Using Diversity in Cloud-Based Deployment Environment to Avoid Intrusions

Anatoliy Gorbenko[1], Vyacheslav Kharchenko[1], Olga Tarasyuk[1], Alexander Romanovsky[2]

[1]Department of Computer Systems and Networks (503), National Aerospace University, Kharkiv, Ukraine
A.Gorbenko@csac.khai.edu, V.Kharchenko@khai.edu, O.Tarasyuk@csac.khai.edu
[2]School of Computing Science, Newcastle University, Newcastle upon Tyne, UK
Alexander.Romanovsky@ncl.ac.uk

Abstract. This paper puts forward a generic intrusion-avoidance architecture to be used for deploying web services on the cloud. The architecture, targeting the IaaS cloud providers, avoids intrusions by employing software diversity at various system levels and dynamically reconfiguring the cloud deployment environment. The paper studies intrusions caused by vulnerabilities of system software and discusses an approach allowing the system architects to decrease the risk of intrusions. This solution will also reduce the so-called system's *days-of-risk* which is calculated as a time period of an increased security risk between the time when a vulnerability is publicly disclosed to the time when a patch is available to fix it.

1 Introduction

Dependability is a system property which includes several attributes: availability, reliability, safety, security, etc [1]. In this paper we focus on ensuring security of web services by protecting them from malicious attacks that exploit vulnerabilities of system components. A computer system providing web services consists of hardware and a multitier system architecture playing the role of a deployment environment for the specific applications. The dependability of the deployment environment significantly affects the dependability of the services provided.

A web service application can be created as servlets and server pages, java beans, stored database procedures and triggers run in a specific deployment environment. This environment is constructed from a number of software components (Fig. 1). Typical examples of system components for web services are operating system (OS), web and application servers (AS and WS), and data base management systems (DBMS).

Vulnerabilities of operating system and system software represent threats to dependability and, in particular, to security, that are additional to faults, errors and failures traditionally dealt with by the dependability community [1, 2].

Intrusion tolerance is a general technique, which aims at tolerating system vulnerabilities that have been disclosed and can be exploited by an attacker. This is an active area of research and development with many useful solutions proposed (e.g. [3, 4, 5]). However, traditionally the focus has been on intrusion detection (e.g. [6, 7]), or combining different security methods [8, 9] (intrusion detection systems and firewalls, authentication and authorisation techniques, etc.). Less attention has been given to understanding how to make systems less vulnerable and to avoid intrusions while being configured and integrated out of COTS-components, such as OS, WS, AS, and DBMS.

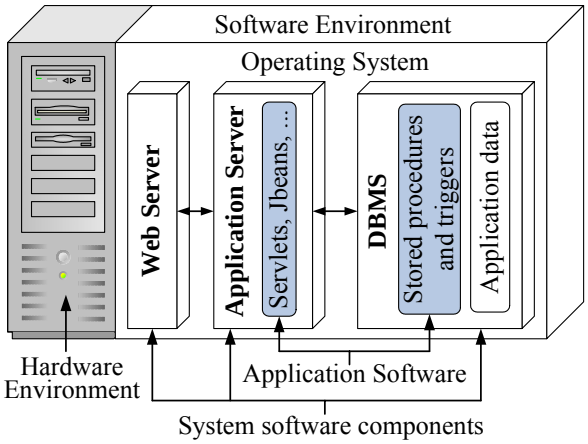

Fig. 1. General multitier architecture of a web system

In the paper we propose a general system architecture aiming at decreasing the risk of intrusion and reducing number of *days-of-risk* [10]. This architecture employs the diversity of the system software components and uses a dynamical reconfiguration strategy taking into account the current number of vulnerabilities (and their severity) of each component.

Such vulnerability information can be obtained by querying existing vulnerability databases (e.g. Common Vulnerabilities and Exposures, CVE and National Vulnerability Database, NVD), publicly available in the Internet. Besides, an implementation of the proposed architecture relies on the emerging cloud infrastructure services [11], known as Infrastructure as a Service (IaaS). IaaS provides a platform virtualization environment and APIs that can enable such dynamic reconfiguration by switching between pre-built images of the diverse deployment environments.

