus ensure proper operation. This approach provides several implementations of a Web application by using display technologies and different platforms, in order to compare the output reached, represented in HTML, in order to ensure proper operation of the system. Fig. 2 presents our model for *uCLAVS* architecture, as well as layers, levels and services offered.

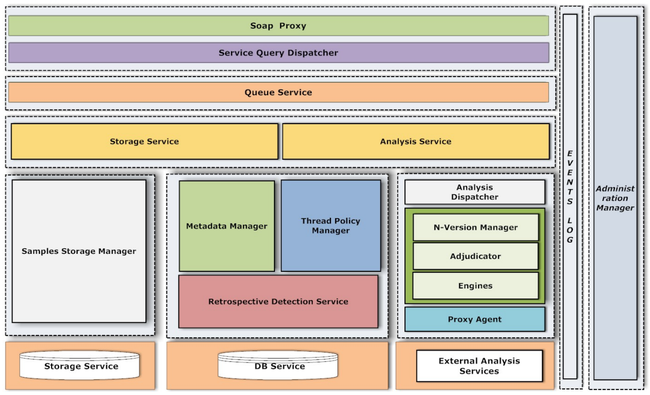
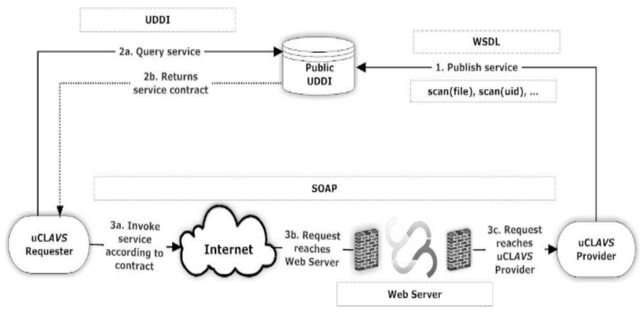


Figure 2. uCLAVS Architecture Components

The description of the components in the architecture is presented as:

- *Proxy SOAP:* The SOAP proxy is responsible for serializing / de-serialize *uCLAVS* messages exchanged between consumers and their customers so that implementations can use a transparent manner.
- *Dispatcher:* Dispatcher requests plays a central role in the architecture, functions as a queue that lets manage incoming service requests, while reporting to the event log.
- *Queue:* The queue in the architecture is a data structure, characterized by a sequence of elements that push the insert operation performed at one end and pop the mining operation on the other. It is administered by the dispatcher.
- *Storage Service: uCLAVS* provides a simple interface that uses Web Services to store and retrieve any amount of data, anytime, from anywhere on the Web.
- *Service Analysis:* Define a Web Services interface, which requested an analysis of a possibly malicious file, this will get the hash of the file to be analyzed by the engines.
- *Analysis Query Dispatcher (AQD):* It behaves as the dispatcher, but its function is to administer and manage the demands of the engine.
- *Adjudicator:* Responsible for monitoring the engine's performance and work.
- *Engines:* Correspond to analysis engines, *uCLAVS'* additions within the concept proof that use five different engines, Clamv, F-Prot, Avast, BitDefender, Kaspersky
- *Proxy Agent:* proxy agents are responsible for communicating the architecture and services that are provided by external agent services.
- *Metadata Manager*: The metadata manager is responsible to organize and control the information delivered by different engines.
- *Policy Manager Threat:* The threat alert manager is responsible for delivering the final report of threats.
- *Retrospective Detection:* The retrospective detection emphasizes the virus detection that are not in the database, just as we analyze the files that are suspicious but do not have an alert level is still high to be considered malicious or not.
- *Log Events:* The event log is responsible for controlling access of different users to implement or use the services of uCLAVS.
- *Administration Manager:* Provides an interface to manage all the processes mentioned above.

Fig. 3 shows the context of the interaction of roles within our platform uCLAVS.



