Towards Security Benchmarking of Transactional Systems

Afonso Araújo Neto
CISUC, University of Coimbra
3030-290 – Coimbra – Portugal
acaneto@dei.uc.pt

***Abstract*—The multiplicity of available software and component alternatives has boosted the interest in benchmarks that allows fair selection among candidate alternatives. The huge success of performance and dependability benchmarking, however, markedly contrasts with the small advances on security benchmarking. In this paper we discuss the problems faced by security benchmarking and propose approaches for security benchmarking of transactional systems.**

1 Introduction

The importance of security in computer systems is indisputable. Most organizations today are dependent of some kind of computer installation to manage business critical activities. A central kernel of these installations is comprised of their information infrastructures, composed by one or more databases and surrounding operational applications. While simple measures may be enough to protect a standalone computer, protecting a complex system environment against the enormous number of different and unexpected threats is a very difficult, highly time consuming, and error prone task. Methods for helping system administrators in assessing and securing their installations are then of utmost importance.

Due to the increasing diversity of alternative software and components, system administrators and developers have nowadays the chance to select the software that best fit their needs based on quality attributes such as performance, usability, dependability and security [9]. The problem of performance and recovery time in databases has been largely studied in the past [17][18]. In fact, several works compare the performance and dependability capabilities of different hardware and software products. Nevertheless, security has been largely absent from previous benchmarking effort. This way, and in spite of the pertinence of having security benchmarks for database systems, the reality is that no security benchmark has been proposed so far, in a clear contrast with the benchmarking of performance and dependability.

A benchmark is a procedure that allows assessing and comparing systems (or components) according to a given characteristic (e.g., performance, availability, security)[17]. The concept of benchmarking can be summarized in three words: representativeness, usefulness, and agreement. A benchmark must be as representative as possible of a given domain but, as an abstraction of that domain, it will always be an imperfect representation of reality. However, it is useful, in the sense that its results allow making informed decisions regarding the benchmarked targets. One expected usage of a security benchmark is to compare the security characteristics of alternative systems and installations. At the same time, users must agree that the benchmark fulfills the role expected. In fact, as it is only a representation of reality, parties may disagree that it really portrays a useful and realistic view, and, thus, it would not guide its users to the same conclusions. Simply claiming that a benchmark is useful is not enough; users must actually feel that it is useful.

Databases play a central role in the information infrastructure of most organizations and it is well known that security aspects must be an everyday concern of a database administrator (DBA). When installing a database server, the DBA must consider several complex requirements, including performance, dependability and security. These requirements have strong implications in the hardware and software to be used in the database infrastructure.

The security of an information infrastructure impact all different players involved in different ways. For instance, the security precautions a low privileged user of the system has to take are completely different than of an operator responsible for backups of the system, so their points of view concerning the importance of specific security tasks might be different. The DBA usually is the person responsible for the security of the database, and the one capable of implementing security precautions and enforcing security guidelines. We assume that the DBA is the person who will benefit the most of a security benchmark in this context, being, therefore, the benchmark user.

A DBA trying to compose a secure environment has to deal with three distinct major problems, each one with its own characteristics and peculiarities. Helping in each of the following management decisions requires different tools and approaches:

1. *Decisions regarding what software will be used as the main Database Management System (DBMS)*. Today, several alternatives, paid and free, are available, and selecting one among many is a task that any new infrastructure manager has to go through. Although security is not the only factor important in this decision, no methodology to fairly help a DBA exists.

2. *Decisions regarding the configuration of the system and environment*. It is important to notice that what we call configuration is actually much more broad than a simple parameter setting of the DBMS. This actually includes also the operating system where it is installed and the surroundings of the machine (or machines) itself, along with network characteristics. This issue also applies to systems already deployed without taking benchmarks into consideration.

3. *Decisions regarding what applications may connect to his database*. Are those applications secure, or even trustworthy? Given a set of alternative applications, a security benchmark should at least give some hint at which one should be preferred. At the same time, when stuck with one insecure application, it should help improve it.

Dealing with the problems mentioned above requires a structured approach, and at least three issues have to be considered:

a) *Security characterization*: It is impossible to talk about the security of a system if we do not know its current state. Deciding an approach to evaluate the security of a system is a very big challenge, and requires different methods for each of the problems presented before.

b) *Security improvement*: Given a security state of a system, how one enhances it? What is missing? One key aspect of this issue is that this improvement should be guided and providing a prioritization is necessary in order for the enhancement be affordable. Without prioritization, the only alternative left to the DBA is either do random enhancements without proof of their utility, or correcting everything that is found to be insecure in a single step, which is usually impossible.

c) *Comparing and selecting alternatives*: In other words, given two or more security states, which one is better? This is only possible by devising useful security metrics related with these security states, which is a very complex and unsolved problem in security research.

The goal of this work is to investigate and propose solutions to most of these problems. A single method is incapable of tackling all these issues at the same time, so we have to attack the problem in several steps. In this paper we we'll discuss our approach to these aspects, and the specificities required by each of the three angles discussed before. Several of these are already accomplished, while others are still under heavy research [1][2][3][4][5][6][7][8][14].

