

Role Comparison Security Report – Database Server Role

Herbert H. Thompson, Ph.D.

Fabien Casteran, M.Sc.

June 2005



1990 W. New Haven Ave., Melbourne, FL 32904

Tel: (321) 308-0557 Fax (321) 308-0552

info@securityinnovation.com www.securityinnovation.com

Executive Summary

When considering which platform to deploy across an enterprise or to serve a particular role, IT decision makers have always looked at a small set of criteria, such as purchase price, compatibility with existing applications/technologies, maintainability and deployment cost. Historically, security has been conspicuously absent from that list. In many cases though, the cost to enterprises of poor security acquisition and deployment decisions has eclipsed other traditionally evaluated costs.

With this in mind, the industry urgently needs objective measures of platform security that are meaningful in a deployed context. Any meaningful security measure *must* be based on a holistic view of a system and must also consider what role that system will serve.

In early 2004, we set out to conduct critical security comparisons of several platform solutions in a variety of server roles. Our first comparison considered the web server role, one in which organizations centralize application deployment within a single server role that provides database, web serving, and application scripting functionality. In the current report we consider the more granular role of database server, where a machine must manage, store and retrieve data in a highly available way. Specifically, we compare three distinct solutions fulfilling this role: Microsoft Windows Server 2003 running Microsoft SQL Server 2000 Service Pack 3 database server, Red Hat Enterprise Linux 3.0 (RHEL 3.0) running the MySQL¹ database server and Red Hat Enterprise Linux 3.0 (RHEL 3.0) running the Oracle 10g database server.

Hard-data comparisons are made on reported vulnerabilities that affect these systems, as well as patches that are relevant to the security of the system as a whole. Specifically, we will consider the vulnerabilities for these systems that were fixed in the one year period beginning March 1st 2004 to February 28th 2005.

In our analysis we leverage the inherent modularity of Linux to consider the “minimal install” system for the MySQL database server which yields a smaller attack surface. For the Oracle case, we follow Oracle’s recommended configuration to build our server and only consider vulnerabilities that affect the configuration as deployed. For the Microsoft-based solution there are many components which are difficult or impossible to completely remove from the operating system and therefore we consider a “complete” installation, and count vulnerabilities for every application included with the Windows Server software in our analysis.

Many will remember the Slammer worm of January 2003 which was targeted towards Microsoft’s SQL Server 2000 SP2 and earlier. Threats like Slammer were a major wakeup call for IT departments to take server security and patch management seriously. In this report, we look at key metrics that aid those administrators in assessing how the

¹ Red Hat Enterprise Linux 3 servers shipped with MySQL 3.23, and this is the version under enterprise maintenance by Red Hat, so this is the version analyzed in the report. Utilizing any other version would mean customers must provide self-maintenance for the server.

operational security of three different database systems stack up over the last year. Looking at just the database applications by themselves, our study found that SQL Server 2000 had zero vulnerabilities in the one year time period, MySQL had 7 vulnerabilities and Oracle 10g had the most with 30 vulnerabilities.

When examining the full server stack needed to fulfill the database role, our study found a total of 63 such vulnerabilities for the SQL Server 2000 on Windows Server 2003 based solution compared with 116 vulnerabilities for the minimally configured MySQL on Red Hat Enterprise Linux server based solution and 207 vulnerabilities for the Red Hat Enterprise Linux server based solution running Oracle.

Additionally, when examining the “days of risk” – time between when a vulnerability is publicly disclosed to when a patch is released by the vendor for that vulnerability – we found an average of 32.0 days of risk per vulnerability for the Windows Server 2003 solution with SQL Server 2000, 61.6 days of risk per vulnerability for the Linux solution with MySQL and 38.7 days of risk for the Linux solution running Oracle.

We hope that the results of this study will provide valuable guidance to the IT manager who must make platform acquisition and deployment decisions to both maximize value and minimize security risk.

Scope of Analysis

To get a full view of Security Risk, one has to consider two important factors:

- Vulnerability of software, systems or networks (whichever is appropriate)
- Threats against those vulnerabilities

Of the two factors, our own experience leads us to believe that the latter is more difficult to quantify and predict in an objective manner. This is an exciting and open field and we strongly encourage others to consider this as an area for thoughtful research. However, given that there are research opportunities in both areas, we have chosen to try and make progress in studying and measuring the vulnerability factors first; this is a critical precursor to other threat-based metrics. We thus do not consider the threat profile in this study and instead focus on underlying system vulnerability.

As noted in our previous Web Study², we want to look at customer-focused comparative measures of security for platforms – in particular, we examine those measures that the vendors have the ability to affect and improve. This current study furthers that objective and aims to provide key insights into the security of several common platforms in database server role.

Acknowledgements

This study and our analysis were funded under a research contract from Microsoft. As part of the agreement, we have complete editorial control over all research and analysis

² Security Innovation analysis report “Role Comparison Report – Web Server Role,” March 2005, http://www.sisecure.com/pdf/windows_linux_final_study.pdf

presented in this report. We stand behind our methodology and execution of that methodology to determine objective results that will be useful to customers and security practitioners.

We encourage others to examine, thoughtfully analyze and comment on this work. Our goal was to perform an analysis based on fact (not speculation) using a transparent and meaningful methodology. The synthesis of feedback into further incarnations of our research benefits everyone, regardless of affiliation: our goal in discussion is to generate light, not heat.

In that spirit, we would like to thank Mark Cox of Red Hat for working with us to resolve discrepancies between his own vulnerability disclosure dates and those we had compiled using the methodology published in March 2005. Mark's feedback was extremely valuable and helped us refine the methodology slightly (see Appendix A) to accommodate private bug databases that are later made public and resulted in a mutually validated data set, used in all analysis within this study.

This study is the second installment in our comparative efforts of platform security under a variety of roles. When we released our first study on Web Server role comparisons in March of 2005, we received thoughtful feedback from the academic, user and analyst communities and would like to thank all of those who provided early feedback. These comments have been incorporated into this current document and we look forward to comments on this work as well.

Finally, we would like to thank Richard Ford of the Florida Institute of Technology for his co-creation of the underlying methodology used here as well his extensive contribution to the analytical underpinnings of these studies.

Introduction

Security is a serious concern for those deploying modern computing systems – so much so that the relative security of different solutions can be a major factor in choosing platforms and applications. Additionally, given product deployment lifecycles, many companies are making choices that will affect their operations for as much as the next ten to fifteen years.

In this document, we present a role-based comparison of the relative security of three different solutions satisfying the Database Server role. By concentrating on *roles*, it is possible to create server comparisons that are meaningful and centered upon customer requirements. We believe that this key aspect has been missing from previous quantitative comparisons; its inclusion provides for more meaningful comparisons of equivalent functionality.

The dynamic management, storage and retrieval of data are some of the most critical server functions in many environments. This role exists across a broad set of industries and in this paper, we assume that an organization has a high-level objective of having a high performance, highly available, robust and extensible database system and that there are multiple platforms/servers that could fulfill these requirements.

To this end, we first provide a general description of the Database Server role, and then describe a typical deployment of this role using Windows Server 2003 with Microsoft SQL Server 2000, Red Hat Enterprise Linux (RHEL) 3 with the MySQL database server and Red Hat Enterprise Linux 3 with the Oracle 10g database server. At a high level, we describe our assumption of the requirements of such a role and later use this as a roadmap for the specific implementations needed for the Windows Server 2003 and RHEL solutions. Security comparisons – both quantitative and qualitative – are then made of this role in these configurations.

Finally, we will conclude by reviewing the results and what actions vendors could take to improve their security as measured by this methodology.

Database Server Role Description/Definition

The need for highly available and reliable data management is critical for today's organizations. Database servers fill this need by providing applications access to information in an organized and useful form. To meet the demands of business, such servers must fit a broad spectrum of user requirements. Typically, database servers interact with both internet and intranet facing components and therefore the need for security and fault tolerance is high. In addition to the host operating system, for the database server role we need a database server application and software that supports the encryption of transmitted data.

- SSL/Encryption support

We believe that for security and privacy purposes, a typical database server role will have to have a solid implementation of encryption technology; in particular, support should exist for accepted standards such as SSL and TLS for data transmission.

This is critical for protecting sensitive data during transfer between client and server or between application tiers.

- The database server

The database server is responsible for the dynamic storage and retrieval of information. The server must provide a broad range of interfaces to interact with other servers such as application servers, file/print servers and other systems. It must, for example, be equipped to handle an ODBC connection and receive and respond to SQL queries.

- A Server Platform

Examining a Database server in isolation from an underlying operating system would be ignoring the foundation that underpins the database role. A perfectly secure database installed on an insecure platform will not be able to maintain its security. We therefore must consider overall *solution security* in our analysis and thus include platform issues. In this study we specifically look at Windows Server 2003 and Red Hat Enterprise Linux 3.

Platforms Compared

In our analysis we wanted to choose platforms to compare what would be most meaningful to customers. Given the high level of interest in Windows versus Linux, we chose the most recent version of Windows server software, Microsoft Windows Server 2003 and Red Hat Enterprise Linux 3³⁴. We note that there are many different distributions of Linux that we could have selected, but recent analyst reports indicate that business customers are largely selecting to deploy Red Hat or SuSE in production environments. Thus, we selected Red Hat for these tests, as it is the current leading distribution⁵. Furthermore, due to its strong position in the enterprise solutions market, Red Hat is arguably the best representative of the open source distributors and the preferred candidate for our analysis.

We originally applied this methodology to the Web Server role in a report released in March 2004 focusing on roles built on top of Windows Server 2003 and Red Hat Enterprise Linux. The current report focuses on results on these same platforms in the Database Server role. While both reports look at the same platforms, they follow a basic methodology which is generic and could be applied to other products, vendors and server roles.

³ Although we look at the ES version of Red Hat Enterprise Linux 3, the packages installed in the AS version are very similar and thus we expect the results with that platform to be comparable.

⁵ Source: IDC report “Worldwide Linux Operating Environments 2004-2008 Forecast and Analysis: Enterprise Products Pave the Way to the Future”, December 2004.

⁴ Since this study has been completed, Red Hat has released Red Hat Enterprise Linux 4. See “Analysis Note” below for more details.

Red Hat Enterprise Linux ES 3

Red Hat is the world's leading provider of open source solutions to the enterprise⁵. Because of its strong position in the Linux enterprise solutions market, Red Hat is the best representative of the open source distributors and the preferred candidate for this analysis.

Solutions include Red Hat Enterprise Linux ES 3, which is the enterprise solution server that Red Hat recommends for small to medium web configurations. After a successful IPO in 1999, Red Hat enjoys strong brand name recognition, and is considered by many to be the most recognized name in the open-source OS distributions. Additionally, Red Hat Enterprise Linux is widely deployed in the web server role, which makes it an obvious and meaningful candidate for analysis.

Red Hat Enterprise Linux 3 uses a hybrid kernel approach with features from the Linux 2.6 kernel back-ported for use with the stable Linux 2.4 kernel. RHEL ES 3 includes support for numerous architectures and provides both the features and the support required by large organizations.

In order to build a functional database server using RHEL, we must first choose different components that fulfill the functional requirements detailed in the next section of this report. Below, we list these primary components, along with a short justification of their selection.

Analysis NOTE: Red Hat shipped Red Hat Enterprise Linux 4 (RHEL4) as a successor to RHEL3 in January of 2005, but under their enterprise agreements will be supporting RHEL3 for several more years. Given that we wanted to study at least 12 months of data, RHEL4 was not an option.