The rest of the paper is organised as follows. In Section 2 we describe a general architecture of intrusion-avoidance deployment environment making use of diversity of systems components and dynamical reconfiguration. Section 3 investigates diversity of the system software used as a deployment environment for modern Web applications. In Section 4 we discuss the vulnerabilities of some popular operating systems and demonstrate the feasibility of the approach proposed.

2 Intrusion-Avoidance Architecture

2.1 Diversity of Deployment Environment

Design diversity is one of the most efficient methods of providing software fault-tolerance [12]. In regard to multitier architecture of web-services, software diversity can be applied at the level of the operating system, web and application servers, data base management systems and, finally, for application software, both separately and in many various combinations. It is an obvious fact that some system software components may be incompatible with each other (i.e. Microsoft IIS and MS SQL can be run only on MS Windows OS series and incompatible with other operating systems). Hence, this fact should be also taken into account during the multiversion environment integration and selection of the particular system configuration.

Platform-independent Java technologies provide the crucial support for applying diversity of different system components. Thanks to JVM, Java applications can be run on different operating systems under control of various web and applications servers (see Table 1). These components form a flexible deployment environment running the same application software and allowing to be dynamically reconfigured by replacing one component by another one of the same functionality (e.g. GlassFish AS can be replaced with Oracle WebLogic, or IBM WebSphere, etc.). At the same time, the .NET applications can employ only restricted diversity of the deployment environment limited to Microsoft Windows series of operating systems and different versions of Internet Information Server and MS SQL.

Table 1. Diversity level and diversity components of Java deployment environment.

Diversity Level	Diverse system components
Operating Systems (OS)	WinNT Series, MacOS X Server, Linux, FreeBSD, IBM AIX, Oracle Solaris, HP-UX, etc
Web-server (WS)	Apache httpd, Oracle iPlanet Web Server, IBM HTTP Server, lighttpd, nginx, Cherokee HTTP Server, etc
Application server (AS)	GlassFish, Geronimo, Oracle WebLogic, JBoss, Caucho Resin, IBM WebSphere, SAP NetWeaver, Apple WebObjects, etc
DBMS	MS SQL Server, MySQL, Oracle Database, Firebird, PostgreSQL, SAP SQL Anywhere, etc

2.2 Intrusion-avoidance architecture making use of system components diversity

The proposed intrusion avoidance approach is based on the idea of running at the different levels of the multitier system architecture (OS, WS, AS and DBMS) only those components having the least number of vulnerabilities. The rest diverse components should be hold in a stand-by mode.

When a new vulnerability is disclosed, the most vulnerable system component should be replaced with the diverse one having fewer numbers of open (i.e. yet unpatched) vulnerabilities. Such dynamic reconfiguration should also take into account severity and potential harmful consequences of different vulnerabilities, their popularity, availability of exploit code, etc. When a product vendor patches some vulnerability the system can be reconfigured again (after patch installation and re-estimation of the security risks). To compare the vulnerability level of the diverse components and spare configurations the weighted metrics can be used (1), (2). The proposed metrics estimating the system security risk can be also extended by taking into account other vulnerability attributes apart from severity and popularity.

$$VLC_i = \sum_{j=1}^{N_i} S_j \cdot P_j \,, \tag{1}$$

where VLC_i – vulnerability level (security risk) of the i-th system component; N_i – number of open (yet unpatched) vulnerabilities of the i-th component; S_j – severity of the j-th vulnerability; P_j – popularity of the j-th vulnerability.

$$VLS_k = \sum_{i=1}^{M_k} VLC_i \,, \tag{2}$$

where VLS_k – vulnerability level (security risk) of the k-th system configuration; M_k – number of system components (usually, each system configuration uses four basic system components: OS, WS, AS, DBMS); VLC_i – vulnerability level (security risk) of the i-th system component.

The general intrusion-avoidance architecture is presented in Fig. 2. The architecture employs the IaaS (Infrastructure as a Service) cloud technology [11] providing crucial support for dynamic reconfiguration, storage and maintenance of the images of spare diverse system configurations. The core part of such architecture is a configuration controller. It retrieves information about emerging vulnerabilities of the different software system components and information about patches and security advisories released by the companies (product owners). By analysing such information the configuration controller estimates current security risks, selects the less vulnerable system configuration and activates it. Other functions performed by the controller are patch and settings management of the active and spare diverse configuration.