UDDI
2a. Query service
2b. Returns service contract
Public UDDI
WSDL
1. Publish service
scan(file), scan(uid), ...
SOAP
uCLAVS Requester
3a. Invoke service according to contract
Internet
3b. Request reaches Web Server
3c. Request reaches uCLAVS Provider
uCLAVS Provider
Web Server

Figure 3. uCLAVS Roles Interaction Context

For the Service implementation we have used the PERL language package SOAP::Lite available from CPAN. SOAP:: Lite that is a collection of Perl modules which provide a lightweight API to use SOAP clients and servers. The requests processing is in charge of *distpacher*, implementation of uCLAVS Distpacher summary follows:

```
use SOAP::Transport::HTTP;
my $uri   = 'urn:uclavservice';
SOAP::Transport::HTTP::CGI
                ->serializer  ($serializer)
                ->deserializer($deSerializer)
```

```
                 ->dispatch_with(
                 {"$uri:analysis" =>

'uCLAVS::Dispatcher::AnalysisService',
                 "$uri:storage"    =>

'uCLAVS::Dispatcher::StorageService'})
                 ->on_action(\&onAction)
                 ->handle;
```

The following simplified code shows the integration for a free antivirus engine embedded uCLAVS distribution.

```
sub scan
{
my ($self, $path) = @_;
my ($exitcode, $scan_response) =
eval { $self->_run_commandline_scanner
(join(' ', $self->{command},
@{ $self->{args} }, $path, '2>&1')); };
if($@) {
return     uCLAVS::Analysis::Scanner::Result-
>error($@);
}
if(0 == $exitcode) {
return     uCLAVS::Analysis::Scanner::Result-
>clean();
}
if(1 == $exitcode) {
my ($virus_name) =
$scan_response =~ m/\[infected by: (.*) .+/i;
if(!$virus_name) {
$virus_name = 'unknown-Avast-virus';
}
return     uCLAVS::Analysis::Scanner::Result-
>virus($virus_name);
}
(…)
}
```

An example of service description for the process "*doScannerAnalysisResponse*" is shown in the following Web Services description:

```
parts     fileHash

           type  xsd:string

Used      Operation doScannerAnalysis
by        in PortType uCLAVSPort

Source   <wsdl:message
         name="doScannerAnalysisRequest">
           <wsdl:part name="fileHash"
         type="xsd:string"/>
         </wsdl:message>
```

In Fig. 4 we present relational models for the management of meta-information files and the results of the analysis.

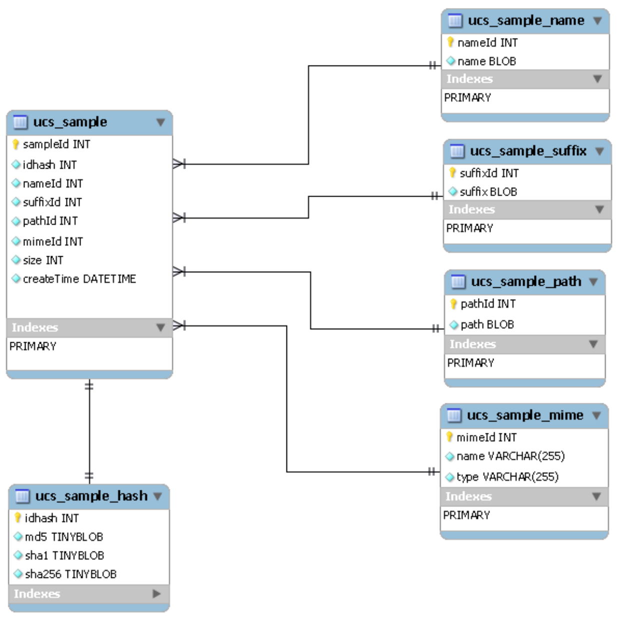


Figure 4. Relational model for meta-info Files

The Client implementation has been developed using PERL and implementation of SOAP: Lite client built a GNU /Linux for *uCLAVS*. In the case of platforms under Windows was used. For .NET we have used *POCKETSOAP*.