Due to the inclusion of the 2.6 SELinux kernel in RHEL4, some speculation may occur that the "enhanced security" would significantly change the results of the analysis in this study, if it was possible to use RHEL4.

"Security Enhanced" for the 2.6 kernel refers to a new role-based access control capability and not to code quality, so it is worth taking a little space to see how vulnerabilities and patching is trending in RHEL 4.

According to the Red Hat web site, between February 15 and the end of April 30, 2005, Red Hat has issued:

- 36 security advisories for Red Hat Enterprise Linux 3 AS, and
- 61 security advisories for Red Hat Enterprise Linux 4 AS

Though these advisories represent all packages for the Red Hat Enterprise Linux releases, and not just the minimal package set we study, the data from this limited timeframe does not seem to indicate drastic security vulnerability improvement for RHEL4.

Additionally, the three vulnerabilities fixed in MySQL during the period after RHEL4 released (and within our study period) affected MySQL as shipped with both RHEL3 and RHEL4.

Red Hat Enterprise Linux 3 – MySQL Database Server

While there are several potential choices for an open source database server such as PostgreSQL and MySQL, we chose to look at the MySQL server because of its popularity in the marketplace. In the results section of this study however, we examine vulnerability count differences between MySQL and PostgreSQL.

The following are the applications that are required to fulfill the database server role on the Red Hat platform with MySQL:

- Database server: MySQL
Over 4 million active MySQL installations worldwide make the MySQL database server the most popular open source database⁶. Users of the database server can choose to use it under the GNU General Public License or under a commercial license. MySQL is lightweight and can handle multiple connections in a fast and reliable way. It lacks some of the features other database servers (like PostgreSQL) support such as views and sub-queries, however its performance is superior to most open source competitors. MySQL is also the database component of the so called “LAMP”⁷ configuration, the set of free software programs commonly used together to run dynamic web sites and ships as a add-on component of RHEL ES 3.

Red Hat Enterprise Linux 3 – Oracle Database Server

Oracle has established itself as a leading provider of database technologies in the server marketplace on the Linux platform⁸. A recent report by technology analyst firm IDC⁹ estimates that Oracle holds a 41.3% market share for relational database. Oracle Corporation is a major commercial software vendor with revenues in excess of \$10 billion in FY04.

For this configuration, we install Oracle according to their deployment guidelines¹⁰ and then analyze the resulting system. The following are the applications that are required to fulfill the database server role on the Red Hat platform with Oracle:

- Database server: Oracle 10g

⁶ BZ Media survey, “Database and Data Access, Integration and Reporting Study”, <http://www.bzmedia.com/bzresearch/5914.htm>, May 2004.

⁷ LAMP, or “Linux, Apache, MySQL, Perl/PHP/Python” is considered to be the “standard” open-source configuration of a dynamic web server.

⁸ Oracle corporation, http://www.oracle.com/database/feature_db_dbleadership.html

⁹ IDC Corporation, “Robust Recovery in Worldwide RDBMS Market, But Results Tempered by Currency Environment, IDC Reveals,” <http://www.idc.com/getdoc.jsp?containerId=prUS00089505>

¹⁰ Oracle corporation, “Installing Oracle Database 10g on Linux x86,” http://www.oracle.com/technology/pub/articles/smiley_10gdb_install.html

With its latest 10g product release, Oracle provides an extensible and robust platform with support for management and configuration to serve data to meet a diverse set of user needs.

Microsoft Windows Server 2003

The Windows Server 2003 is a server operating system that can assume several different roles including the database server role. The Windows Server 2003 platform is well supported by an established company, and leverages the strong Microsoft Windows brand name. As such, it provides the stability, support and manageability required for commercial deployment.

For the Microsoft Windows Server 2003 platform under the database server role the following configuration is assumed:

- Database server: SQL Server 2000 Service Pack 3
SQL Server 2000 is Microsoft's enterprise database server. It includes all the features one would expect from an enterprise database server, in addition to functionality designed to increase its usability and performance. We will be looking at SQL Server 2000 SP3, the release that was available when Windows Server 2003 shipped in 2003.

Requirements

Comparing the security of a server based on an analysis that considers all security issues reported on all applications that *might be installed* on this operating system is neither realistic nor helpful to decision makers. In reality, very few users have all of the server components installed or running on their systems. Thus, the focus of this report is to compare the security of systems configured in the database server role, as this is much more representative of "real world" scenarios.

One of the security strengths frequently cited with respect to Linux is that its modularity allows a true "minimal build" of a server role, thereby reducing its effective attack surface and making it more secure. Administrators however must operate within manageability constraints and also deploy in a manner that is still supported by vendors. For the Oracle on Linux configuration, we install both the operating system (RHEL 3) and the database server (Oracle 10g) according to the exact method specified by Oracle¹¹ which is detailed in Appendix B of this report. For the MySQL on Linux build, we leverage the modularity of Linux to provide a minimal installation (and thus minimal attack surface) for this server.

On the SQL Server 2000 and Microsoft Windows Server 2003 build, components like Internet Explorer will be present on the workloads since they cannot be easily uninstalled from the database server role. In our research, we therefore assume that a customer would

¹¹ http://www.oracle.com/technology/pub/articles/smiley_10gdb_install.html

need to patch *any* issue that is present in Microsoft's server software. So, while Internet Explorer issues will count against the Microsoft build, similar vulnerabilities affecting Mozilla will not be counted against the MySQL on RHEL 3 build. In our study, we deploy the software and physically validate configurations.

To assess security we will consider both quantitative and qualitative metrics. Our analysis will then validate the three solutions on both fronts, as well as examine the database applications alone.

Quantitative Metrics Description & Introduction

Historically, platform security comparisons have been made on quantitative and readily available data. Such comparisons have frequently made a simple count of "security bulletins" or "security advisories" issued by vendors.

While advisory counts are popular, there is little evidence to support their usefulness in making strategic security-aware deployment decisions, since vendors control how many vulnerabilities might be addressed by a single security advisory. These simple bulletin counts therefore may not represent the underlying security quality of the products very well, unless extra care is made to assure vendor behavior is similar. For example, SuSE Enterprise Linux and Red Hat Enterprise Linux have a high degree of correlation between components. However, though both fix a similar set of core (e.g. Linux kernel, X Windows, hardware drivers, rsync) component vulnerabilities, the number of SuSE security advisories is significantly lower due to their use of summary advisories. If one were to carry out a simple analysis of advisory count, it would paint a better – but ultimately misleading – picture of the relative security of SuSE with respect to Red Hat. In reality, the number of vulnerabilities fixed by each vendor is of the same order of magnitude, but Red Hat is more granular (and one might argue transparent) in its release of security advisories.

Instead, we take a role-based approach to measuring security based on likely deployed configurations. For quantitative data, this approach means only considering those vulnerabilities and patches that apply to the deployed role. For instance, we shall not consider a vulnerability reported in a component that is not installed in our functioning database server solution.

Time Period

The methodology used for this comparison could be applied to any fixed time period. For our database server role comparison, we will consider only those vulnerabilities the solution vendor (Microsoft, Oracle or Red Hat) has released a fix for between March 1, 2004 and February 28, 2005, spanning one calendar year of deployment. We will therefore include vulnerabilities disclosed before March 1, 2003 if and only if they were fixed during this period. Similarly, we will not consider vulnerabilities *disclosed* within this one year period but fixed after Feb 28, 2005. One could select different time periods in the future and repeat this study using the new time periods and begin to study trends as well. While our goal was to apply metrics over a one year period, these particular dates were chosen because they represented the most current information that could be analyzed at the time of our study.

Qualitative Metrics Description & Introduction

Beyond patches and vulnerabilities, there are “softer” qualities of security that are difficult to quantify but undeniably impact deployed security. Qualities like security lifecycle support, bulletin descriptiveness, default security features and the like all have a direct impact on deployed role security. In our report, we will outline how the two solutions stack up on these criteria for the convenience of readers.

Assumptions & Rules

As indicated previously, we analyze three basic database server configurations, one on Microsoft Windows Server and two on Red Hat Enterprise Linux.

- **MySQL on Red Hat Enterprise Linux ES 3** For this configuration, we consider a minimal install where the server is specifically tailored to serving the role by only installing the minimum set of components when building the system. The minimal install scenario provides for a reduced attack surface for the server, and makes the assumption that an administrator who prioritizes security above other criteria deploys the server.
- **Oracle 10g on Red Hat Enterprise Linux ES 3** Oracle provides detailed installation recommendations for users to install Oracle 10g on RHEL 3. While additional steps could be taken to harden the server or remove certain components, we followed vendor recommended procedures as this is likely to be the most common scenario and within the support specifications of Oracle.
- **SQL Server 2000 on Windows Server 2003** As Microsoft’s design does not make it easy to remove some additional components from the Server operating system, our system build and analysis includes all components that ship with Windows Server 2003.

Beyond installation, the following is a list of user concerns and needs that we assume for this analysis:

- The user requires the features, trust, support and professional maintenance provided by a trusted software distributor. For example, in the case of an open-source solution like Red Hat Enterprise Linux, we assume that, in the interest of manageability and its associated cost, users will only install versions of the OS components blessed and released by the OS vendor, so that their support contract remains valid.
- The database server is expected to accommodate a wide range of component, application and platform interfaces, and provide data in a high performance, reliable and robust way.
- The different components of the database server should be compatible and easily integrated.
- The database management system must be able to execute queries, simple and complex, in a fast and reliable way.

When considering the relative security of different solutions, it is important not just to consider *what* is installed, but also *how* it is installed. Thus, the default security features – essentially, the *context* in which a role is deployed – are important when considering the long-term security viability of a solution.

Contextual information that is important with respect to security includes:

- Open ports in the default/configured deployment
- Users (and their privileges)
- Other applications that may modify behavior with respect to vulnerabilities
- Technology that mitigates the vulnerability of a system (e.g. buffer overrun protection)
- Environmental considerations (such as “a server satisfying this role is usually located behind a firewall”)
- General attack surface considerations such as privilege level of exposed services

Additional assumptions on Quantitative Data

The quantitative data available is related to vulnerabilities, patches and patch quality. This information can be obtained from public bug reporting lists and from the vendors issuing the patches. While these results represent only the vulnerability dimension of security risk, they do provide insight into the aspects of security quality that are under the control of the vendor – code security quality and security response. These metrics, however, must be considered in combination with several other important qualitative factors when choosing a platform based upon cost of security maintenance and likelihood of security breach.

In order to make an unbiased comparison between two platforms, the set of assumptions used in gathering the data is crucial. This information is also essential to making the experiment reproducible. The following is the set of assumptions that was used to gather the vulnerability information and patch information.

- i. It is assumed that Red Hat customers only install patches released by Red Hat or Oracle (in the case of the Oracle configuration) and are taking other similar steps to ensure they comply with their maintenance contract. Similarly, Windows customers only utilize fixes released by Microsoft.
- ii. All the fixes released by Red Hat, Oracle and Microsoft pertaining to the systems will be recorded, along with the applications to which the patches pertain. For each fix, the vulnerabilities addressed will be entered using the Mitre vulnerability identifier¹² (e.g. CVE-2004-0079), if one has been assigned.

¹² The Common Vulnerabilities and Exposures (CVE) database is a widely accepted standard for identifying specific vulnerabilities. CVE is available online at <http://www.cve.mitre.org>. Also see section “Mitre CVE List” below.

