Security Aspects for Business Solution Development on Portal Technology

Ovidiu RĂDUTĂ, Adrian MUNTEANU
Institute for Doctoral Studies within Academy of Economic Studies
Bucharest, ROMANIA
ovidiu.raduta@gmail.com, adrianm21@yahoo.com

In the scope of portal development, in order to talk about security issues, concerns, and solutions, it is necessary to define a few terms: authentication, authorization, Single Sign-On (SSO), confidentiality, integrity, and non-repudiation. Focusing on the scope of what the portal developer and designer need to know, below it will be explained these concepts, considering it is important to define and make a brief analysis of these terms for understanding of achieving the security goals.

Keywords: *Role-Based Access Control, Data Encryption, Data Integrity, Non-repudiation, Grid Computing*

1 Introduction

In recent years, more and more companies have been creating portals in order to offer to their employees, partners and customers to access critical and sensitive information. Most portal developers have applied high level security challenges in this regard. Many vendors have created particular interrelated solutions for some of the security requirements, trying to tie developers to a particular product. Generally, this is a good choice and often works well until the product doesn't work [11]. In this case, the software is no longer supported, or you need to readjust the architecture. Of course that a developer who has been in that boat feels his pain because he has been there. It appears a question: How can portal developers meet their security challenges with a standards-based, open-source security solution? This paper answers that question – first, presenting some essential security concepts, and then by highlighting you standards, techniques and open-source tools that it can use in order to secure your portal solution [8].

Second, this paper will talk about Grid portals, which are an increasingly popular mechanism for creating customizable, Web-based interfaces to Grid services and resources. Due to the powerful, general-purpose nature of Grid technology, the security of any portal or entry point to such resources cannot be taken lightly, in particular if the portal is running inside of the trusted perimeter, such as a Science Gateway running on an SDSC machine for access to the TeraGrid.

2. Core Security Concepts.

First, we will talk about the main concepts regarding portal security

2.1. Authentication

Authentication is the first point in providing access control and this involves validating the identity of a user. Most of cases in a portal environment, authentication may be achieved through user name/password login, validating a user's client certificate, or through validation via smart card or biometric device.

In order to proceed, authentication it is necessary to develop a solution which is usually based on a repository for validating these identities and integrating it with the system. Regarding it, a mutual authentication means to provide the identification for both of parties involved in communication, and this is done using a particular security protocols, such as SSL/TLS. To ensure that the message was sent by the expected sender, it is used a

message origin authentication which is not replayed. [15]
Being one of the most important aspects to providing security, authentication at the portal level will dictate how your application interacts with other enterprise applications and Web services. [7]

2.2.Authorization

When an user is validated, the next step we have to assume is what the user has permission to do. The access control separation in two distinct mechanisms, authentication and authorization, provides a logical separation of first validating identity, and then validating what resources that entity has access to consume or produce. Authorization defines the user permissions, roles, and other useful credentials which are used to permit access to certain portal services. An access control strategy – Role-Based Access Control (RBAC) is provided – (useful because of capability is prevalently in J2EE architectures).
In extension, RBAC (as framework for the authorization management of credentials) is essentially useful in many portals, relational databases, or commercial web-based systems. As it can be seen in Figure 1, a key component of RBAC maps roles to permissions, and maps users to roles. There are also shown views of some authorization mechanisms of the traditional access control – using Access Control Lists (ACLs). So far, large and complex ACLs made links users-permissions, and restricted access to resources by permissions and users. Because there are many cases when the number of permissions is usually very high, an access control lists for discretionary access can be difficult to be kept (with subject -object mappings). So, generally, abstracting the user and the resources permissions is very useful in authorization management. [15]

