CONTENTS

Automat. Comput. Appl. Math.
Volume 27 (2018), Number 1

COMPUTER SCIENCE

VMI-based Protection and Control of a Linux VM Running Security-Sensitive Applications

Andrei Luțaș*, Bogdan Sîrb†, Adrian Coleșa *, Alexandru Gurzou*, Adrian Pop*
Sandor Lukacs†, Raul Toșa†, Daniel Țicle†
*Computer Science Department, Technical University of Cluj-Napoca, Romania
†Bitdefender, Romania

***Abstract*—We propose several methods that use virtual machine introspection (VMI) to protect the entire software stack of a Linux virtual machine (VM). At the user-space level we prevent exploits and malicious code injection, while at the kernel-level we protect the kernel image and important system registers. We also control the processes that could be run and kernel modules that could be loaded. We applied our solution to protect a specially prepared trusted (green) Linux VM, aimed for running security-sensitive applications, in contrast to an untrusted (red) VM in which security-insensitive applications are run. Our protection methods introduce a very small performance overhead (under 2%) during processes' runtime. The high overhead (about 50%) is only during process creation, which is however a rare event on real-life systems (just few for our green VM) and just a one-time price paid during a process lifetime.**

***Keywords*—virtual machine introspection (VMI), virtual memory area (VMA), Linux, code injection prevention, exploit prevention**

I. Introduction

The traditional virtualization-based security strategy is based on analyzing the protected VM's state from the virtualization system, i.e. the *hypervisor* (HV) and reacting to malicious actions or contents. Such a strategy, named *VM introspection (VMI)* [1], was used for both intrusion detection and prevention. The main problem the VMI faces is the *semantic gap* [2], which makes semantically understanding the VM's memory contents difficult. There two different ways [3], [4] to narrow the semantic gap by using knowledge about either: (1) the internal details of the guest OS in the introspected VM, or (2) the architectural elements imposed to the VM.

The first method is highly specific to the guest OS, but provides good precision and fine-grained security-relevant details. The second one is guest OS's agnostic, yet provides low precision and only coarse-grained details. So, practical solutions [5] combine both of them, with a greater weight of the first one.

We propose a VMI-based solution that provides both user and kernel space protection in a Linux VM. At the user-space level we prevent exploits and malicious code injection, while at the kernel-level we protect the kernel image and important system registers. While based on intercepting OS-specific memory manipulation functions, our solution is minimal and general in the sense that it monitors only low-level operations performed on virtual memory areas (VMA) by all memory allocation modules in the OS, independent of the semantic data structures they store in that areas. This is different by other strategies [5], which intercept higher-level functions that manipulate specific data structures.

Another virtualization-based protection strategy is to run the user applications in two different VMs [6], based on their sensitivity to security aspects: (1) security-sensitive applications in a trusted VM, named the *green VM*, and (2) the other applications in a different, untrusted VM, named the *red VM*.

We applied our VMI protection to a Linux green VM on an end-user system using a variant of the red-green VM separation strategy, which keeps active only one VM at a time [7]–[9], while having the other VM shutdown. We control the processes that could be run, kernel modules that could be loaded and protect both of them. The relatively high overhead our VMI protection introduces on process creation operations is highly attenuated on such systems, due to the small number of applications that are run and their relatively large runtime (e.g. starting a Web browser even with tens of milliseconds delay go unnoticed in comparison to the thousands of seconds of usage).

The main contributions of our paper are:

- propose a VMI-based protection method of user-processes against exploits and code injection, by monitoring all operations performed on processes' virtual memory areas;
- apply our VMI-protection solution to a small Linux green VM, running just few user processes.

II. Protection Strategy Overview

A. Two-Level Protection

Our VMI protection strategy is split in kernel-space protection and user-space protection. The first provides protection against kernel-specific attacks, such as rootkits, kernel-exploits or privilege-escalation attacks. The second protects against process-specific attacks, such as code-injections or exploits.

Our protection is real-time, VM's components being protected when created and attacks caught when they take place, allowing us to block the attack and provide its timeline, by identifying both the components that were under attack and the components that issued it. For example, the OS kernel is protected as soon as it is initialized, e.g. when it configures specific system registers, such as the interrupt descriptor table or system call registers. User-mode processes are also protected as soon as they are created, such that by the time the threads of a process start running, the VMI has already enabled protection of that process. We consider the protected VM's guest OS to be clean at the moment it starts, so our main concerns are attacks that may take place while it is running.