- iii. For each vulnerability, we consider its severity to be the severity rating assigned by the National Institute of Standards (NIST) ICAT vulnerability rating system. While vendors have their own rating systems for severity, these schemes are not necessarily comparable and we thus use this independent source to facilitate meaningful cross-vendor comparisons.
- iv. When an application is installed by default, all updated versions will be considered to be default applications as well.
- v. For purposes of the time period we study, we assume systems are fully patched up to the date of the study. For example, in looking at the time period from March 1 2004 to February 28 2005, we assume all patches from the start date of our period had already been deployed on both systems.
- vi. The “first public” or disclosure date for a vulnerability is the date at which the vulnerability was first released on a public list or web site (Bugtraq, Red Hat, Microsoft, Full-disclosure, k-otik, etc.) devoted to security, or a publicly accessible list of bugs or problems posted to the home site of a package or its mailing list. We do not consider discussions on the Linux “vendor-sec” alias as public. We also do not consider non-public discussions on bugzilla as a public disclosure.
- vii. Dates for patches are based on the release date for the distribution of interest.
- viii. Release dates for a vulnerability patch or fix are specific to a distribution/architecture. If a fix for a component (e.g. `libpng`) is released on 01/01/1970 for a certain Linux distribution (e.g. Gentoo Linux) and a fix for the same issue is released for Red Hat on 01/10/1970, the release date for the fix on Red Hat will be 01/10/1970. This is not applicable for the Windows platform.
- ix. For past issues, the release date for a patch is the first published vendor report that includes the patch for the applicable platform for which the patch fully fixed the vulnerability. If the patch had to be re-issued to address some portion of the security issue, the later date is used.
- x. Documentation packages will not be entered. Applicable only for the Red Hat platform.
- xi. 80x86 is the only architecture considered for the Red Hat platform.
- xii. It is assumed that patches are installed in the order of release.
- xiii. Functional bugs are only entered as vulnerabilities in the event that patches introduce them.
- xiv. If a vulnerability has a single unique ID but is fixed in *different packages* through the application of *different patches*, such vulnerabilities shall be counted as the number of patches that apply to the installed components. For example, if a vulnerability was assigned a single CAN ID but affected two separate applications, *a* and *b*, and a patch was released separately for *a* and *b*, these will count as two vulnerabilities. While there are only a handful of these incidents in our data analysis, we believe this to be the most prudent treatment given that

multiple patches and multiple points of exposure speak directly to user security pain.

- xv. If the product vendor does not actively participate in the Mitre CVE program, we consider the vendor's own enumeration of vulnerabilities as a count of the vulnerabilities in that application. Specifically, we consider Oracle vulnerabilities unique as enumerated by the Oracle security bulletins. Given that such vulnerabilities will not have an independent severity rating (ICAT or CERT) we consider all such issues to have a rating of "none" on the ICAT severity scale.

In order to compile the data in a single centralized location we created a database that was populated using different sources and consolidated using CVE identification names where available and vendor identifiers where no CVE name exists. By creating such a database, we have the ability to easily query exploits, vulnerabilities and patches based upon a number of different criteria, such as vulnerability implication, role or days of risk.

Mitre CVE List

In our analysis we frequently refer to the CVE or CAN identifier of a vulnerability. CVE stands for Common Vulnerabilities and Exposures and is a taxonomy that attempts to standardize the naming of all publicly known vulnerabilities and exposures.

Initially, vulnerabilities are assigned a candidate number (CAN). These candidates are then examined by CVE's Editorial Board, made up of industry experts, where a decision is made for their inclusion in CVE. CVE is maintained by the Mitre Corporation, a not-for-profit organization that performs independent research and analysis for the U.S. Government. In our analysis, we refer to a vulnerability as distinct if it has its own CVE or CAN identifier. In the case of vendors that do not actively participate in the CVE program, we are forced to use the vendors own enumeration of security vulnerabilities as a count.

While Microsoft and Red Hat are active participants in the CVE program, Oracle is not. Our count of Oracle vulnerabilities is thus based on their own vulnerability numbering system which is tied to specific security alerts they release. We believe that using these identifiers still gives us an accurate count of vulnerabilities but it makes the process of directly tying a non-vendor public disclosure to a vulnerability nebulous. We thus take the conservative approach of counting these vulnerabilities as having 0 days of risk, which in the extreme case may bias days of risk calculations in the favor of Oracle.

NIST ICAT Severity Ratings

One of the most hotly debated topics in security is the severity of impact a particular vulnerability can have. The National Institute of Standards has introduced the ICAT Metabase, which contains information about known vulnerabilities and uses CVE identifiers to catalog its entries. One interesting aspect of ICAT is the severity rating it assigns to vulnerabilities. ICAT ratings are a generally accepted and objective way for system administrators and IT professionals to gauge the impact of a vulnerability on a typical system. Although ICAT severity ratings do not offer contextual guidance for how severe a particular vulnerability is in a certain context, they do provide an objective way to classify vulnerabilities. In this capacity, we use ICAT severity ratings in our analysis to

group vulnerabilities into classes. ICAT uses three broad severity classes for vulnerabilities: High, Medium and Low as defined below¹³:

A vulnerability is “high severity” if:

- it allows a remote attacker to violate the security protection of a system (i.e. gain some sort of user or root account),
- it allows a local attack that gains complete control of a system,
- it is important enough to have an associated CERT/CC advisory.

A vulnerability is “medium severity” if:

- it does not meet the definition of either “high” or “low” severity.

A vulnerability is “low severity” if:

- the vulnerability does not typically yield valuable information or control over a system but instead gives the attacker knowledge that may help the attacker find and exploit other vulnerabilities.
- we feel that the vulnerability is inconsequential for most organizations.

Unrated vulnerabilities – While ICAT contains severity ratings for the majority of the vulnerabilities in CVE, there are several vulnerabilities that remain unrated. Ratings are continuously updated on the site and this study’s rating information is current as of January 27th, 2005. When evaluating the statistics presented later in this report referring to severity, we encourage you to treat vulnerabilities with a rating of “Not Known” with extreme caution as they have the potential to be high severity. In the case of Oracle, as there is no direct mapping of CVE names to individual vulnerabilities,¹⁴ and given that ICAT indexes its database by CVEs, we consider all Oracle vulnerabilities to have a rating of “Not Known”.

A word of caution on ratings – When examining vulnerabilities, there is a tendency to ignore or devalue those rated as “Medium” or “Low”. This tendency is a mistake, as we have seen several attacks that have exploited several lower rated vulnerabilities to gain complete remote control over a machine. The synergistic combination of vulnerabilities that are rated low and medium can lead to an exposure of the highest severity. Another issue is the contextual severity of a vulnerability. Rating systems like ICAT assign severity labels to vulnerabilities based on their potential impact to a system or application in isolation. These ratings offer minimal insight into the impact of *one* vulnerability on a deployed solution. For example, protections like host-based firewalls may mean that a

¹³ Information is taken from the official ICAT documentation found at http://icat.nist.gov/icat_documentation.htm

¹⁴ Oracle has published a mapping of CVE names that were assigned externally to some of the vulnerabilities listed in its security alerts at http://www.oracle.com/technology/deploy/security/pdf/public_vuln_to_advisory_mapping.html. These CVE names however only represent a subset of the problems and there is no direct mapping of CVE names to specific vulnerabilities enumerated in the bulletins.

particular system is not vulnerable to certain exposures that are rated as “high”. While contextual ratings can be used for patch deployment prioritization in an organization, it is important to remember that configurations are constantly in flux and contextual protection is only temporary. Vulnerabilities must be fixed at their root cause to truly limit exposure in the long run. Another aspect to consider is that a single vulnerability may apply to multiple systems (such as multiple versions of Windows or multiple distributions of Linux) and that it may have a different contextual severity for that particular version. In this respect, CAN-based severity can be a fairly blunt instrument for administrators to dissect the impact of a vulnerability on a specific application/OS version or their particular deployment of that system.

Analysis NOTE: After we published our Web Server Role study, one question we received was why we didn’t do severity analysis by vendor rating systems, rather than ICAT severity ratings. There are several reasons that ICAT is the better choice for analysis by severity.

First, Red Hat did not provide severity ratings during the entire time period studied for the first study (the full year 2004), so it would not have been possible. Red Hat introduced severity ratings to their advisories in February 2005.

Second, though Microsoft and Red Hat severity rating labels appear similar, it is not clear that they are exactly comparable or that they are applied in the same way.

Finally, we want our methodology to handle other vendors. Even if the Microsoft and Red Hat rating systems were identical, Oracle doesn’t use it, nor does Apple, or Novell, or anyone else.

In summary, we used ICAT because we need a rating that is objective and applies across vendors and products.

Another interesting question we received during early methodology review was “why didn’t you use the CERT severity metric?” Basically, the CERT severity metric is subjective and would change almost daily based upon their definition. For full details, see our note in the Web Role report¹.

¹ Security Innovation analysis report “Role Comparison Report – Web Server Role,” March 2005,

Analysis of Database Server Role

The following section describes in detail the steps taken to configure the different roles. They are presented in order for our data and results to be reproducible by a third party.

MySQL on Red Hat Enterprise Linux

Overview of installation

Installation of the Red Hat Enterprise Linux platform is wizard driven and easily executed. To achieve the minimal installation for the database server role with MySQL, we deselected all options from the installation

packages and then manually selected only the 'mysql' package group for installation. This represents the minimum installation of Red Hat configurable through major installation wizard options, which allows the server to function in the Database Server role. During installation, the boot loader password is off by default.

Mysql-server is not actually on the RHEL 3 CDs, so we downloaded this module and installed it after completing the rest of the installation.

Notable security features

During platform installation, a firewall is installed and does not let any traffic into the system by default. The firewall is a simple ingress firewall, blocking incoming traffic to the system, but doing nothing for egress. The `up2date` software, which automatically downloads updates for the system, is installed by default. During installation, the wizard asks which type of traffic should be unblocked by the firewall.

Oracle 10g on Red Hat Enterprise Linux

Overview of installation

Installation of the Red Hat Enterprise Linux platform is wizard driven and user friendly. Oracle has laid out precise recommended installation guidelines for 10g on Red Hat Enterprise Linux 3 which are detailed in Appendix B of this report. It is interesting to note that the resulting system has many more add-on OS components than the MySQL build. The Oracle system for example includes X Windows (Gnome), Mozilla and several other additional packages.

Notable security features

During platform installation, a firewall is installed and does not let any traffic into the system by default. Oracle's installation guidelines recommend turning the firewall off during the installation procedure for testing purposes. The `up2date` software, which automatically downloads updates for the system, is installed by default. While `up2date` manages patches that are supported from Red Hat, administrators must still manage updates released for Oracle.

SQL Server 2000 on Windows Server 2003:

Overview of installation

Installation of the Windows Server 2003 is wizard driven and very straightforward. The default configuration does not install superfluous applications. At the end of the installation, another wizard automatically prompts the user for configuration of a role for the server.

Notable security features

The firewall is installed, but must be enabled, and then does not let any traffic into the system. SQL Server 2000 communicates over TCP/IP using the sockets network library. The Windows automatic update service is installed and running by default. As of the date of publication of this report, this tool now manages updates to applications installed on top of Windows Server 2003, including SQL Server 2000.

Vulnerability Count Results

Database Software Only

Before looking at the full function database servers, let's first examine the database software alone.

SQL Server 2000 SP3

In the 12 months studied, SQL Server 2000 contributed zero vulnerabilities to the database server role. While SQL Server 2000 SP3 did have some vulnerabilities prior to the 12 month period, examining the history of the product, we found it to have the lowest discovery rate for vulnerabilities of the databases studied.

Since SQL Server 2000 SP3 shipped in January of 2003, there have been a total of five vulnerabilities affecting the database software through February 2005, which averages out to 0.2 vulnerabilities per month.