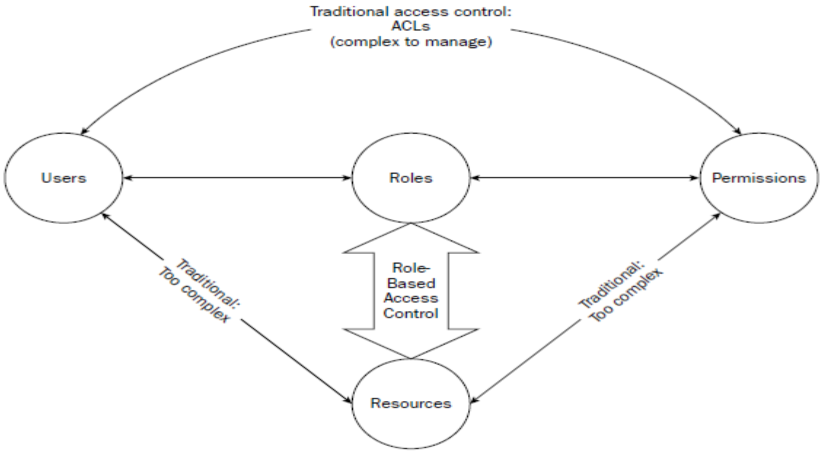

Fig. 1. Key component of RBAC

Although, in a certain organization, mapping these permissions to never-changing roles can happen at one time and users can be assigned and unassigned to these roles during the lifetime of the individual, making the access management control easier to keep. [9]
I consider that the most difficult part of setting up a role-based access control is that an organization must define its roles based on the proper processes. They consider that the technical part of the solution is the easiest phase. They have to provide an enough flexible schema in order to be able to cope the reorganization's rigors. With "Introduction to Role-Based Access Control", NIST (see references) provides an appropriate explanation: "*With role-based access control, access decisions are based on the roles that individual users have as part of an organization. Users take on assigned roles (such as doctor, nurse, teller, manager). The process of defining roles should be based on a thorough analysis of how an organization operates and should include input from a*

wide spectrum of users in an organization." ([10], pp1)

2.3.Confidentiality

Keeping the information secret it is mandatory when sensitive information is sent. Confidentiality is considered the security goal for hiding information and encryption it is an appropriate solution to provide it. With encryption, a plaintext message is modified with a cryptographic algorithm in order to produce a ciphertext message. In the same time, it is a must to have the possibility to decrypt the data using a key. Many different encryption/decryption algorithms, even if it is about a symmetric algorithms (secret-key) or an asymmetric algorithms (public key), can be used to offer different protection levels of sensitive data. Different things like key management for distributing keys, ciphers to use, and cryptographic protocols that provide these services, are needed to create solutions in order to satisfy confidentiality requirements. [12]

Many higher-level protocols like Transport Layer Security (TLS) and Secure Sockets Layer (SSL) (its later version), offer bulk encryption between two points. At this level, the cipher is determined and the encryption key is known at the beginning of the protocol in order to establish a "shared secret" understood on both sides. It is good to know that SSL is a point-to-point protocol which can be used for one-way or two-ways authentication between two points only. Generally, it is enough such a session for environments with a simple client and server in order to protect the data confidentiality in the transmission.

Encryption requirement. If you have some encryption requirements, the portal should be able to put it up, but the portal architect has to foresee that the cryptographic mechanisms will impact the performance. [13]

Requirements. Generally, to have requirements satisfied, it is enough to have a simple SSL/TLS connection between the user and the portal. If not, that means it is necessary to be size encryption between the user, the portal, and all nodes, web services involved and enterprise applications in the solution. (thereby, the solution becoming a bit more difficult).

Secret data. If it is needed to keep some confidential information away from web services (the encryption needs to be directly between the portal and the eventual data source), you will need a shared secret between the portal and the data source (not just bulk encryption between the nodes). Here, XML Encryption, a W3C standard, being quite useful for this. [12]

2.4.Data Integrity

In transit, ensuring a non-altered data is a must. To validate the integrity of a message means to use techniques in order to prove that data integrity is kept. Because on a TCP/IP network could occur some message injection, packet tampering, or IP spoofing, many applications require a digital signatures, a MAC (Message Authentication Codes), or hash algorithms.