The paper is divided as follows. Section 2 presents our approach to characterize the security of a system. Section 3 we explain how this characterization can be extended in order to provide prioritization of the improvements required by the system. Section 4 we present our approach to the problem of devising security metrics, which we replace by trust-based metrics when doing actual benchmarking. Section 5 briefly discusses the problems regarding the security of the applications that use the database.

2 Characterization of the system state

Maybe one of the hardest challenges of our work is in being able to characterize a system or environment in security terms. Several security evaluation methods have been proposed in the past. For example, the Orange Book [11] and the Common Criteria for Information Technology Security Evaluation [10] define a set of generic rules that allow developers to specify the security attributes of their products and evaluators to verify if products actually meet their claims. Another example is the red team strategy [16] which consists of a group of experts trying to hack its own computer systems to evaluate security. To the best of our knowledge, not only none of these security evaluation methods are oriented towards security comparison, but also they are too complex to be used in real installations where the administrators have limited resources.

The representativeness of a security benchmark might be expressed as the ability to identify the maximum number of potential weaknesses in the environment being assessed. The general approach used to tackle this problem is based on the analysis of a comprehensive list of *security best practices* that can be applied to the domain in question.

A security best practice is a security precaution, which can be a policy, a procedure or merely a choice for a configuration option that is used to improve the security of a particular system or scenario. For example, "*always check the type and length of an input parameter*" is a valid security best practice for the development of any web application, which can prevent, depending on the circumstances, SQL injection and buffer overflow attacks. One important aspect concerning best practices is that they can be provided in many forms (e.g., security books, software product manuals, specialized forums, etc) by a large variety of sources (e.g., vendors, developers, system administrators, academics, etc) and are usually based on field experience.

In our proposal, we evaluate the system by comparing it against a comprehensive list of security best practices advised for the domain in question. We already applied this methodology to two distinct domains, DMBS [6] and web servers [14], and confirmed that it works and is considered to be useful by the users. The characterization process is carried out using a set of tests designed to be answered by the administrator. In the end, the DBA ends up with a list of security practices that are not implemented in his system, and have the opportunity to correct the problems.

A related but distinct task is concerned with the characterization of software security properties prior to its deployment. Now the problem is different, as we are trying to help the DBA choose the software that will provide the best security benefits when deployed.

To accomplish this, we propose an approach to systematically assess and compare the security features offered by different software packages in the context of database systems. The benchmark consists in evaluating the characteristics of software packages against a comprehensive list of security concerns, which must be taken into account when deploying a database installation. This list is based on the comprehensive securities best practices lists of our previous works. Basically, each security practice is mapped into a desirable system state (*System State Goal*) that represents the state of the system when the practice is being correctly applied. That goal is used to extrapolate the functionalities and steps needed to implement the practice and mapped into the security mechanisms required for accomplishing it. The application of the benchmark results in a final metric that represents an estimation of the importance of the mechanisms present in the whole package. Additionally, and more importantly, the assessment provides a gap analysis table that helps DBAs to compare the actual security features of a set of software packages with the features that should be provided to fulfill a given set of security best practices.

We demonstrated our approach by evaluating several software packages combinations. We focused particularly on the features and mechanisms provided by Operating Systems (OS) and Database Management Systems (DBMS), which are the two main software systems in a database server. Preliminary results show that our method is quite effective in comparing the security features of different software products, allowing DBAs to make educated decisions when selecting the software for a given database installation.

3 Improving the security of deployed instances

In our benchmark, the improvement of the security of a deployed DBMS instance is more or less straightforward: the DBA should implement the missing important security best practices, and properly enforce the ones already implemented. However, how to prioritize this list, as it is impossible to accomplish all at once?

In the context of our work, we must characterize the relevance of the security practices from a usefulness perspective. In fact, we want the benchmark to be as accurate as possible, but also as representative as possible, which means that taking too much environment assumptions into account would probably make it less useful for a large number of scenarios (i.e., scenarios where those assumptions are not valid). To tackle this problem we use the consensual judgment of several experts. The expectation is that, on average, the most important practices are emphasized, even if there is no unanimity [2]. Our proposal is based on the assumption that a relative ordering established by a consensual average opinion of several security experts gives a very reliable, yet flexible, way of accounting for the importance of the security mechanisms.

4 Trust-based security benchmarking

Actually comparing the actual security level of deployed instances is notably very hard. The only known way to accomplish this is though the definition of security metrics capable of portraying *the degree to which security goals are met* in the system/environment [15]. However, the problem of security quantification is a longstanding one. In fact, Enterprise-Level Security Metrics were included in the 2005 Hard Problems List prepared by the INFOSEC Research Council, which identifies key research problems related to information security [13]. So far, no consensual security metric has been proposed [12].

Traditional security and insecurity metrics are hard to define and compute [18] because they involve making isolated estimations about the ability of an unknown individual (e.g., a hacker) to discover and maliciously exploit an unknown system characteristic (e.g., a vulnerability). In spite of the usefulness of such metrics, they are not necessarily the only way we can look at security aspects.