MySQL 3.23.58

In the 12 months studied, MySQL contributed seven vulnerabilities to the database server role, which also represents the total found through the end of February 2005 since RHEL 3 shipped in October 23, 2003. This result averages out to about 0.4375 vulnerabilities per month, double the SQL Server 2000 SP3 discovery rate.

PostgreSQL

Although PostgreSQL is not a part of any of the three server builds analyzed, as a close competitor to MySQL in the open source world it is interesting to look at its potential impact had we chosen it over MySQL. In the 12 months studied, PostgreSQL had five vulnerabilities patched by Red Hat, and six vulnerabilities total through the end of February 2005 since RHEL 3 shipped in October 23, 2003. The one year result averages out to about 0.417 vulnerabilities per month, very similar to the MySQL rate.

Oracle 10g

In the 12 months studied, Oracle 10g contributed 30 vulnerabilities to the database server role, or about 2.5 per month, which also represents the total found through the end of February 2005 since Oracle 10g shipped in February 2004. This lifespan result averages out to about 2.3 vulnerabilities per month, a discovery rate quite a bit higher than either MySQL or SQL Server 2000.

There were some additional challenges related to analyzing the Oracle 10g vulnerabilities, when compared with either Red Hat or Microsoft. The 30 vulnerabilities fixed in Oracle 10g were identified in security alerts (see them at <http://www.oracle.com/technology/deploy/security/alerts.htm>) from Oracle as follows:

- Alert 68, Oracle Security Update identified vulnerabilities that affected 10g as (using the Oracle notation) DB01, DB02, DB05, DB08, DB09, DB10, DB11, DB12, DB13, DB14, DB15, DB16, DB20, DB22, DB24, DB25, DB26, DB27, and DB28 for a total of 19 vulnerabilities addressed.
- Critical Patch Update, January 2005 identified vulnerabilities that affected 10g as DB03, DB06, DB07, DB08, DB09, DB11, DB12, DB13, DB14, DB15, and DB16 for a total of 11 vulnerabilities addressed.

These alerts do not provide information about which CVE identifiers relate to each of the Oracle vulnerability identifiers, however, Oracle does provide a mapping document at http://www.oracle.com/technology/deploy/security/pdf/public_vuln_to_advisory_mapping.html. This document maps the CVE identifiers only to the Oracle Alert documents and not to individual vulnerabilities and only maps a total of ten CVEs out of the 30 identified vulnerabilities fixed in Oracle 10g.

After extensive review, we can't tell how the CVEs map to the individual DBNN identifiers in the Alerts. It appears that some finders separately publish details on Bugtraq or other sources after Oracle issues a patch, which results in CVE ids being assigned, but for the other issues, this has not happened. For purposes of analysis in the study then, we make the following assumptions:

- The DBNN vulnerabilities that do not have a CVE identifier were either internally found by Oracle or reported under responsible disclosure to Oracle for fixing, since we can't find other disclosure discussions about them.
- The vulnerabilities without a CVE identifier will be assumed to be disclosed on the date that Oracle issued their Alert, thus contributing zero days-of-risk.

This seems like an area where Oracle could make some improvements for the sake of customers by participating in the Mitre CVE process. Without a CVE identifier assigned, commonly used Intrusion Detection and Scanning products that are CVE compliant will not necessarily find and report the vulnerabilities.

On the other hand, this bit of obscurity does seem to make it harder for attackers as well, since so little detail about the vulnerabilities has been disclosed.

Database Server Role – Server OS and Database Software

The following table summarizes our findings with respect to vulnerability counts for the analyzed installation:

Severity	Windows Server 2003/SQL Server 2000	RHEL ES 3/Oracle 10g	RHEL ES 3/MySQL 3.23.58
High	27	73	41
Medium	18	63	49
Low	0	10	8
Not Known	18	61	18
Total	63	207	116

Table 1: Vulnerability counts for the three solutions compared

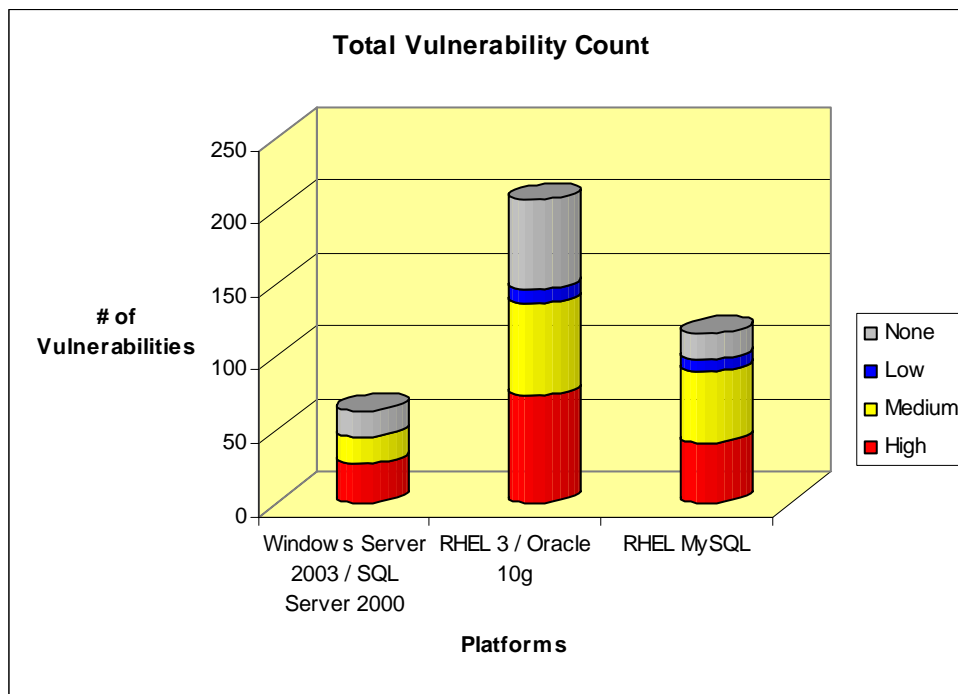


Figure 1: Vulnerability counts for 2004 for the three configurations considered.

This table includes vulnerabilities of all software that is installed on the platform by the installation procedures outlined in the previous section, Figure 1 represents these results graphically. In the case of the MySQL RHEL solution, we heavily leverage the modularity of Linux to create a system with minimally installed components. Oracle’s suggested installation procedures on the other hand favors feature richness by requiring a graphical windowing interface and suggesting the installation of Red Hat packages like “Developer Tools” and others which expand the functionality of the server yet increase the attack surface of the system. In the SQL Server 2000 on Windows Server 2003 case, we consider the complete installation of all components that ship with the platform.

The Windows solution came out ahead with 63 vulnerabilities compared with 116 on the MySQL RHEL system and 207 on the Oracle RHEL system. For the Windows case we assumed a full install of the base operating system which means we counted all vulnerabilities for Windows Server components whether it was actually installed or not. For the other cases, we only count those vulnerabilities which affect installed packages on those systems.

A close look at the data reveals that Internet Explorer had more vulnerabilities than any of the other Windows components in our study period, accounting for 16 of the 63 fixed security issues on the SQL Server 2000 on Windows Server 2003 solution. Microsoft may benefit from an attack surface perspective by making its future server operating systems more modular so that users can remove these components if so desired. Considering that nearly 30% of the vulnerabilities fixed in our one year analysis period on Windows Server 2003 were Internet Explorer related, giving administrators the option to remove such components would reduce the vulnerability count for Windows Server 2003 even further.

For the two Linux-based builds, the Kernel was the biggest contributor to the total vulnerability count with 38 vulnerabilities reported in our study period affecting both the MySQL on RHEL solution (38 of 116 vulnerabilities were in the kernel) and the Oracle on RHEL solution (38 of the 207 vulnerabilities were in the kernel). On the Oracle build, after the Linux kernel, the Oracle 10g software contributed the next highest number of vulnerabilities with 30 during the time studied.

It should be noted that Red Hat Enterprise Linux ES 3 is modular in its installation procedure and can function without some of the components which are “core” in the Windows system. This is potentially significant because fewer installed components means a reduced attack surface and fewer patches that may need to be applied. It should also be noted that the attack surface of the Oracle build can also be reduced by removing some of the components suggested in their installation procedures.

The vulnerability analysis shows an advantage to the Windows Server 2003 platform when considering several important metrics. The total vulnerability counts for the Windows platform are significantly lower than those for both the Oracle and MySQL solutions built on RHEL ES 3; similarly, when looking at just the high severity bugs, counts are clearly in Microsoft’s favor.

Without a rigorous methodology, one might expect the minimal MySQL on Red Hat Enterprise Linux build to have the least software vulnerabilities, followed by Oracle on Red Hat Enterprise Linux and SQL Server 2000 on Windows Server 2003. However, the actual results seem to demonstrate that the Microsoft Security Development Lifecycle (SDL) is resulting in software with a lower vulnerability incidence rate, contributing less vulnerabilities and less patching for customers than either Oracle or the Open Source alternatives studied.

Though the focus of the study is the 3 configurations described, we anticipate that some readers may also ask two other questions:

- What about Oracle on a minimal Linux, similar to the MySQL build?
- What about Oracle when installed on Windows Server 2003, rather than Linux?

The chart Figure 2 shows variations of the builds that we studied and shows the answer to these questions. The database server builds with the least vulnerabilities to the one with the most vulnerabilities for the 12 month period studied are, in order:

1. (least) SQL Server 2000 SP3 on Windows Server 2003
2. Oracle on Windows Server 2003
3. MySQL on Red Hat Enterprise Linux 3 ES (minimum build)
4. Oracle on Red Hat Enterprise Linux 3 ES (minimum build)¹⁵
5. (most) Oracle on Red Hat Enterprise Linux 3 ES (Oracle installation instructions)

These results show that the Oracle on Windows solution had less vulnerabilities than the Oracle on Linux solution, but that SQL Server on Windows had the least vulnerabilities by a significant margin.

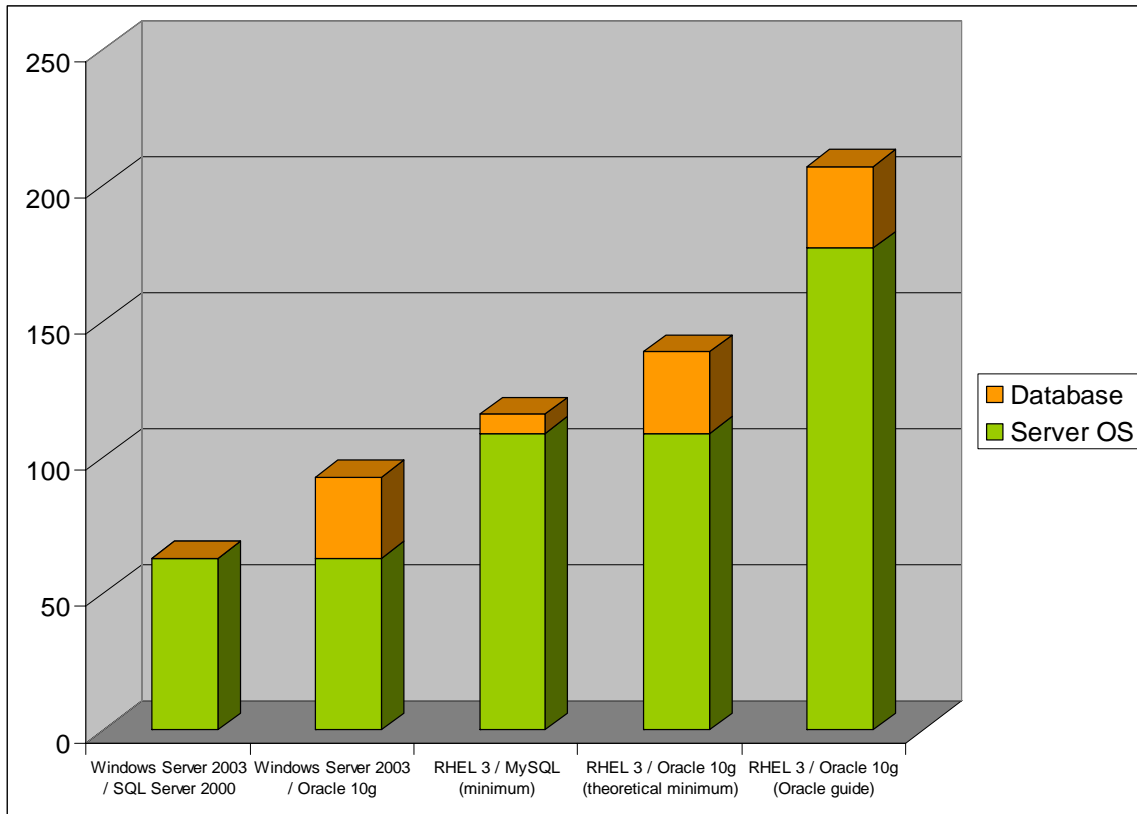


Figure 2: Database & Platform Variations

¹⁵ The numbers for Oracle on the minimal Red Hat configuration assume that we can actually run Oracle effectively after removing recommended installation components. The true number of vulnerabilities is therefore likely to be somewhat higher than the figures shown in the graph for this build.