B. Guest OS Function Interception

Since the Linux kernel is built with function tracing support, each function starts with a CALL instruction (5 bytes)

Time performance and power efficiency of Interleaved Scrambling Technique for cache memories

Mădălin Neagu
Computer Science Department
Technical University of Cluj-Napoca
Cluj, Romania
Email: Madalin.Neagu@cs.utcluj.ro

***Abstract*—Every bit of information that travels through Internet is, at some point, stored in the internal memory of a device for a period of time. The cache memory is at the core of every device and plaintext data that is used by the CPU is retained there for a small amount of time. Although the cache memory is considered secure due to the fact that it is embedded in the same chip as the CPU, several types of attacks like cold-boot or side-channel have been discovered that target these types of memories.**

The Interleaved Scrambling Technique is a methodology proposed to counter attacks against cache memories by concealing or hiding the sensitive information from the adversary. The proposed technique scrambles the plaintext data with different scrambling vectors, thus the cache memory stores scrambled data only. When the data is needed, a descramble operation is performed and the original information is restored.

In this paper, the time performance and power efficiency of the Interleaved Scrambling Technique is discussed and explained. Furthermore, the CACTI tool is used to properly evaluate the proposed technique, in terms of area overhead, power consumption and access times.

***Keywords*—memory systems, data scrambling, cache memories, cold-boot attack, side-channel attack, time performance, power efficiency, CACTI**

I. Introduction

Attacks on computer memories that target different vulnerabilities are common in the last decade, mostly because they store important or sensitive information. In 2014, the authors of [2] reported that 62% of the companies worldwide have encountered payment frauds, especially targeting credit/debit cards. Biometric information is also a known target for attacks, as presented in [3]. Usually, the information is encrypted and cryptographic operations are executed by a software, which denotes heavy memory usage [4], while the cache memory is used to store the sensitive data at risk [5].

Due to the remanence effect [6], the data stored in a memory is visible for a small period of time after a power loss. Furthermore, if the memory modules are kept in a low temperature environment, the information can still remain intact for about 5 minutes [7]. Thus, a *cold-boot* attack can be applied [7], [8] to recover the last data stored in the memory, by using a backup system to download the contents. A solution to this is data encryption [8], [9], [10], but it creates a decrease of the bus throughput. The encryption algorithm uses the cache to store the encryption keys in plaintext (due to performance issues), which allows for different types of attacks [4]. Avoiding main memory transfers while the encryption algorithm is running is also a possibility [12].

However, the cache memory is also a target for attacks and in [11], [12] non-invasive attacks are investigated and discussed, while semi-invasive ones are explained in [13]. They are know as *side-channel* attacks and they use leaked information by a cryptographic component or device, like power consumption, time, etc. Hence, AES encryption algorithms are a target of such attacks [12]. Also, direct reading of cache memory is possible, as performed in [13], [14].

A methodology was proposed by the same authors in [17], [18], designated to increase the security of the cache memory against cold-boot and side-channel attacks. The proposal named Interleaved Scrambling Technique (IST) uses data scrambling to conceal/disguise the plaintext data from the cache memory and make it unusable if successfully retrieved by other means. This paper contains the evaluation of the IST in terms of time performance and power efficiency. Also, the CACTI tool is used to evaluate the area overhead, power consumption and access times of the proposed technique.

The rest of the paper is organized as follows: Section II contains a short review of the Interleaved Scrambling Technique, Section III describes and explains how the time performance and power efficiency is evaluated and in Section IV the CACTI tool results are presented. The last section, V, contains the conclusions of the paper.

II. Short review of Interleaved Scrambling Technique

As presented in section I, there are several proposals for memory systems to defend or protect against *cold-boot* or *side-channel* attacks. Between them, scrambling data vectors with a unique or secret key is the one that most often appears in literature. Keep in mind that data scrambling does not assure strong security as compared to encryption, but it is faster, requires a low area overhead and consumes less power.

For the rest of this section, the data scrambling is explained below, followed by a short review of the Interleaved Scrambling Technique.