3 Demonstration and Simulation

In this section we analyze vulnerability statistics of different operating systems gathered during 2010 year and show how the security risks would be reduced by employing the proposed intrusion-avoidance technique. To perform a demonstration we have developed the testbed software implementing the functionality of the configuration controller (see Fig. 3) and have simulated its decision-making and reconfiguration process by use of operating systems vulnerability statistics.

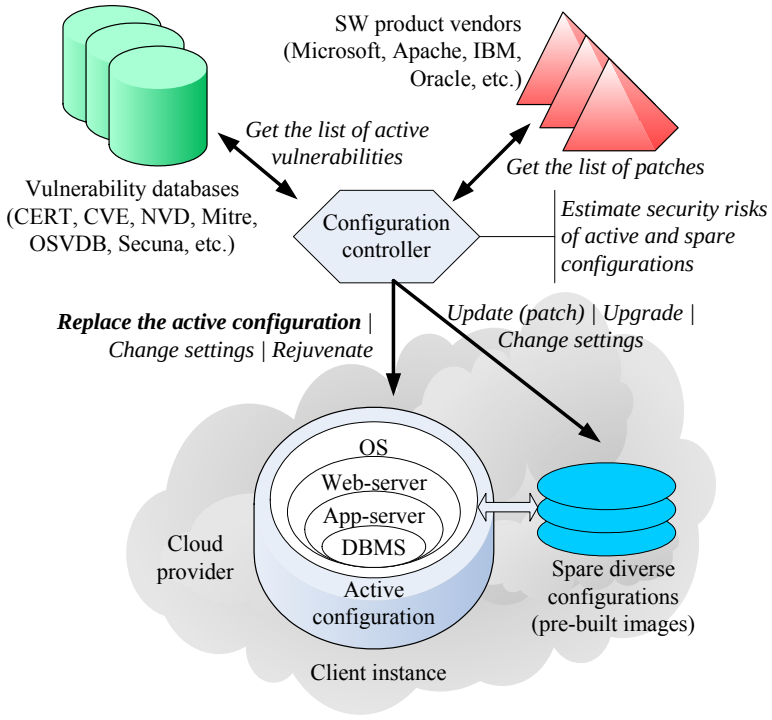

Fig. 2. General architecture of the cloud-based intrusion-avoidance deployment environment