Days of Risk Discussion

Cumulative & Average

In our metrics, we refer to cumulative and average days of risk. Each is important to consider because each offers some insight into the user-centric security of the solution and also the security service of the vendor. *Cumulative days of risk* refers to the overall exposure of a solution as well as the security service of the vendor in one statistic. Of particular interest are the cumulative days of risk broken down by severity. This speaks to the total number of exposure days – which may overlap for different vulnerabilities – when the system was at greatest risk through vulnerability exploitation. Comparing the numbers, we see that the SQL Server 2000 on Windows Server 2003 solution had 952 cumulative days of risk for vulnerabilities with the “high” ICAT rating, compared to 1525 for the My SQL on RHEL solution and 2539 for the Oracle on RHEL solution, showing a clear lead by Microsoft.

Average days of risk are a valuable measure of vendor security response, as they speak to the average time between when a vulnerability is disclosed to public and when a vendor fix is available. The numbers for the period considered show a lead by Microsoft in this category as well with an average of 32.0 days, representing the time period that customers are exposed to higher levels of risk per vulnerability as compared with an average of 61.6 days of risk for the MySQL on RHEL solution and 38.7 days of risk for the Oracle on RHEL solution. It is interesting to note how these figures reflect the responsible vulnerability disclosure that Microsoft actively promotes, which leads to many vulnerabilities with zero days of risk, meaning that the vulnerability is disclosed as the fix is released. Note that Oracle’s extensive use of responsible disclosure also results in the Oracle on RHEL build having a lower average days of risk than the MySQL on RHEL build, though the larger number of total vulnerabilities still leads to a larger cumulative days of risk. A telling related statistic is that the median days of risk for vulnerabilities in the Microsoft solution is 0 as opposed to a median of 23 days of risk for the MySQL on RHEL solution and 18 for the Oracle on RHEL solution.

One interesting aspect of the challenge faced by Red Hat that is not obvious from a simple examination of the raw numbers is the delay between a fix becoming available within a product, and the inclusion of that product as an “approved” Red Hat package. For example, CAN-2004-0836 discusses a bug in MySQL’s `mysql_real_connect()` function. This was entered into the MySQL bug database on 4th June 2004, and fixed in the source tree 17th June 2004. However, Red Hat only packaged this fix in RHSA-2004:611, issued on the 27th of October. This problem of the management of published fixes from a third-party is a difficult one, and one which could represent a significant challenge to Linux on a go-forward basis.

Analysis NOTE: One of the key things we learned during reviews of our preliminary results is that there are strong opinions on the value, or lack of value, for days-of-risk metrics. For us, we see a pretty clear distinction in customer risk for before and after an event becomes widely known to the public. Thus, days of risk is an interesting measure to see how the combination of a vendor's disclosure policies, response process, and patch test and release process combine to shrink (or not shrink) the time period when customers are exposed without a patch alternative from the vendor. It is a real-world measure of a real-world problem, and for better or worse is affected by the way in which software is developed.

One point that was raised was that this type of metric automatically skews in favor of a closed source solution. Another point questions Microsoft's advocacy of "responsible disclosure" as it serves to make the days of risk number lower and Microsoft look better. In fact, a vendor with a longer quality and testing process might benefit more from responsible disclosure, but since that is the policy that leads to less risk for customers it seems to emphasize and not detract from the importance of the metric.

Actually, the data is pretty clear that both Microsoft and Linux community follow responsible disclosure to some degree. For example, for the Samba issue fixed by Red Hat in RHSA-2004:670-10, one can follow the references to see that the vendors kept the issue from being made public on the date when the vendors were first notified. Here is a partial bugzilla entry:

Jerry at Samba reported to vendor-sec on 20041209 a remote root flaw in Samba, affecting all versions. Requires authenticated user. Issue was discovered by iDEFENSE.

This issue is currently embargoed until 20041216:1200UTC

One might argue that the issue was "public" when first reported to vendor-sec, due to the number of people on the list. However, we used 12/16/2004 as the first public date the issue was widely known. Red Hat had 15 issues fixed with zero days and most of those benefited from responsible disclosure privately to the Linux vendors – actions we applaud.

Further, it seems clear that customer risk could be reduced even further by more security response coordination by Linux vendors. Note the example of RHSA-2004:413-07, which patched a kernel issue and made the issue public on 8/3/2004. SuSE users didn't have a patch until 8/9/2004 and Gentoo Linux users waited until 8/25/2004, according to the references in the CVE list.

The following tables show the results obtained for the days of risk analysis. Both the cumulative days of risk and the average days of risk are calculated because they represent different metrics about the vulnerabilities and the patching response.

Severity	Windows Server 2003/SQL Server 2000	RHEL ES 3/Oracle 10g	RHEL ES 3/MySQL 3.23
Days of Risk: High Severity	952	2539	1525
Days of Risk: Medium Severity	573	3314	3594
Days of Risk: Low Severity	0	574	741
Days of Risk: Not Known	490	1590	1290
Cumulative Days of Risk	2015	8017	7150
Average Days of Risk Per Vulnerability	31.98	38.73	61.64

Table 2: Days of risk the three systems studied

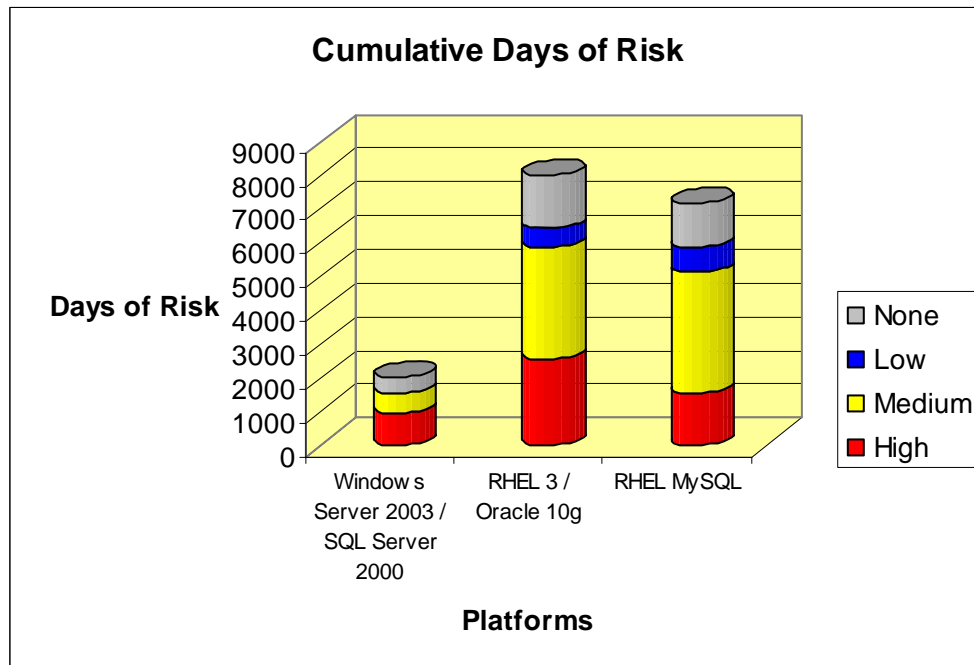


Figure 3 - Cumulative days of risk for the three systems studied

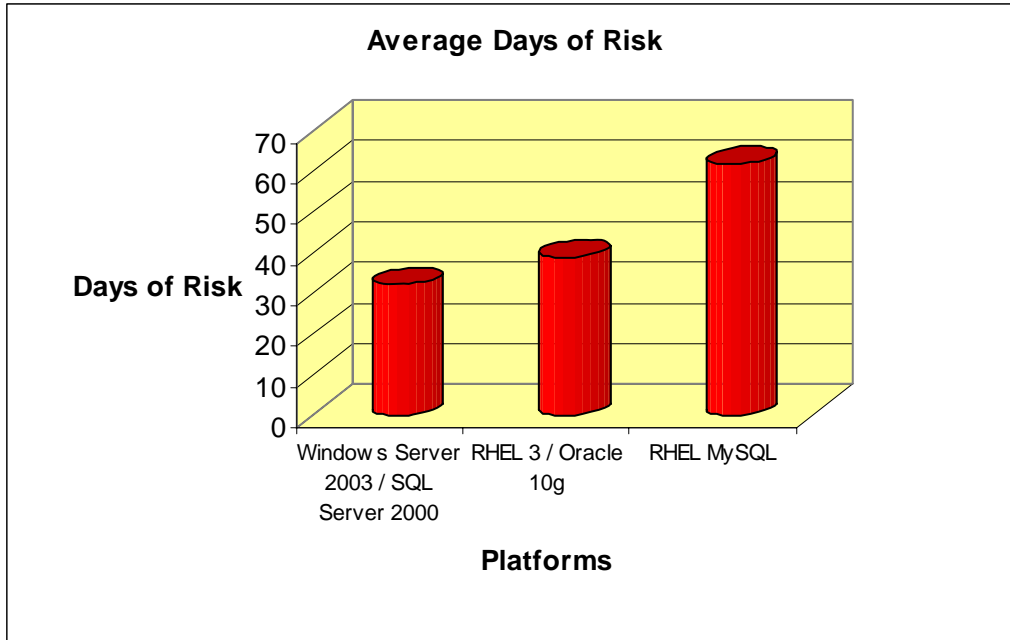


Figure 4 - Average days of risk for the three systems studied

Days of Risk Distribution

It is interesting to look beyond the average days of risk and examine the proportion of vulnerabilities fixed within a certain time window. Figures 4 – 6 break down the vulnerabilities fixed in 2004 for the platforms/configurations considered categorically by days of risk. In all platforms the proportion of vulnerabilities in the 0-30 days of risk categories is greatest. Looking at the data points reveals that 87.2% of the vulnerabilities in the 0-30 category for Microsoft are zero as opposed to 24.3% zeros for the MySQL on RHEL solution and 41.1% for the Oracle on RHEL solution within this category. This speaks to the disclosure model differences of the two platforms. This distribution is important, as research into rate of vulnerability exploitation has shown a strong increase in rate of exploitation as a function of time elapsed since disclosure. Thus, vulnerabilities with a long lifespan have a disproportionate impact on overall platform security, over and above their effect on days of risk calculations.