A. Data Scrambling

The data scrambling operation is performed on the data vector DV and scrambling vector SV:

$$SD = f_S(DV, SV) \equiv DV \oplus SV \tag{1}$$

The outcome of the equation is SD, the scrambled data, where each bit is the XOR between the corresponding bits of DV and SV. If SV is hidden, it is not possible to know

Image Rectification and Matching Solutions Optimized for Dedidated Hardware Accelerators

Cristian-Cosmin Vancea
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
e-mail: Cristian.Vancea@cs.utcluj.ro

Sergiu Nedevschi
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
e-mail: Sergiu.Nedevschi@cs.utcluj.ro

***Abstract*—The late explosion of hardware accelerated devices on market has enabled much interest towards fast and energy efficient solutions for the computationally intensive problem of image matching and dense 3D reconstruction. From the perspective of camera projection model the task can be simplified if the images are rectified for epipolar alignment in a preprocessing step. We focus our interest on identifying a rectification model and several classes of matching algorithms as candidate for real-time implementation into embedded devices installed on vehicles. Then we make a thorough analysis of existing solutions and identify the capabilities offered by the hardware underneath. We present our solutions for image rectification and matching in FPGA. We point the benefit from implementing an original caching technique which optimizes access to the image memory when performing rectification. Additionally, we have designed several optimization strategies capable to reduce the hardware costs for image matching. Our system running on an old class FPGA performs image rectification and matching in real-time and classifies between the top best solutions in computational performance and energy efficiency.**

Keywords—stereovision; image rectification; image matching; FPGA; digital design

I. INTRODUCTION

The field of autonomous vehicles requires state-of-the-art solutions in domains such as perception, decision-taking, motion planning and control. Perception plays a significant role in the pipeline by extracting accurate information from the surrounding environment, which is critical for searching a feasible trajectory at the higher decision-taking level. Details such as vehicle dynamics, position of obstacles, road boundaries, represent critical data for accurate and safe in-traffic decisions. With digital world becoming more visual the number of images needed to be processed in real-time is increasing. Such a task is demanding in both, computation and memory. Single-machine environments [1] often lack sufficient computational power and memory, while multi-machine environments add communication and control overhead. For hardware makers the push is to create low-cost systems to ensure real-time analysis and reduce the amount of data traffic. There is increasing demand for smaller, energy-efficient and powerful computational systems that can be placed outside offices, in the end devices, such as smartphones or automated vehicles. That requires not only new materials but also different packaging to ensure high performance with low energy cost and increased density.

In terms of energy efficiency Field Programmable Gate Arrays (FPGA) are superior to high-end Graphics Processing Units (GPU), which offer impressive floating-point performance. Technology is advancing rapidly and with the increasing number of integrated Digital Signal Processing (DSP) units in the FPGA fabric, their floating point capability is continuously improving. The key point for both classes of devices is their high degree of parallelism.

Regarding applications for Advanced Driver Assistance Systems (ADAS), the need to facilitate smarter vision tasks is well concentrated on implementation with hardware accelerators [2][3][4] due to their capabilities for real-time analytics (detection, recognition, classification and tracking), video processing (2D, 2.5D, 3D visualization and reconstruction) and intelligent data transport through I/O components (Ethernet, AVB and CAN).

Nowadays markets share a large variety of components based on LIght Detection And Ranging (LIDAR), radar, ultrasonic, infrared or video bandwidth. These sensors provide data in different formats and one of the most challenging tasks is to correlate measurements into a unique representation viable for processing. Images captured by cameras are represented by pixels, which must be converted into distances expressed in a meaningful world coordinate system. Aiming for rapid and energy efficient 3D reconstruction we make an ample overview of state-of-the-art in hardware optimized solutions for image rectification and matching. We identify the algorithmic models commonly used in the literature and we analyze a consistent list of implementations for various platforms, while considering for possible accuracy flows, their speed, the computational performance and the energy consumption. We contribute with our original architectures implementing image rectification and matching in FPGA. For image rectification we developed an original caching technique which improves the access to image memory. The image matching entails optimization solutions capable to reduce the amount of allocated hardware.

The paper is organized as follows. In Section II we present the analytical model for image rectification and the classes of algorithms for image matching used by a majority of implementations proposed in the literature. In Section III we make a thorough study of existing solutions for image rectification and for image matching, which entail optimization strategies for hardware acceleration. We also refer to our proposed solutions and we initiate a parallel analysis to properly identify the contributing elements. Section IV offers a brief list of conclusions.