IV. ONTOLOGY

The problem presented in our model defines multiple concepts, relations and the meaning integration with other security technologies like firewalls and vulnerabilities scanner. Ontology can help to optimize the knowledge representation, reduce the messages size, decrease the redundancy and allow incorporating more intelligence in the information analysis.

For now, multiple axioms and rules are describing the attack types by a total of 25 intrusion classifications obtained through the applied clustering algorithm (K-Means), described in [6] and near 4800 instances for our ontology. To design and implement the ontology, OWL language was used, which allows formalisms and knowledge representations, some of the most important influences in the design of OWL are derived from its predecessor DAML + OIL logic of descriptive paradigm frameworks models and RDF / XML. To define the rules we are using the Semantic Web Rule Language (SWRL), it is an expressive OWL-based rule language. SWRL allows writing rules expressed in OWL concepts providing reasoning capabilities. A SWRL rule contains an antecedent part to describe the *body*, and a consequent part, referring the *head*. At the moment we have defined around 2224 attacks and malware signatures and 2210 prevention rules instances in our ontology, in addition we described the events sequence that happen in real time. The predicates definition describes sequence attacks having input causes and possible consequences. A hybrid source data was used, additionally the attacks were created with other testing tools, normal traffic captured using *JPCap* and *tcpdump* and processed. The raw data was converted to XML then processed to OWL instances; furthermore the ontology is updated via SPARQL sentences. The ontology represents the signatures for known attacks (network intrusions and malware detected) and novel attacks, the intelligent behaviour uses the inference model and reasoning tools integrating neuronal networks in the multi-agent system, we have published this solution in [6]; this implementation provides an ontological model for reaction rules creating the prevention system. For this ontology we have defined rules that allow properties inferences and reasoning process. The *Malware-Behaviour* is defined as ontology' attributes, from the captured and processed attack instance using the detection engines embedded in *uCLAVS* architecture, identifying the *Type of Intrusion*. This value is extended to other types of intrusion that describes not only the malware detected by the engine, but also the set of intrusions detected by the attack detection engine. The description for the axiom that describes a *RootAccess* state is denoted as:

$$
\begin{aligned}
\text{RootAccess} \equiv \ni (&\text{IntrusionState} \cap \text{InputTraffic} \cap \\
&\text{NetworkNode}) \\
\cap \ni\ &\text{Generated\_by}(\text{AnomalousSSH} \cup \\
&\text{AnomalousFTP} \\
&\cup \text{WebAttack} \cup \text{TelnetAccess}) \cap \\
&\text{UID\_Try}(\text{UID\_0})
\end{aligned}
$$

Using JESS as complement to incorporate reasoning and inference model, it's possible to give the semantic capabilities to the ontology. JESS is a rule engine and open source environment that allows to construct Java applications with reasoning using knowledge providing from declarative rules [10]. An example for an inferred axioms using JESS in our model is giving by:

```
        SystemWebAccessState
  (http:/url/../OntoIDSSMA.owl#
        WebAttack_XSS,Nodo2)
       SystemMalwareDetState
  (http://url/../OntoIDSSMA.owl#
        W32/Scar-H,Nodo1)
        UnderSynFloodAttack
  (http://url/../OntoIDSSMA.owl#
           Nodo1,True)
        UnderBufferOverflow
  (http://url/../OntoIDSSMA.owl#
           Nodo3,True)
```

From these representations, it's possible to construct attacks scenarios to define preconditions and post-conditions to determine for example that A is a set of subclasses of relations between attacks A_1 where if an attack is an attack type A_2, then we say that A_1 is A_2 subclass of attack and the super-class of A_1 and if an attack has two states S_1 and S_2 prerequisites and there is an instance of A with their respective bodies and S_1 and S_2 are prerequisites of a state, then to succeed. This SWRL representation is given by:

```
   A(?p) ∧ S₁(?q) ∧
PreconditionIntrusionState(?p, ?q)
   ∧ S₂(?z) ∧
   PreconditionIntrusionState(?p, ?q) →
State(?p, true)
```

Complementary, our correlation function using Ontologies and Multi-agent systems has been described in [11], where we have explained how to integrate the semantic model in MAS having an intelligent behaviour and correlation based on similarity attributes technique.