Consider the definition of a useful security metric: "*the degree to which security goals are met in a given system allowing an administrator to make informed decisions*". An interesting alternative would be a metric that systematizes and summarizes the knowledge and control that a particular administrator has about his own system, and how this knowledge relates to important threats his system faces. This metric would still fit the security metric requirement. Basically, the idea

is not to measure just the system characteristics, but to extend the measurement to the relationship between the system and the person (or persons) that is in charge of it (defined here as the system administrator). Such a metric would allow the administrator to become aware of the security characteristics of the system, gathering knowledge to backup decisions. This kind of metric is what we call a **trust-based metric**, in the sense that it exposes and quantifies the trustworthiness relationship between an administrator and the system he manages. We first propose the idea of trust-based metrics in [8] and explore full benchmark proposals in [1][4][5].

In these works we argue that a highly useful trust-based metric can be based on the evaluation of how much active effort the administrator puts in his system to make it more secure. Note that effort is used broadly, including not only real effort (e.g. testing an application) but also effort put on becoming aware of the state of the system (e.g. identifying that the server currently loads insecure processes). This effort can be summarized as the level of trust (or rather distrust) that can be justifiably put in a given system as not being susceptible to attacks. As an instantiation, we propose a trust-based metric called *minimum untrustworthiness* that expresses the minimum level of distrust one should put in a given system or component to act accordingly to its specification.

5 Selecting secure applications

A DBMS securely configured helps very little if the applications that connect to the database are inherently flawed. In particular, web applications today are crawled with vulnerabilities, and techniques to help administrators choosing secure options is very needed.

We are currently trying to apply our trust-based metrics concepts to evaluate and compare applications in terms of the trustworthiness they can have. We are exploring the idea in two fronts: one using the results of several vulnerability scanners in order to perform security benchmarking and another trying to look for evidences of implementation security best practices in the code of the applications [7]. Although results look promising, we are only starting the research on this area.

References

[1] Araújo Neto, A., Vieira, M., "A Trust-Based Benchmark for DBMS Configurations", 15th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC09) , Shanghai, China, 2009.

[2] Araújo Neto, A., Vieira, M., Madeira, H., "An Appraisal to Assess the Security of Database Configurations", 2nd Intl Conference on Dependability, DEPEND 2009, Greece, 2009.

[3] Araújo Neto, A. and Vieira, M. , "Appraisals based on Security Best Practices for Software Configurations", Fourth Latin-American Symposium on Dependable Computing (LADC 2009), João Pessoa, Paraíba, Brazil, September 2009.

[4] Araújo Neto, A. and Vieira, M. , "Benchmarking Untrustworthiness in DBMS configurations", Fourth Latin-American Symposium on Dependable Computing (LADC 2009), João Pessoa, Paraíba, Brazil, September 2009.

[5] Araújo Neto, A. and Vieira, M. , " Benchmarking Untrustworthiness: An Alternative to Security Measurement", International Journal of Dependable and Trustworthy Information Systems , Vol. 1 , #2 , pp. 32-54, IGI Publishing , April 2010.

[6] Araújo Neto, A. and Vieira, M. , "Towards Assessing the Security of DBMS Configurations", IEEE/IFIP Intl. Conf. on Dependable Systems and Networks (DSN 2008), Anchorage, USA, June 2008.

[7] Araújo Neto, A. and Vieira, M. , "Towards Benchmarking the Trustworthiness of Web Applications Code", 13th European Workshop on Dependable Comp. (EWDC 2011), Pisa, Italy, 2011.

[8] Araújo Neto, A. and Vieira, M. , "Untrustworthiness: A Trust-based Security Metric", 4th International Conference on Risks and Security of Internet and Systems (CRiSIS2009), France, 2009.

[9] Barbacci, Mario R. et al, "Quality Attribute Workshops (QAWs)", Third Edition, CMU/SEI-2003-TR-016, 2003.

[10] Common Criteria, "Common Criteria for Information Technology Security Evaluation: User Guide", 1999.

[11] Department of Defense, "Trusted Computer System Evaluation Criteria", 1985.

[12] Jansen, W. "*Directions in Security Metrics Research"*. NISTIR 7564. 2009.

[13] INFOSEC Research Council. "Hard Problem List." 2005.

[14] Mendes, N., Araújo Neto, A., Durães, J., Vieira, M. and Madeira, H. "Assessing and Comparing Security of Web Servers," 14th Pacific Rim International Symposium on Dep. Comp. (PRDC'08).

[15] Payne, S. C. "A Guide to Security Metrics." SANS Institute Information Security Reading Room. 2006.

[16] Sandia National Laboratories, "Information Operations Red Team and Assessments™".

[17] TPC Council, www.tpc.org.

[18] Torgerson, M. "Security Metrics for Communication Systems." 12th ICCRT Symposium, Newport, Rhode Island, 2007.

[19] Spainhower, L., Kanoun, K., "Dependability Benchmarking for Computer Systems", Wiley-IEEE Computer Society Press, 2008.