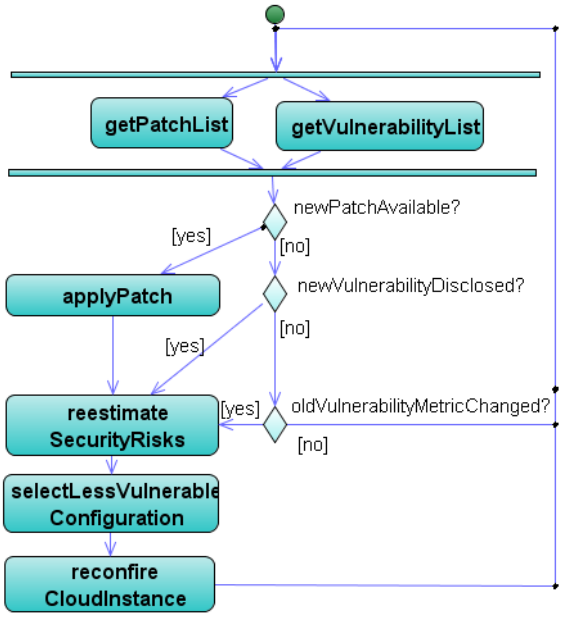

Fig. 3. Basic configuration controller's functionality: UML Activity diagram

3.1 Vulnerability Databases

There are a number of databases (supported by commercial and government institutions) gathering and publicly providing information about software vulnerabilities: CVE (cve.mitre.org), NVD (nvd.nist.gov), XForce (xforce.iss.net), CERT (www.cert.org), Secunia (secunia.com), etc. They help customers and product owners in identifying and solving the known security problems. The most reputable and complete databases are CVE and NVD, providing XML-formatted vulnerability information. This information (see Fig. 4) includes a unique vulnerability identifier `<vuln:cve-id>` and the date of its disclosure `<vuln:published-datetime>`, vulnerability description `<vuln:summary>`, severity `<cvss:score>` and impact attributes (impact on confidentiality `<cvss:confidentiality-impact>`, integrity `<cvss:integrity-impact>` and availability `<cvss:availability-impact>`), the list of vulnerable software `<vuln:vulnerable-software-list>` and other related data.

```
<entry id="CVE-2010-0020">
  <vuln:vulnerable-software-list>
    <vuln:product>
      cpe:/o:microsoft:windows_server_2008::sp2:x32
    </vuln:product>
     ...
  </vuln:vulnerable-software-list>
  <vuln:cve-id>CVE-2010-0020</vuln:cve-id>
  <vuln:published-datetime>2010-02-10T13:30:00
  </vuln:published-datetime>
   ...
  <vuln:cvss>
    <cvss:score>9.0</cvss:score>
    <cvss:access-vector>NETWORK</cvss:access-vector>
    <cvss:access-complexity>LOW</cvss:access-complexity>
    <cvss:confidentiality-impact>COMPLETE
    </cvss:confidentiality-impact>
    <cvss:integrity-impact>COMPLETE</cvss:integrity-impact>
    <cvss:availability-impact>COMPLETE</cvss:availability-impact>
    <cvss:source>http://nvd.nist.gov</cvss:source>
  </vuln:cvss>
   ...
  <vuln:cwe id="CWE-94" />
  <vuln:cwe id="CWE-20" />
  <vuln:references xml:lang="en" reference_type="UNKNOWN">
    <vuln:source>MS</vuln:source>
    <vuln:reference href="http://www.microsoft.com/technet/
      security/Bulletin/MS10-012.mspx" xml:lang="en">MS10-012
    </vuln:reference>
  </vuln:references>
  <vuln:summary>The SMB implementation in the Server service in
Microsoft Windows Server 2008 SP2 does not properly validate
request fields, which allows remote authenticated users to
execute arbitrary code via a malformed request, aka "SMB
Pathname Overflow Vulnerability." </vuln:summary>
</entry>
```

Fig. 4. Fragment of NVD's vulnerability record: an example

Unfortunately, none of existing vulnerability databases provides centralized information about patches and fixes available and exact dates of their issue. This complicates estimation of *days-of-risk* for each particular vulnerability. Assessment of the security risk and estimation of the number of open (i.e. yet unpatched) vulnerabilities of some particular software component can be done on the base of on-line monitoring of both vulnerability databases and vendor security bulletins.

To browse and query the NVD database we have developed a special software tool called *NVDView*. This tool allows to get an aggregated vulnerability information about the specified software product during the specified period of time. The results of vulnerability analysis of different operating systems obtained with the help of *NVDView* tool are discussed in the next section.

3.2 Operating systems vulnerability analysis

In this section we provide a retrospective vulnerability statistical analysis of the different operating systems using information provided by NVD database. Table 2 reports a number of vulnerabilities disclosed during 2010 for Novel Linux v.11, RedHat Linux v.5, Apple MacOS Server v.10.5.8, Sun/Oracle Solaris v.10 and Microsoft Windows Server 2008.

Table 2. A number of vulnerabilities disclosed in 2010 for different operating systems.

Operating System	Number of vulnerabilities disclosed per month												Total number
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	
Novel Linux 11	3	7	5	11	5	4	0	1	33	5	18	23	**115**
RedHat Linux 5	3	7	5	11	6	4	0	1	32	4	18	23	**114**
Apple MacOS Server 10.5.8	2	0	25	0	0	10	0	3	0	0	18	0	**58**
Sun/Oracle Solaris 10	1	1	0	0	0	0	10	0	0	11	0	0	**23**
MS Windows Server 2008	3	16	8	14	2	5	2	14	5	6	0	16	**91**