Description Logics: Applications on the Semantic Web

Andrei Zamfira
Dept. of Computer Science
Politehnica University of Timisoara
Timisoara, Romania 300006
Email: andreizamfira@gmail.com

Horia Ciocarlie
Dept. of Computer Science
Politehnica University of Tmisoara
Timisoara, Romania 300006
Email: horia.ciocarlie@cs.upt.ro

***Abstract*—This parer is intended as a literary review research article in which are presented the domains of application of the newly emerged technology, the Semantic Web. As the title suggests, the main goal is to show the role that logical formalisms have in the creation of languages from the stack of Semantic Web, the inference and reasoning algorithms created for them etc. Are presented the main DL languages that have been proposed in the literature by some famous scholars, then the current work makes a series of analyses and comparisons regarding the features that have been added in each one in order to capture and express new situations from the real world, and are explained with concrete examples. Next is shown the correspondence between these logical formalisms and ontology languages, how were built on top of them, are made comparisons related to their syntax and semantics and presented in tabular form. The article ends with a state-of-art and conclusion sections which summarizes the current work and state its achievements.**

***Keywords*—Semantic Web, Descripiton Logics, ontology language, knowledge representation, logical formalism**

I. Introduction

A. World Wide Web

The World Wide Web (abbr. WWW, or simply Web) was invented at the beginning of 90s by the english scholar Tim Berners-Lee, which proposed to use the hypertext to link and access information on the Internet, like a web with nodes. It is a system of interlinked hypertext documents, called web pages that consist of multimedia contents (texts, images, video), which can be accessed and viewed using a web browser. It operates based on a client-server architecture, where documents are stored into a web server and accessed and displayed by a web browser by means of their identifiers in form of Uniform Resource Identifiers (URIs, URLs). The day of 6 August 1991 is considered the debut of the Web as a service publicly available on the Internet. It ran on a NexT computer at the European Center for Nuclear Research (CERN) in Geneva, Switzerland.

Many people use the terms Internet and World Wide Web to denote the same thing. It should be clear that they are not synonyms. Internet is a global system of interconnected networks that communicate using the TCP/IP suite of protocols. Some of the most important services offered by means of it are: electronic mail, telephony (VoIP), file transfer and share, live (real-time) streaming, TV and radio broadcasts, peer-to-peer networks, mobile applications, etc. The World Wide Web is just another service that runs on the Internet, among others described above.

B. Semantic Web

The Semantic Web is a term introduced at the beginning of 2000s by none other than the inventor of the World Wide Web and founder of its Consortium (W3C), the english scholar Tim Berners-Lee. Its main goal is to automatize the processes on the current (classical) Web, allowing machines to perform tasks that can only be done by humans, such as find, interpret, process, combine information on the Web. The Semantic Web is not a separate entity of the classical Web but an extension to it, that adds new data and metadata to web documents to extend them to data that have a semantic structure. This type of extension to data allows the Web to be automatically processed by machines and manually by humans. It is used synonymously as the third generation of WWW, the Web 3.0. Fig.1 shows its architectural stack, with technologies and standards created for each layer.

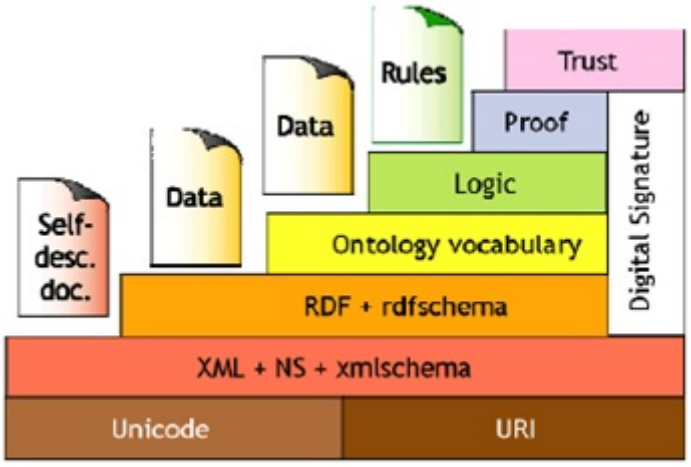

Fig. 1. Architecture of the Semantic Web (stack of technologies)