The portal's architecture offers integrity challenges regarding users and the portal, and also between portal, web services and the enterprise data source. Beginning from requirement set, SSL/TLS may provide it (message integrity between users and portal). In addition, other standards can be used to achieve integrity. One example is XML Signature (a W3C standard), which provides message integrity in addition to non-repudiation (see the next paragraph). A mechanism that achieves such a solution for Web services is provided by OASIS Web Services Security specification.

2.5. Non-repudiation

The side effect of digitally signing a message is called non-repudiation, which is a security service used when a user has sent a transaction or a message. There are many business-to-business (B2B) systems where non-repudiation is often an essential requirement. Considering the digital signature is based on public key encryption, the sender of the message cannot repudiate successfully the fact that he signed the message. Although non-repudiation is tied in the context of an user signing something, we also can bring this term when discuss about an enterprise with portals, applications, and web services. A portal may sign a portion of its message to a Web service, and a Web service may sign a portion of its messages. A side effect of digitally signing a document is also integrity. Because the signed message is actually the signature of the hash of the message used for proving integrity, many simply view non-repudiation as very strong integrity. XML Signature is a W3C standard used for providing non-repudiation, and is used in other standards, such as the WS-Security standard. The following section describes such standards. [12]

2.6. Auditing

Audit is the process of verifying that security requirements have been satisfied, with corrections suggested where they haven't been met. Essential to effective auditing is that actions are traced and logged through all parts of the system. With web applications, this means that logging of significant operations must happen in the web server, web application and data layer, in addition to the web application itself. Events of interest include: errors, failures, state accesses, authentication, access control and other security checks, in addition to application-specific operations and actions. Care must be taken to protect the integrity of logging and trace data, even (and perhaps especially) in the case of system failures. Logs that are tampered with or destroyed are useless in performing an effective audit. Auditing of log and trace data can either be done manually or it can be automated and often a combination of both is used. Either way, auditing should be done on a regular basis. Automated systems that continually monitor, detect, and in some cases even correct (or at least recommend corrections for) security problems can be particularly useful for maintaining a secure web application. Somewhat related to logging and auditing, Web applications should be careful to ensure that errors or failures somewhere in the system do not introduce security vulnerabilities. Attackers, for instance, are often able to exploit the detailed error information provided by web applications to gain unauthorized access.

2.7. Session Management.

Web-based applications, in contrast to desktop-based application clients, have a challenge with regard to where client-related state information is stored. A desktop application would store state locally on the client machine, but because of the relatively stateless nature of the web browser, client state in web applications tends to be stored remotely on the server. The challenge then becomes securely managing and associating session state with an authenticated client identity. Unlike the other areas mentioned above, session state management is not strictly a security concern. However, the potential for security vulnerabilities in this area as well as its unique relevance to web applications merit its discussion here.

Many web application containers include built-in session management capabilities, and in most cases, it is desirable to leverage this functionality where possible.

When session state management must be built by the web application, care must be taken to ensure that session state can not be tampered with and is securely (i.e., cryptographically) and consistently mapped to an authentication token. From the client perspective, session identifiers are often included in cookies that are automatically saved and presented by the web browser. Such information could also be presented elsewhere in user input data. Care must be taken to protect the integrity and confidentiality of these session identifiers as attackers can use this information to gain unauthorized access to the system (see the attack scenario below). As much as possible web application interfaces should be constructed such that users can keep their session state secure (often this means including sensible logout procedures, among other things).

- **Grid Portal Security Requirements**

So far, we discussed about security concepts (for portal). From now on, we will present the security needs of web applications discussed above.