It is clear, that the total (cumulative) number of vulnerabilities is not the best security metric to compare different software product. Since some period of time after vulnerability disclosure (called *days-of-risk* [10]) the product vendor issues a patch to fix the vulnerability. Thus, the most important characteristics is a number of the residual or open (i.e. yet unpatched) vulnerabilities. Cumulative and residual numbers of vulnerabilities for different operating systems starting from January, 1 2010 until December, 31 2010 is shown in Fig. 5 and Fig. 6. Unfortunately, none of the existing vulnerability databases provides the exact dates for patching the issues. As a result, it is not possible to estimate precisely how many vulnerabilities have remained unpatched by the specified date. Thus, in our work we used the following two basic assumptions:

1. We did not take into account number of residual vulnerabilities by the 1st of January 2010;
2. To eliminate fixed vulnerabilities in the Fig. 5 we used an average *days-of-risk* statistics provided in [13] (according to this survey, Microsoft Windows in average has 28.9 days-of risk; Novel Linux – 73.89 days-of risk; Red Hat Linux – 106.83 days-of risk; Apple Mac OS – 46.12 days-of risk and Sun Solaris – 167.72 days-of risk).

We believe that assumptions used do not affect our results as the purpose of this section is not to compare security of different operating systems but to demonstrate effectiveness and feasibility of the proposed intrusion-avoidance technique making use of operating systems diversity.

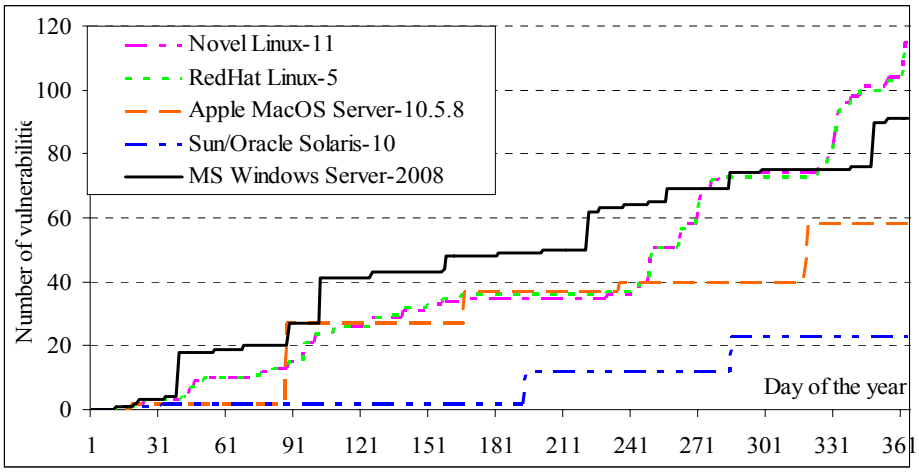

Fig. 5. Cumulative number of vulnerabilities in 2010 for different operating systems.

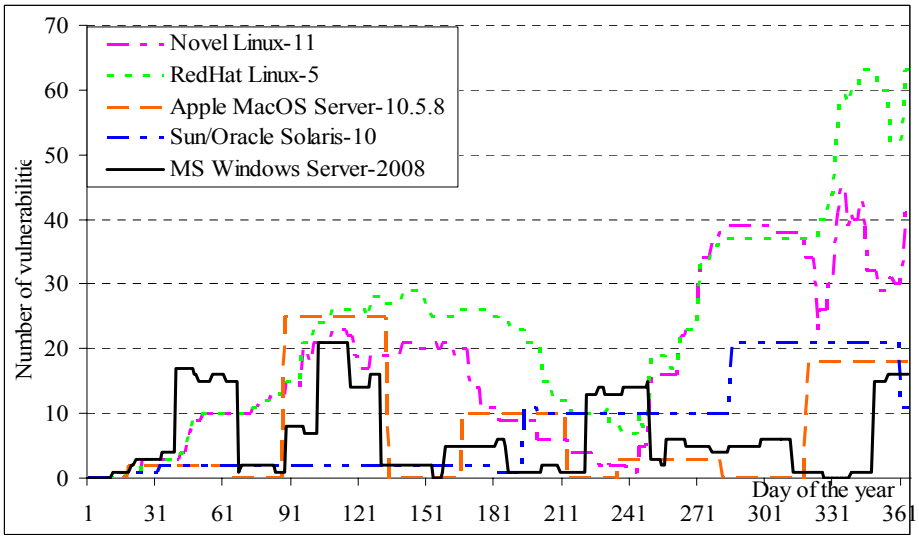

Fig. 6. Number of residual vulnerabilities in 2010 for different operating systems.