V. RESULTS

The tests were made with about 1.2 million samples and approximately 25,000 malware instances from a total of 31 groups. As shown in Fig. 5, the malware detection rate is between 85% and 95%, uCLAVS using an architecture based on multi-engine and the properties of minimizing the burden on customers, reach a detection rate near to 97% in the first experiment. Table I and Fig. 6 shows the detection rate for the 6 most popular antivirus and our proposal *uCLAVS* depending on the age of malware samples for 1 week and 1 month.

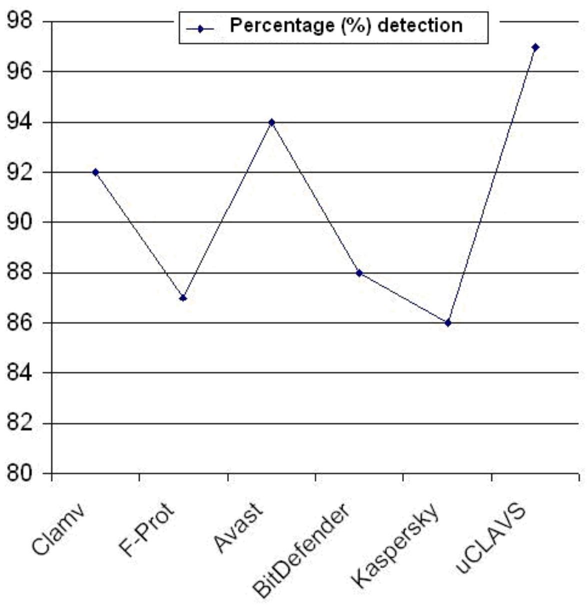


Figure 5. uCLAVS Detection Rate vs other Anti-Malware Engines

TABLE I. DETECTION RATE WITH THE AGE OF MALWARE ON DIFFERENT PLATFORMS ANTIVIRUS

AntiVirus	1 month	1 week
Avast	54,2%	51,1%
AVG	84,4%	82,2%
BitDefender	81,2%	79,3%
ClamAV	56,7%	54,2%
F-Prot	53,4%	51,2%
Kaspersky	89,1%	86,8%
uCLAVS	***97,07%***	***93,4%***

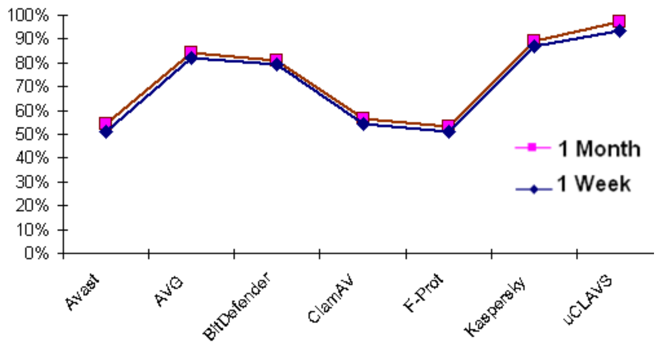


Figure 6. Malware detection rate with the age of Malware on different platforms AntiVirus

Integrating the Ontology having our Complete Intrusion Detection Model the Fig. 7 (a) shows the detection rate percent comparing with other IDS having six (6) sensors; this performance may improve by the integration of pattern recognition, classification and inference model. Additionally, the network overhead is reduced using multi-agent system and incorporating OWL embedded in the message for information exchange between the multiples engines as shown in Fig. 7(b). Snort were used for the standard IDS to compare the results achieved having multiple sensors.