The fundamental technologies of the architecture are: RDF (Resource Description Framework - https://www.w3.org/RDF), OWL (Web Ontology Language - https://www.w3.org/OWL) and SPARQL (Simple Protocol and RDF Query Language - https://www.w3.org/TR/rdf-sparql-query/). The first two are the data formats in which data is published on the Semantic Web, and also act as ontology languages. The third is the standard query language for data on the Semantic Web. The layers that had not been yet standardized are 3 upper most: Logic, Trust and Proof. This is due to the existing challenges with which we are confronted and that falls under the umbrella of these 3 domains, the most important are:

Description Logics in Inference and Reasoning Systems and Services

Andrei Zamfira
Dept. of Computer Science
Politehnica University of Timisoara
Timisoara, Romania 300006
Email: andreizamfira@gmail.com

Horia Ciocarllie
Dept. of Computer Science
Politehnica University of Timisoara
Timisoara, Romania 300006
Email: horia.ciocarlie@cs.upt.ro

***Abstract*—In the current work we continue with our investigations into the Description Logics domain, this time with focus on the inference tasks in DL knowledge bases, the algorithms created for them (decision procedures) and the systems that implement these techniques inside their architectures. A private section will be dedicated to the complexity of reasoning results, what time an algorithm requires to solve a specific problem and how this differs between other DLs. Will be discussed the results for the important DLs and enumerated others, and also will provide the reader with references in the literature where he can find more information about this field. Will be made also a state-of-the-art about inference and reasoning in DLs where will be presented some of the most valuable works read by author in creation of this paper. The article will conclude with a section of conclusions, where will be shown the achievements of the research and will state future directions of our investigations.**

***Keywords*—Description Logic, inference task, reasoning algorithm, computational complexity, language expressivity**

I. Introduction

In this work we continue the investigation in Description Logics that we started in previous article, now we will approach the inference and reasoning aspects of DL knowledge bases. A DL system not only stores terminological axioms and assertions (facts) but also provides inference and reasoning services to make deductions about them. In general, reasoning over a knowledge base is the process of discovering implicit knowledge from the explicitly stated one.

Description Logics have distinctive logical properties. They put emphasize on the decidability of important reasoning problems, such as satisfiability of concepts or of knowledge base, provide reasoning solutions that are decidable, such as tableaux algorithms that deduce implicit knowledge from the explicit one. Highly optimized DL reasoners (e.g. FaCT++, Pellet, Racer, HermiT) proved that tableaux algorithms for expressive DLs lead to a good performance of the system even on very large knowledge bases.

This paper, as was previously stated, is meant as a literature review in which we will show off the reader this area of the domain. Will be presented the important notions and give references in the literature where it can be found more knowledge. We will present the main inference problems into a DL knowledge base and the techniques that have been created to solve them (decision procedures), each one in a private section. In another section we will discuss the complexity results for solving these problems in some particular DLs, as were demonstrated by some famous scholars such as Horrochs, Baader, Calvanese et al. Also will be made a series of analyses and comparisons regarding the influence that addition of constructors to a language have on the complexity of reasoning (i.e. the growing in expressivity) starting from the basic language ALC. Will be shown what combinations of constructors are the most inoffensive (i.e. does not change the complexity of reasoning), and what constructors generate "blows in complexity".

II. Inference Problems

Reasoning over a knowledge base is the process of discovering implicit knowledge from the explicitly stated ones.

The inference problems can be divided into two classes [27]:

a) general: involve verification of the truth value of a proposition

b) complex: built from atomic (general) ones

Let L be a description logic, K a knowledge base, C,D two concept names, and a an individual.

General inference problems are:

1) knowledge base satisfiability: K is satisfiable if it has a model (i.e. is non-contradictory)

2) concept satisfiability: a concept C is satisfiable w.r.t. K if there is a model I of K such that $C^I \neq \phi$

3) concept subsumption: $C \subseteq D$ w.r.t. K (written $K| = C \subseteq D$) if in every model I of K we have $C^I \subseteq D^I$ (all instances of C are instances of D)

4) concept equivalence: $C \equiv D$ w.r.t. K (written $K| = C \equiv D$) if they include each other w.r.t. K ($K| = C \subseteq D$ and $K| = D \subseteq C$)

5) instance checking:

- an individual i is an instance of a concept C w.r.t. K (written $K| = a : C$) if in every model I of K we have $a^I \in C^I$ (is an element of the interpretation of C)
- two individuals (a,b) are an instance of a role r w.r.t. K (written $K| = (a, b) : r$) if in every model I of K we have $(a^I, b^I) \in r^I$ (is an element of the interpretation of r)

In case of a DL that offers all boolean operators (such as ALC), all the above mentioned problems are reducible to the former (KB satisfiability) [3]. Example:

$(T, A)| = a : C$ iff $(T, A \cup \{a : \neg C\})$ is inconsistent.