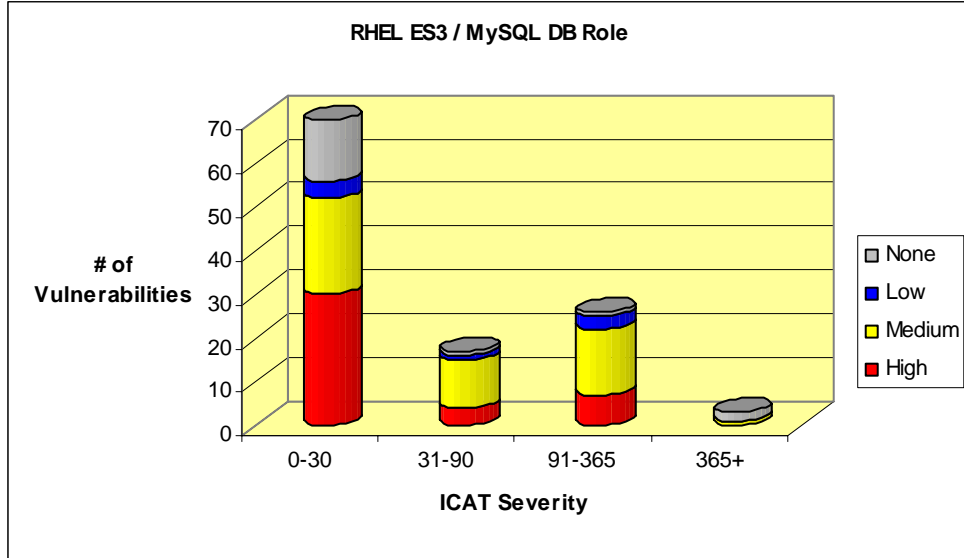


Figure 5: Days of Risk for RHEL 3 with MySQL broken down by time to fix

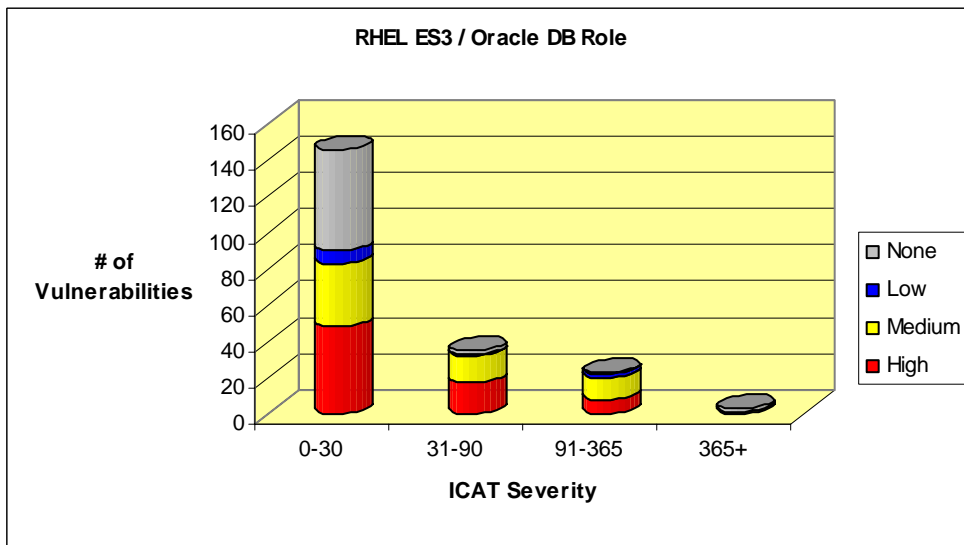


Figure 6: Days of Risk for RHEL 3 with Oracle broken down by time to fix

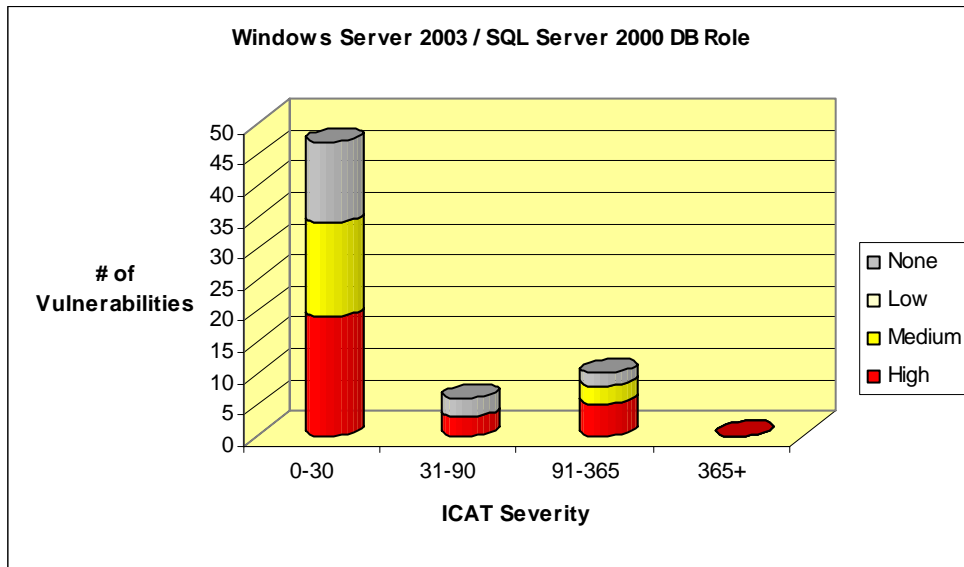


Figure 7: Days of Risk for SQL Server 2000 on Windows broken down by time to fix

Detailed Look at Fixes Taking 90+ Days

All three configurations considered have some vulnerabilities that have had more than 90 days between vulnerability disclosure and the release of a fix. Below, we will take a detailed look at those vulnerabilities for the solutions analyzed.

SQL Server 2000 on Windows Server 2003 – There are ten vulnerabilities fixed in the one year period under consideration that had more than 90 days of risk, and of these, five were designated by ICAT as high severity. Of the remaining vulnerabilities, three were designated as medium severity by ICAT and two had not been assigned a severity rating at the time of writing. Five of these vulnerabilities, CAN-2005-0053, CAN-2004-0727, CAN-2004-0841, CAN-2003-1041 and CAN-2003-1048, were in the Internet Explorer web browser, four were in the core system and one was in ASP .NET.

MySQL on RHEL – There were twenty nine vulnerabilities fixed in the one year period under consideration that had more than 90 days of risk, and of these, seven were designated by ICAT as high severity, sixteen as medium, three as low and three had not yet received severity rating by ICAT at the time of the study. Twelve of these vulnerabilities were in the operating system kernel. The remainder was spread out among a variety of packages, with MySQL as the second biggest offender (behind the kernel), with seven vulnerabilities.

Oracle on RHEL – There were twenty six vulnerabilities fixed in 2004 that had more than 90 days of risk, and of these, eight were designated by ICAT as high severity, thirteen as medium, two as low and three had not yet received severity rating by ICAT at the time of the study. Twelve of these vulnerabilities were in the operating system kernel with the remainder spread out among a variety of packages.

Qualitative Security Criteria

While the analysis of vulnerabilities and patches release gives us significant insight into the “exploitability” of a system, there are additional factors that may be important for those who must make platform procurement and deployment decisions. In this section we outline a variety of criteria that business decisions makers might consider in making security platform decisions. These criteria will aid in making sensitive security decisions in the context of practical operational considerations that might be important to an IT department. Specifically, we make the comparison between the two platforms - Red Hat Enterprise Linux 3 and Windows Server 2003 – that have served as a base for the three configurations studied in this report.

Of particular interest are those security features that are a standard part of the platform. For example, customers pay close attention to authentication features, support for VPNs, Autoupdate capabilities, buffer overrun protection, managed code capabilities and audit trails. Some of the most important qualitative issues are outlined in the following sections; this list is not exhaustive and should not in any way be taken as a comprehensive list of security-related features. Rather, it highlights those features that are readily measurable by an informed customer.

Port Protection/Firewall

Both vendors provide support for a firewall by default. Both firewall applications are basic and are based on IP tables. The firewall is installed by default on both platforms and blocks all incoming requests when running. For Red Hat Enterprise Linux 3, the firewall is on by default. For Windows Server 2003, the firewall must be turned on manually.

The ICF (Internet Connection Firewall) on the Windows Server 2003 platform performs basic functions and can be configured to log successful connections or dropped packets. The ICMP settings can be used to allow the server to communicate network status information. The default ICMP settings disallow most ICMP communications except for outgoing ICMP echo requests.

The `iptables` software on the Red Hat server has a text-based interface (“Security Level”) that enables basic firewall configuration (i.e. turning the firewall on or off and allowing connections towards a limited number of services). This interface also does not allow changing the ICMP configurations (ICMP echo replies are allowed by default) or monitor firewall logs. The most important feature is packet level filtering, which allows the administrator to establish firewall rules based on any aspect of the packet. The command line options also allow logging of the traffic based on the matching of packets with a certain rule.

Lifecycle Support Policy

One aspect of security is the duration of the support service. If a product is no longer supported, the users may be forced to upgrade or face increased risks of a security breach. While the information here is current as of the publication of this report, support policies have evolved significantly over the past several years for both vendors and are likely to

continue to be in flux. We therefore encourage you to visit vendor websites for the most current information.

The trend for servers in recent years, driven by customer requirements, is for the standard lifecycle to extend. Red Hat 9 only lasted one year, but with its Enterprise releases, Red Hat introduced a support lifecycle commitment that is currently 7 years for security-related issues¹⁶. Similarly, in 2002, Microsoft standardized its support lifecycle policies, and has recently extended that lifecycle to 10 years¹⁷. Beginning with its 9i Database Server release, Oracle's "Error Correction Support" policy has been extended to five years from product release with a 12 month notification to customers before it discontinues support¹⁸.

Microsoft's support of Windows Server 2003 started on the release date, May 28, 2003 and is planned to continue until May 2013. It's support for SQL Server 2000 began with the product's release on November 30, 2000 and is planned to continue for seven years from the next release of its database server (slated for 2005), taking support out till 2012. Red Hat's support for Enterprise Linux 3.0 started on the release date – October 23, 2003 – and is planned to be maintained for 7 years, ending in 2010. Error Correction Support for Oracle's 10g Database Server began with its release in early 2004, and is scheduled to continue till 2009.

In summary, the vendors have all committed to at least five years of long-term security support for their server software. Decision makers should examine the implications of those policies as they relate to their own production needs.

Bulletin/Advisory Descriptiveness

Security bulletins are often the only piece of information that system administrators use to make decisions concerning patch deployment from a risk-management perspective. It is therefore important that advisories contain sufficient information for these individuals to make informed and contextually relevant security decisions.

Red Hat advisories (see <https://rhn.redhat.com/errata/rhel3as-errata-security.html>) are very succinct and contain a small description of the vulnerability(ies) they address. The advisory contains little information about the context in which a vulnerability is present. No information about the patch, apart from the file's version number/patch number is provided.

Microsoft security bulletins contain data on mitigating factors, possible workarounds, consequences of patching, and a description of the patching process (see <http://www.microsoft.com/technet/security/current.asp> for Microsoft Security Bulletins). The advisories also contain information on the scope and the consequences of the vulnerability including the extent of the damage that could occur. In addition they include information on all the files that will be modified.