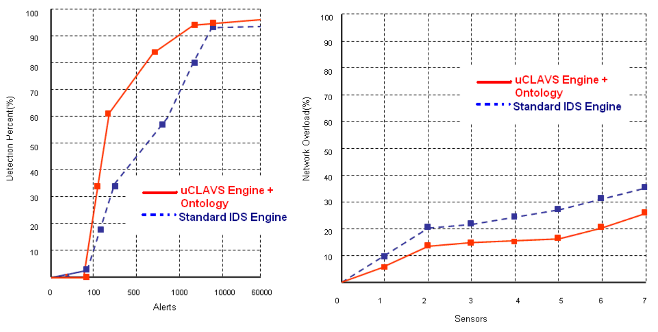


Figure 7. (a) Detection Percent vs. Alerts Number (b) Network Overload

VI. Conclusions and Future Work

This paper presented the architecture of a deployed service in the cloud computing called *uCLAVS.* For the definition of architecture has been used in Web Services-based technologies that define a communication link by using different standards under the W3C and integrating Ontologies for Malware and Intrusion Detection Events, which allowed different clients implemented (linuCLAVS and WinuCLAVS) access the service through the XML standard, using SOAP. uCLAVS to use different scanning engines have a malware detection rate over 97 %, which is higher than any of the other engines used in the proof of concept: Clamv, F-Prot, Avast, BitDefender, Kaspersky.

By the time the whole system integration is still under development, we will continue testing and integrating the model the semantic inferences rules in the Web Services system searching an approach to provides a complete correlation behaviour using an extended ontology having intrusion and malware signatures.

References

[1] J. Oberheide, E. Cooke, and F. Jahanian: CloudAV: N-Version Antivirus in the Network Cloud. En Proceedings of the 17th USENIX Security Symposium (Security'08). San Jose, CA. . 2008

[2] S. Link. Server-based Virus-protection On Unix/Linux. University of Applied Sciences Furtwangen. *http://www.openantivirus.org/diploma-thesis.pdf.* Consulted, 2008

[3] S. Al-Mamory and H. Zhang: Intrusion detection alarms reduction using root cause analysis and clustering, Butterworth-Heinemann. PP. 419-430, 2009

[4] J. Undercoffer, T. Finin, A. Joshi, and J. Pinkston: A target centric ontology for intrusion detection: using DAML+OIL to classify intrusive behaviors. Knowledge Engineering Review - Special Issue on Ontologies for Distributed Systems, Cambridge University Press., PP. 2-22, 2005

[5] S. Mandujano, A. Galvan, and J. Nolazco. An ontology-based multiagent approach to outbound intrusion detection. in Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on Security. PP 94, 2005

[6] G. Isaza, A. Castillo, M. Lopez, and L. Castillo. Towards Ontology-based intelligent model for Intrusion Detection and Prevention. in 2nd International Workshop on Computational Intelligence in Security for Information Systems CISIS'09 Springer.PP 109-116, 2009

[7] R. Dalla: Code Obfuscation and Malware Detection by Abstract Interpretation. , Ph.D. Thesis, Università degli Studi di Verona. PP. 127, 2007

[8] P. Szor: The Art of Computer Virus Research and Defense (illustrated edition.). . Addison-Wesley Professional. PP 245-252 , 2005

[9] M. Papazoglou: Web Services: Principles and Technology (1o ed.). Prentice Hall. , PP. 22, 2007

[10] E. Friedman-Hill and L. Sandia. Jess, The Rule Engine for Java Platform. Consulted: 2009; http://www.jessrules.com/jess/docs/index.shtml, 2009

[11] G. Isaza, A. Castillo, M. Lopez, L. Castillo, et al. Intrusion Correlation using Ontologies and Multiagent Systems. S.K. Bandyopadhyay et al. (Eds.): ISA 2010, CCIS 76, pp. 51–63, 2010. Springer-Verlag Berlin Heidelberg 2010The 4th International Conference on Information Security and Assurance (ISA 2010). In Miyazaki, Japan. June 2010.