These problems can be transformed into reasoning w.r.t. a TBox T, which means reasoning w.r.t. the knowledge base (T, Φ). This is referred to as *terminological reasoning*. One important property of DLs says that reasoning with TBox is not influenced by ABox, which means satisfiability w.r.t. (T,A) coincides with satisfiability w.r.t. T with condition that A is consistent (has a model). [4]

Environmental monitoring based on ontology

Cornelia Melenti
Computer Science Department
Technical University from Cluj-Napoca, Romania
e-mail: cornelia melenti@cs.utcluj.ro

Ioana Laura Magyar
Technical Department TEF
Robert BOSCH SRL, Cluj-Napoca, Romania
e-mail: magyar.ioana.laura@gmail.com

***Abstract* - This paper presents a formal description of an ecosystem based on ontological representation. First, we present a brief knowledge about general ecology and its applications and the basic theory of ontological definitions and representations. In the second we propose an environmental system ontology (structure, relations and laws), a formal description using Protégé and an example scenario of using ontology in monitoring nature reserves. In conclusion, we present the future directions of the subject.**

Keywords: environmental ontology, GIS, web services environmental monitoring

I. INTRODUCTION

A. About geo-ontology

In the last 20 years, the ontology notion became more and more used in computer science and information systems. Coming from philosophy, where it is a branch that studies the object's structure and nature, the characteristics and relationship between each department of reality (Smith, 1999), ontology gives a base for the exchange of information and an essential precondition for entity definition and relationship between entities for a certain domain. Both the entities and the relationship between them are described in a natural language, so that it becomes comprehensible for the machines as well as the humans.

In the main, the purpose of creating and using an ontology it's for the capacity of using knowledge, in a distributed way, as well as for a better reusing of them. Ontology captures general knowledge of a certain domain, which changes a little bit in time and specifies concepts and relationships with which the knowledge has to be multiplied and processed. Semantic relationship gives to the user an abstract view about the studied domain's information.

Used under a specification form and as a support for knowledge base, ontology begins to be used in web applications, as an important component of the semantic web. This is seen as an evolution of the World Wide Web where are defined the information and the services. In the semantic web, ontology creates the web's possibility for the man as well as the machine to understand and satisfy their needs, related to the web's matter, creating a universal environment for data, information and knowledge exchange.

From this project's point of view ontology is seen as a formal and explicit specification of a concept. This means that the ontology includes a set of hierarchical structured concepts that describe a certain domain. With all this the ontology can be used on a wide extent of applications and a large number of purposes. Regarding the big complexity which involves the ontology, we will discuss about the environmental ontology, as a described domain, with a goal of structuring and the possibility of extracting the specific information.

In the last decades were many progresses in developing an understandable theory regarding geographical information (Berry's geographical matrix, Berry 1964) and embracing this kind of theory in specific phenomenon and process representation and modeling. With all this, the first Geographic Information Systems (GIS) have been developed ignoring the domain's achievements and focusing more on the geographical part than on processes. Therefore, these systems implemented just a limited part of data, arriving to a new functionality forcing or coming in to ad-hoc extensions. Moreover, working alone and without any standards, different GIS producers came upon difficulties new application's integration, due to the fact that the used ontology weren't matching their data models. The expense of the standards absence begins to grow with the GIS application growing.

B. Environmental ontology

Environmental problems cover a lot of subjects, and the borders are, sometime, uncertain. Through Directive 2003/4/EC [7] the public access to environmental information offers a list of general concepts about environmental information. The air, the water, the atmosphere, the soil, the biodiversity, the substances, the energy, the radiation, the pollution, the ranges of contamination, are just few concepts regarding environment. These concepts are used by many disciplines, departments, terminology, classifications, overlaying each other and sometimes loose with each other. Considerable efforts were made for realizing common classifications and terminology of more disciplines, but this fact can be achieved just at a high level.

Even though ontology is, as a rule, independent of any programming language, it is necessary choosing a language in which can be made the description (section I.C). For exchange and ontology combination achievement, the language has to be formal. The use of a natural language for an ontology description is improper due to the fact that