¹⁶ <https://www.redhat.com/apps/support/>

¹⁷ <http://support.microsoft.com/default.aspx?scid=fh;%5Bln%5D;LifeWin>

¹⁸ <http://www.oracle.com/support/policies.html>

Oracle Security Alerts have improved some in the past year, but provide much less detail concerning either vulnerabilities fixed. Oracle provides small summaries of the individual security issues fixed in a tabular format and also provides a 1-2 sentence “precondition for exploitability” statement for each vulnerability. One of the major challenges of the enterprise customer, however, is linking information in vendor bulletins with external information sources that may provide insight into threat profile and true risk. This can certainly be improved by Oracle’s active participation in the Mitre CVE program and their inclusion of CVE names for vulnerabilities in their bulletins as Red Hat and Microsoft do.

Patch Impact Disclosure

An important concern for system administrators is that a patch may disrupt the functionality of some other, sometimes unrelated, application that is running on the system. This may be caused by incompatibilities or downtime related to reboots or restarts during the patch installation process. Such concerns often cause administrators to delay the deployment of patches while patch validation takes place, thereby increasing the time during which systems are vulnerable but decreasing the chances the patch has unforeseen side-effects.

Microsoft security bulletins contain information concerning the files that will be modified, reboots and the impact of not patching. In addition, Microsoft provides tools such as the Baseline Security Analyzer that determines if the update is required on the system or not. However, the bulletins do not contain information concerning the amount of time required for installation.

The Red Hat security advisories are vulnerability oriented; that is, the information available relates to the cause of the vulnerability and the potential danger this vulnerability introduces. The advisories do not contain any information concerning the impact of the patching process, or required reboots. The only patch related information available is the versions of the packages that are deployed in the patch. The *precise* impact of a patch could be determined by examining the source code changes made between releases. While this is impractical in many environments, it is an additional possibility for systems that are critical.

Oracle Critical Patch Updates contain information on which high-level component is affected by the vulnerabilities listed but give no indication as to patch impact. Enterprise customers are therefore left to manual trials to determine what impact the patch may have to the running system.

Patch Deployment Technology

Timely patching of machines is a potent method of improving one’s probability of remaining secure. Even the current crop of worms which has plagued CSOs worldwide has been primarily addressable by rapidly and effectively patching machines, as in each case, worms found in the wild have exploited vulnerabilities which were already addressed by currently-available patch sets. Despite the conflict that exists in most large organizations between automated patch deployment and compatibility testing of patches that are mandated, the ability for machines to automatically update themselves is

extremely valuable for individual users and users within a small company that does not have the resources to support a patch management staff.

Microsoft and Red Hat both have auto-update features that enable a system to obtain security and functional updates automatically. These auto-update applications (Windows Automatic Update for the Microsoft operating system and `up2date` for the Red Hat operating system) can be set to notify and download, or to notify, download and install patches automatically.

An early advantage of the `up2date` application over the Windows automatic update was that it could also keep track of non-OS software through subscription channels on the Red Hat Network. However, Windows automatic update now also updates applications installed on top of the operating system, including SQL Server 2000. In addition, Microsoft has other tools available for patch management that can manage other Microsoft products in a centralized tool (SUS/WUS – Software Update Services/Windows Update Services – and SMS 2003 – System Management Server 2003) and also allow more flexibility.

With the release of 10g, Oracle offers its Enterprise Manager tool that automates the notification and delivery of Oracle patches through a connection to Oracle’s MetaLink support site. Enterprise Manager can be configured to notify administrators of patch availability based on their deployed products and versions and allows administrators to schedule deployments in a centralized manner.

Patch Release Timing (Grouping)

All three vendors have adopted a policy of releasing grouped patches. Microsoft’s policy groups patches on a monthly cycle, Oracle releases its Critical Patch Updates every three months and Red Hat bundles “errata” together when possible but often releases security patches as needed.

Specifically, Microsoft’s policy is to release patches on the first calendar Tuesday of every month. Oracle’s policy is to release its critical patch updates on the Tuesday closest to the 15th of January, April, July and October. Although Red Hat states that there is a grouped release of patches there is no indication of any static date or release cycle allowing system administrators to schedule patch management cycles.

Patch Rollback Capability

In the event of a “bad” patch that wreaks havoc in a system or simply causes other applications to misbehave, its removal may be warranted. Both platforms allow removal of unwanted patches, however the Microsoft patches keep track of the modified files, and uninstalling a patch merely requires the use of the Windows default install/uninstall feature and choosing to remove the patch. Unfortunately, this capability has not been uniformly available for all Microsoft software, but it is available with Windows Server software patches.

The Red Hat patch management tool RPM also allows removal of patches; however it is the user’s responsibility to specify which packages have been modified by the `up2date`

tool. The `up2date` tool can list these packages by using the “`--list-rollbacks`” switch on the command line.

Oracle allows patch rollback through its `opatch` command line tool. Users must supply the platform-specific patch ID number included in the patch readme file. While Oracle’s Enterprise Manager tool does not support patch rollback directly, the process is automatable with remote script execution through Enterprise Manager.

Conclusions

Whenever we deploy a new technology in the enterprise, we think in terms of adding a strategic business asset. Nowhere is this truer than in the management of data, where modern relational databases transform vast quantities of data into information that fuels business. A key challenge is that there are usually several different technologies, platforms, and solutions that might satisfy a set of business requirements, and each solution has costs. Some of those costs, like purchase price, are easy to assess but others, like the cost of exposure from latent flaws in those systems are much more nebulous. In this report, we have studied both quantitative and qualitative data that affects the vulnerability and thus operational security risk of different database server platforms. These results provide some additional aspects of system security to consider when making adoption decisions.

Specifically, for the database server role, we considered three configurations; Microsoft SQL Server 2000 on Windows Server 2003, Oracle 10g on Red Hat Enterprise Linux 3 and MySQL on Red Hat Enterprise Linux 3. In order to produce a meaningful comparison of platforms, the systems studied were manually installed and their configurations were verified.

When considering quantitative data, we examined the number and type of vulnerabilities that have been reported for each platform. We filtered these vulnerabilities based upon the features and packages installed on our three system builds. For each vulnerability, we determined the total time that elapsed between wide public disclosure of the vulnerability and the availability of a patch that closed the vulnerability.

In terms of vulnerability counts, the SQL Server 2000 on Windows solution showed a clear advantage over the Oracle and MySQL solutions built on the Red Hat Linux server. While one must consider the security of the underlying platform for any deployed database system, it is also illustrative to look at the vulnerabilities in the database server software itself. This data shows a fairly significant advantage to SQL Server 2000 over both MySQL and Oracle.

It is interesting to note, however, is that these database servers are at different stages of their respective lifecycles. SQL Server 2000 has been in use since 2000 while Oracle has had more frequent releases, with its 10g database server shipping in early 2004. A telling statistic then is the *total number* of vulnerabilities in the product's lifetime, and *when* in its lifecycle these issues were found. SQL Server 2000 has had a total of 36 vulnerabilities reported over the 4+ years since its release in 2000: 8 in 2000, 4 in 2001, 19 in 2002, 4 in 2004 and 1 in 2004. Oracle enumerates 30 vulnerabilities in the first year of release for its 10g database server, with 10 of these vulnerabilities being externally assigned a CAN identifier. For MySQL the picture is less clear because of the nature of its incremental releases. If one were to look at the one year period ending February 28, 2005, MySQL server had 7 distinct vulnerabilities.

Beyond vulnerability counts, we also looked at the cumulative and average period of exposure to vulnerabilities of the three builds – the so called Days of Risk metric. In terms of average days of risk, Microsoft had the lowest at 32.0 follows by the Oracle on RHEL3 solution with 38.7 and the MySQL solution with 61.6. Both the SQL Server 2000

on Windows Server 2003 and the Oracle 10g on RHEL 3 have benefited in average days of risk through their strong encouragement of “responsible disclosure,” where they attempt to carefully coordinate vulnerability announcement with fix announcement and actively build relationships with new security researchers. Red Hat data shows evidence of leveraging a responsible disclosure policy as well, with several zero day fixes. This helps drive down averages in a way that directly reduce customer risk.

Qualitatively, we outlined many factors that ultimately drive the viability of a particular solution in terms of security. While these aspects of a solution may be difficult to consider in a quantitative way, it is clear that they play a role in determining the ease with which security can be managed and maintained, and have a direct impact on overall risk. Each organization needs to consider these qualitative factors as well as those metrics that can be assigned a hard number when making a deployment decision.

On balance, as security practitioners, we know that database solutions from Microsoft, Oracle and the open source world running on both the Red Hat and Microsoft platforms can be used to provide a secure solution when deployed and administered with the right skills and under the right policies. Looking at the software security factors that each vendor has the ability to directly affect however, such as software security quality and security response, of the three database systems studied, the Microsoft SQL Server 2000 on Windows Server 2003 solution had fewer security vulnerabilities and fewer days of risk compared to the MySQL and Oracle solutions on Red Hat Enterprise Linux 3.

It is impossible (and irresponsible) to provide a comprehensive comparison of security concerns that will apply to all operating environments. Our research has shown that the threat profile a box faces can be an important determinant when assessing overall security. As such, the importance of each contributing factor toward security must be weighted for every threat environment.

Combined Table of Comparisons

The following table summarizes our findings with respect to vulnerability counts for the three configurations considered:

Severity	Windows Server 2003/SQL Server 2000	RHEL ES 3/Oracle 10g	RHEL ES 3/MySQL 3.23.58
High	27	73	41
Medium	18	63	49
Low	0	10	8
Not Known	18	61	18
Total	63	207	116

Table 3: Vulnerability count summaries for the three solutions studied

The table below summarizes the days of risk results for the three configurations considered:

Severity	Windows Server 2003/SQL Server 2000	RHEL ES 3/Oracle 10g	RHEL ES 3/MySQL 3.23
Days of Risk: High Severity	952	2539	1525
Days of Risk: Medium Severity	573	3314	3594
Days of Risk: Low Severity	0	574	741
Days of Risk: Not Known	490	1590	1290
Cumulative Days of Risk	2015	8017	7150
Average Days of Risk Per Vulnerability	31.98	38.73	61.64

Table 4: Days of Risk summaries for the three solutions studied

Appendix A: Step-by-Step Methodology

Our goal in the quantitative vulnerability comparisons of this document was to create and follow a methodology that was clear and that would provide meaningful results to decision makers. In this appendix we include a step-by-step process for reconstructing the data analyzed in this report, as well as the resulting metrics.

Below are the steps that can be followed to build your own set of data for analysis.

A. Build out a spreadsheet of vulnerabilities for Windows Server 2003

- a. Sequentially examine each Security Bulletin released by Microsoft during the time period studied, not relying on the Security Bulletin search list provided. Microsoft Security Bulletins and the vulnerabilities addressed by them originate at: <http://www.microsoft.com/technet/security/current.aspx>.
- b. For each Bulletin, read through and identify if Windows Server 2003 is affected. Typically, Microsoft includes a table in the Executive Summary of the bulletin with a row for each vulnerability listed by CVE Name and showing the Microsoft severity for each platform affected.
- c. For each CVE Name, fill out the following columns. Note that a Bulletin addressing multiple vulnerabilities results in multiple rows:
 - i. CVE Name (e.g. CAN-2004-1028)
 - ii. MSFT Security Bulletin identifier (e.g. MS04-007)
 - iii. Date of Security Bulletin/Fix
- d. Since we assume all components are present on WS2003, there is no need to group components into role groups as we do for Red Hat.
- e. Since we assume all components are present on WS2003, we do not do a validation step to see if the component is physically installed. We assume it is installed in all cases.