3.3 Simulation of dynamic reconfiguration strategy making use of operating system diversity

The results of dynamic operating system reconfigurations performed by the configuration controller taking into account the number of residual vulnerabilities (see Fig. 6) are summarized in Table 3. To simplify our demonstration we took out of consideration severity of different vulnerabilities.

In our simulation we used Novel Linux v.11 as the initial active operating system. Table 3 shows the set of subsequent switches between different operating systems in accordance with the vulnerability discovering and patch issuing process (see Fig. 5). The table also presents the exact dates and periods of active operation of different operating systems. In our simulation, the overall period of the active operation for Novel Linux v.11 was 33 days; for Apple MacOS Server v.10.5.8 – 152 days; for Sun/Oracle Solaris v.10 – 116 days and for MS Windows Server 2008 – 64 days.

Table 3. Operating systems reconfiguration summary.

№	Active operating system	Operation period			Average number of open vulnerabilities
		start date	end date	duration	
1	Novel Linux v.11	01.01.2010	17.01.2010	17	0
2	Apple MacOS Server v.10.5.8	18.01.2010	18.01.2010	1	0
3	Novel Linux v.11	19.01.2010	25.01.2010	7	0.86
4	Sun/Oracle Solaris v.10	26.01.2010	06.03.2010	40	1.8
5	Apple MacOS Server v.10.5.8	07.03.2010	29.03.2010	23	0
6	MS Windows Server 2008	30.03.2010	30.03.2010	1	1
7	Sun/Oracle Solaris v.10	31.03.2010	14.05.2010	45	2
8	Apple MacOS Server v.10.5.8	15.05.2010	16.06.2010	33	0
9	Sun/Oracle Solaris v.10	17.06.2010	13.07.2010	27	1.48
10	MS Windows Server 2008	14.07.2010	01.08.2010	19	1.42
11	Apple MacOS Server-10.5.8	02.08.2010	24.08.2010	23	0
12	Novel Linux v.11	25.08.2010	02.09.2010	9	1.44
13	Apple MacOS Server v.10.5.8	03.09.2010	12.09.2010	10	3
14	MS Windows Server v.2008	13.09.2010	14.09.2010	2	2
15	Apple MacOS Server v.10.5.8	15.09.2010	15.11.2010	62	1.21
16	MS Windows Server 2008	16.11.2010	27.12.2010	42	4.86
17	Sun/Oracle Solaris v.10	28.12.2010	31.12.2010	4	11

As it can be seen from Table 4, the proposed approach to intrusion avoidance allows to hold the minimum possible number of residual vulnerabilities dynamically switching between diverse operating systems. It reduced the average *days-of-risk* to 11.21 and provided 146 vulnerability-free days.

Table 4. Intrusion avoidance summary.

Operating system	Average days-of-risk [13]	Number of vulnerability-free days	Instant number of open vulnerabilities	
			max	avg
Novel Linux v.11	73.89	17	45	17.53
RedHat Linux v.5	106.83	17	63	23.05
Apple MacOS Server v.10.5.8	46.12	134	25	7.288
Sun/Oracle Solaris v.10	167.72	12	21	7.871
MS Windows Server 2008	28.9	27	21	6.466
Diverse intrusion-avoidance architecture with dynamic OS reconfiguration	**11.21**	**146**	**16**	**1.7**

During the remaining period of time the average instant number of the residual vulnerabilities would be equal to 1.7 that is almost four times as less as the best result achieved by Microsoft Windows Server 2008 (6.46 vulnerabilities at once).