B. Build out a spreadsheet of vulnerabilities for Red Hat Enterprise Linux 3 Enterprise Server (RHEL3ES)

- a. Sequentially examine each security advisory for RHEL3ES released by Red Hat during the time period studied. RHEL3ES security advisories and the vulnerabilities addressed by them originate at: <https://rhn.redhat.com/errata/rhel3es-errata-security.html> .
- b. For each security advisory, read through and identify and confirm that RHEL3ES is affected. The advisory identifier, affected component and associated CVE Names are typically all listed in the header of the security advisory.
- c. For each CVE Name, fill out the following columns. Note that an advisory addressing multiple vulnerabilities results in multiple rows:

- i. CVE Name (e.g. CAN-2004-1028)
- ii. Security Advisory identifier (e.g. RHTA-2005:010)
- iii. Date of Security Advisory/Fix
- iv. Each package patched

C. Build out a spreadsheet of vulnerabilities for Oracle 10g

- a. Sequentially examine each Oracle security advisory for 10g released by Oracle during the time period studied. Oracle security advisories and the vulnerabilities addressed by them originate at:
<http://www.oracle.com/technology/depoy/security/alerts.htm>.
- b. For each security advisory, read through and identify and confirm that Oracle 10g is affected. The vulnerability identifier is typically listed as DBNN in a table near the end of the document. Examine the “Last Affected Patch Set” column to determine if the vulnerability applies to 10g.
- c. For each vulnerability, fill out the following columns. Note that an advisory addressing multiple vulnerabilities results in multiple rows:
 - i. Vulnerability identifier
 - ii. Security Advisory identifier (e.g. Alert 68)
 - iii. Date of Security Advisory/Fix
- d. Manually search for cross-references to CVE identifiers. NOTE: this is a difficult step and is only necessary for Days of Risk calculations, vulnerability counts can happen without it.

D. Gather Information to Calculate Days of Risk

- a. For each CVE Name on either server spreadsheet, look up the references listed at <http://cve.mitre.org>. For example, CAN-2004-0021 details are listed at <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0021>.
- b. Follow each reference, examining the date of publication of the referenced web page that made the issue public. Enter the oldest date into the spreadsheet as “Date first public” and the URL into the spreadsheet as “First public reference”.
- c. For bug database entries, the oldest entry may not have been publicly readable when entered. Check the bug server policy first to see if entries are always public. If not, look for an entry indicating that the privacy flag has been changed to make the bug publicly viewable¹⁹.
- d. Since the CVE list does not guarantee to capture the first public reference, though often it does, cross check the references with any references listed in

¹⁹ Note this is a new methodology step added with the release of this report to accommodate vulnerabilities in open source software that were kept private through bug management software and were later made public. Our methodology counts the public date as a bug’s public disclosure.

the security advisory or Security Bulletin. If an earlier public reference is found, use the earlier reference and date.

- e. Additionally, search the Internet and common security newsgroups for public discussion of the security vulnerability and use that date and reference if found.
 - f. Add a column to the spreadsheets called “Days of Risk” and subtract the “Date first public” from the “Date of Bulletin/fix” to calculate the days of risk for that vulnerability.
- E. For the WS2003 and RHEL3ES spreadsheet, add the ICAT severity listing.
- a. Download the latest ICAT Metabase from <http://icat.nist.gov/icat.cfm>.
 - b. For each CVE Name in the server spreadsheets, enter a column value of HIGH, MEDIUM, LOW or UNRATED
- F. For the RHEL3ES spreadsheet, create a spreadsheet for each server role.
- a. Install and build a RHEL3AS system in the configuration being studied (e.g. web server role). Use the ‘rpm -qa’ command to check for each package that is patched with any security advisory during the time period.

With the above five steps completed, you should have spreadsheets capturing the list of vulnerabilities for each platform and server role for a given time period, along with the severity rating, date first public, first public reference and the days-of-risk calculation. From this, you can calculate counts, totals and averages as desired.

Appendix B

The following text is taken from the recommended installation procedures for Oracle 10g on Red Hat Enterprise Linux 3 which can be found at http://www.oracle.com/technology/pub/articles/smiley_10gdb_install.html#rhel3. The information below is current as of May 1st, 2005.

RHEL3

Oracle Database 10g is certified to run the base release of Red Hat Enterprise Linux 3 (Advanced Server and Enterprise Server) without updates. If you have update CDs, you can use the boot CD from the update instead of the boot CD from the base release to automatically apply all updates during the installation. All updates from Red Hat are supported by Oracle.

1. Boot the server using the first CD.
 - You may need to change your BIOS settings to allow booting from the CD.
2. The boot screen appears with the `boot :` prompt at the bottom of the screen.
 - Select **Enter** to continue with a graphical install on the console. (For other installation methods and options, refer to the *Red Hat Installation Guide*.)
 - The installer scans your hardware, briefly displays the Red Hat splash screen, and then begins a series of screen prompts.
3. Language Selection
 - Accept the default.
4. Keyboard Configuration
 - Accept the default.
5. Welcome Screen
 - Click on **Next**.
6. Mouse Configuration
 - Accept the default.
7. Installation Type
 - Select **Custom**.
8. Disk Partitioning Setup
 - A thorough treatment of disk partitioning is beyond the scope of this guide, which assumes that you are familiar with disk partitioning methods.

(WARNING: Improperly partitioning a disk is one of the surest and fastest ways to **wipe out everything on your hard disk**. If you are unsure how to proceed, stop and get help, or you will risk losing data!)

This guide uses the following partitioning scheme, with ext3 for each filesystem:

The 9GB disk on the first controller (`/dev/sda`) will hold all Linux and Oracle software and contains the following partitions:

- 100MB /boot partition
- 1,500MB swap partition—Set this to at least twice the amount of RAM in

the system but to no more than 2GB (32-bit systems do not support swap files larger than 2GB). If you need more than 2GB of swap space, create multiple swap partitions.

-7,150MB root partition— This partition will be used for everything, including /usr, /tmp, /var, /opt, /home, and more. This was done purely to simplify installation for the purposes of this guide. A more robust partitioning scheme would separate these directories onto separate filesystems.

9. Boot Loader Configuration
 - Accept the default.
10. Network Configuration
 - It is usually best to configure database servers with a static IP address. To do so, click on **Edit**.
 - A pop-up window appears. Uncheck the **Configure using DHCP** box, and enter the IP Address and Netmask for the server. Be sure that **Activate on boot** is checked, and click on **OK**.
 - In the Hostname box, select **manually** and enter the hostname.
 - In the Miscellaneous Settings box, enter the remaining network settings.
11. Firewall Configuration
 - For the purposes of this walk-through, no firewall is configured. Select **No firewall**.
12. Additional Language Support
 - Accept the default.
13. Time Zone Selection
 - Choose the time settings that are appropriate for your area. Setting the system clock to UTC is usually a good practice for servers. To do so, click on **System clock uses UTC**.
14. Set Root Password
 - Enter a password for root, and enter it again to confirm.
15. Package Group Selection
 - Select only the package sets shown here. Leave all others unselected.
 - Desktop
 - X Window System
 - Gnome
 - KDE
 - See my comments in the RHES 2.1 section regarding choice of GUI.
 - Applications
 - Editors
 - Graphical Internet
 - Servers
 - Do not select anything in this group.
 - Development
 - Development Tools
 - System
 - Administration Tools
 - Red Hat Enterprise Linux
 - Do not select anything in this group.
 - Miscellaneous
 - Legacy Software Development
 - Click on **Next** to proceed.
16. About to Install
 - Click on **Next**.
17. Installing Packages

- Software will be copied to the hard disk and installed. Change disks as prompted, and click on **Next** when the installation is complete.
- 18. Graphical Interface (X) Configuration
 - Accept the defaults unless the installer does not recognize your video card. If your video card is not recognized, you will not be able to continue.
- 19. Monitor Configuration
 - Accept the default if the installer correctly identifies your monitor. Otherwise, select a compatible monitor from the list.
- 20. Customize Graphical Configuration
 - Accept the defaults.
- 21. Congratulations
 - Remove the installation media from the system, and click on **Next**.
- 22. The system automatically reboots and presents a new welcome screen.
 - Click on **Next**.
- 23. License Agreement
 - Read the license agreement. If you agree to the terms, select **Yes, I agree to the License Agreement** and click on **Next**.
- 24. Date and Time
 - Set the Date and Time.
 - If you want to use an NTP server (recommended), select **Enable Network Time Protocol** and enter the name of the NTP server.
- 25. User Account
 - Create an account for yourself.
 - Do not create an account for oracle at this time. Creating the oracle account is covered later in this section.
- 26. Red Hat Network
 - If you want to use or activate your Red Hat Network account now, accept the default, click on **Next**, and follow the product activation instructions that accompanied your Red Hat product.
- 27. Additional CDs
 - Click on **Next**.
- 28. Finish Setup
 - Click on **Next**.
- 29. A graphical login screen appears.
- 30. Congratulations! Your Linux software is now installed.

Verifying Your Installation

If you've completed the steps above, you should have all the packages and updates required for Oracle Database 10g. However, you can take the steps below to verify your installation.

Required kernel version: 2.4.21-4.EL (This is the kernel version shipped with the base release of RHEL3. This kernel, or any of the kernels supplied in updates, works with Oracle Database 10g.)

Check your kernel version by running the following command:

```
uname -r
```

Ex:

```
# uname -r  
2.4.21-4.0.1.ELsmp
```

Other required package versions (or later):

- gcc-3.2.3-2
- make-3.79
- binutils-2.11
- openmotif-2.2.2-16
- setarch-1.3-1
- compat-gcc-7.3-2.96.122
- compat-gcc-c++-7.3-2.96.122
- compat-libstdc++-7.3-2.96.122
- compat-libstdc++-devel-7.3-2.96.122
- compat-db-4.0.14.5 (listed in the *Oracle 10g Database Installation Guide* as required but not needed here)

To see which versions of these packages are installed on your system, run the following command as root:

```
rpm -q gcc make binutils openmotif setarch compat-db compat-gcc \  
    compat-gcc-c++ compat-libstdc++ compat-libstdc++-devel
```

Ex:

```
# rpm -q gcc make binutils openmotif setarch compat-db compat-gcc \  
\  
>    openmotif compat-gcc-c++ compat-libstdc++ compat- \  
libstdc++-devel \  
gcc-3.2.3-20 \  
make-3.79.1-17 \  
binutils-2.14.90.0.4-26 \  
openmotif-2.2.2-16 \  
setarch-1.3-1 \  
package compat-db is not installed \  
compat-gcc-7.3-2.96.122 \  
compat-gcc-c++-7.3-2.96.122 \  
compat-libstdc++-7.3-2.96.122 \  
compat-libstdc++-devel-7.3-2.96.122
```

Note that the compat-db package is not installed. This package is not available from any of the package groups available during installation and must be installed in a separate step. If any of the other package versions on your system are missing or the versions are earlier than those specified above (other than compat-db), you can download and install the updates from the Red Hat Network.

Installing compat-db

Insert CD2 of the original Red Hat Enterprise Linux media. (This package has not been updated as of Update 2 and is found only on the original media.)

The CD mounts automatically.

Run the following command as root:

```
rpm -ivh /mnt/cdrom/RedHat/RPMS/compat-db-4.0.14-5.i386.rpm
```

Ex:

```
# rpm -ivh /mnt/cdrom/RedHat/RPMS/compat-db-4.0.14-5.i386.rpm
```

```
Preparing... #####  
[100%]  
  1:compat-db #####  
[100%]
```