4 Conclusions

The proposed intrusion-avoidance architecture that makes use of system component diversity can significantly improve the overall security of the computing environment used to deploy web services. Our work is in line with another recent study [14]. The approach proposed to intrusion avoidance is based on dynamical reconfiguration of the system by selecting and using the particular operating system, web and application servers and DBMS that have the minimal number of the residual (yet unpatched) vulnerabilities taking also into account their severity. Such strategy allows us to dynamically control (and to reduce) the number of residual vulnerabilities and their severity by the active and dynamic configuration of the deployment environment. This helps the architects to decrease the risks of malicious attacks and intrusions. The intrusion-avoidance architecture mainly relies on the cross-platform Java technologies and the IaaS cloud services providing the crucial support for diversity of the system components, their dynamic reconfiguration and maintenance of the spare configurations. The existing vulnerability databases like CVE and NVD provide the necessary up-to-date information for the security risk assessment, finding the least vulnerable configuration and reconfiguration decision making. The purpose of the paper is not to compare the security of various software components. Nevertheless, we would like to mention here that the least number of vulnerabilities in 2010 were disclosed in Sun/Oracle Solaris v.10, whereas the largest number of vulnerability-free days was provided by Apple MacOS Server v.10.5.8. At the same time, mainly due to the least *days-of-risk*, Microsoft ensured the least instant number of residual vulnerabilities in its Windows Server 2008. The vast majority of vulnerabilities of Novel Linux v.11 and RedHat Linux v.5 were vulnerabilities in the Linux core. Thus, they occurred in both operating systems and resulted in the similar curves of the cumulative number of vulnerabilities (Fig. 4). However, Novel spent more efforts on fixing security problems in its Linux distributive that was resulted in the lower *days-of-risk* (see Table 3).

Acknowledgments

Alexander Romanvosky is supported by the EPSRC/UK TrAmS platform grant.

References

1. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, Vol. 1(1) (2004) 11–33
2. Cachin, C. and Poritz, J.: Secure Intrusion Tolerant Replication on the Internet. Proc. International Conference on Dependable Systems and Networks (2002) 167–176
3. Veríssimo, P., Neves, N.F., Correia, M.: The Middleware Architecture of MAFTIA: A Blueprint. Proc. 3rd IEEE Survivability Workshop (2000)
4. Pal, P., Rubel, P., Atighetchi, M., et al.: An Architecture for Adaptive Intrusion-Tolerant Applications. Special issue of Software: Practice and Experience on Experiences with Auto-adaptive and Reconfigurable Systems, Vol. 36(11-12) (2006) 1331–1354
5. Nguyen, Q.L., Sood, A.: Realizing S-Reliability for Services via Recovery-driven Intrusion Tolerance Mechanism. Proc. 4th Workshop on Recent Advances in Intrusion-Tolerant Systems (2010).
6. Chatzis, N., Popescu-Zeletin, R.: Special Issue on Detection and Prevention of Attacks and Malware. Journal of Information Assurance and Security, Vol. 4(3) (2009) 292–300
7. Raggad, B.: A Risk-Driven Intrusion Detection and Response System International Journal of Computer Science and Network Security, Vol. 12 (2005)
8. Valdes, A., Almgren, M., Cheung, S., Deswarte, Y. et al.: An Architecture for an Adaptive Intrusion-Tolerant Server. Proc. 10th Int. Workshop on Security Protocols, Lecture Notes in Computer Science, 2845 ed, Cambridge, UK: Springer (2004) 158–178
9. Powell, D., Adelsbach, A., Randell, B., et al: MAFTIA (Malicious- and Accidental-Fault Tolerance for Internet Applications). Proc. International Conference on Dependable Systems and Networks (2001) D32–D35
10. Ford, R., Thompson, H.H., Casteran, F.: Role Comparison Report – Web Server Role. Security Innovation Inc. http://www.microsoft.com/windowsserver/compare/ReportsDetails.mspx?recid=31 (2005) 37
11. Buyya, R., Broberg, J., Goscinskin, A.M. (Eds.): Cloud Computing Principles and Paradigms. Wiley (2011) 664
12. Strigini, L., Avizienis, A.: Software Fault-Tolerance and Design Diversity: Past Experience and Future Evolution. Proc. 4th Int. Conf on Computer Safety, Reliability and Security (1985) 167–172
13. Jones, J. Days-of-risk in 2006: Linux, Mac OS X, Solaris and Windows. http://blogs.csoonline.com/days_of_risk_in_2006 (2006)
14. Garcia, M., Bessani, A., Gashi, I., Neves, N., Obelheiro, R.: OS Diversity for Intrusion Tolerance: Myth or Reality? Proc. Performance and Dependability Symposium at the International Conference on Dependable Systems and Networks (2011) 383–394