To their advantage, Grid resources tend to have their own security (authentication and authorization in particular) mechanisms in place, so breaking into a Grid portal, while concerning, may not necessarily allow the attacker access to backend Grid resources. For instance, simply being able to submit jobs through a portal is not useful without proper Grid credentials to authenticate to the actual job execution service. Consequently, the key security challenge of most Grid portals is that at some level, *they manage Grid credentials on behalf of clients.* Compromised Grid credentials are an extremely serious security breach because they allow an attacker to effectively impersonate a valid Grid user until the credentials are revoked or expire. Thus, extra care must be taken in the management of these Grid credentials, which can effectively be viewed as a special kind of session state. The integrity and confidentiality of these credentials must be maintained even in the case of errors or failures. Accesses to the credentials should be logged and monitored continuously for suspicious behavior. Further the credentials, especially if stored on disk must be protected from other users or applications running on the web application server. A compromise elsewhere in the server's software stack should not lead to compromise of user's Grid credentials.

Vulnerabilities of Web Applications

A great challenge in developing secure web applications is that the vulnerabilities in any component of the architecture can often result in compromise of the web application as a whole. For instance, even though the code of a particular web application might be carefully written and free of security holes, vulnerabilities in the web server could still be exploited, causing the secure web application to be hijacked or overridden with a malicious version. Another challenge of the architectural complexity of many web applications is that it is often difficult to configure all of the components correctly and securely. So, even if the components as developed are free of security vulnerabilities, misconfiguration can unwittingly open the web application to compromise.

The Open Web Application Security Project (OWASP) compiled a list of ten of the most common security vulnerabilities afflicting Web applications [17] (and thus Grid portals, which are just a specific type of web application):

- **Unvalidated Parameters** – input contained in web requests is not properly checked (by the application) before being acted on. Attackers can craft parameters to hijack the application or cause it to behave in dangerous, unexpected ways. *Injection Flaws*, *Buffer Overflows*, and *XSS Flaws* are all specific types of *Unvalidated Parameter* vulnerabilities.
- **Broken Access Control** – access control mechanisms work inconsistently or incorrectly, allowing unintended access to resources. This is particularly troublesome for web application administrative interfaces.
- **Broken Authentication and Session Management** – authentication problems can range from weak authentication mechanisms that are easily broken (plain text secrets to retrieve forgotten passwords), to insufficient session protection (exploiting access to one set of session information to gain access to someone else's) to forged sessions or session cookies (allowing session impersonation).

- **Cross-Site Scripting (XSS) Flaws** – involves exploiting an invalidated parameter vulnerability to send a script to the web application that is in turn delivered to and executed by the end user's web browser. **Buffer Overflows** – specially crafted input results in the execution of arbitrary code on the target server. This is particularly problematic if the server is running as root or an administrator account as the malicious code will also have those privileges. In general, Java applications do not suffer from this type of vulnerability (although the JVM itself could).
- **Injection Flaws** – in contrast to the other invalidated parameter attacks, this refers to when injected code or command strings are passed through the web application directly to some backend system. SQL injection attacks are probably the most common.
- **Improper Error Handling** – this type of vulnerability surfaces when error messages displayed to the user in some way reveal details about how the system or application works (the attacker could then exploit this knowledge). This is typically a problem when very detailed stack traces are displayed to the user giving some information of the structure of the code and its operation. Attackers can also probe for inconsistencies in error messages returned ("file not found" vs. "access denied") to gain a better understanding of the application.
- **Insecure Storage** – storage of sensitive data (passwords, account information, etc.) without proper encryption or access control mechanisms. This could be on disk, in a database, or in memory. Usually one of the other exploits is needed to actually gain access to this insecure data.
- **Denial of Service** – when the sheer volume of requests to the web application overwhelms the capacity, denying access to legitimate users. This is usually an even more troublesome problem as web server DoS attacks (like SYN flooding), because it's very hard for web applications to distinguish between legitimate and malicious requests. The complexity of web applications usually means a fairly low threshold of concurrent connections needs to be exceeded to deny access.
- **Insecure Configuration Management** – problems here range from unpatched software to unchanged insecure default settings to outright configuration mistakes caused by incomplete or incorrect understanding of some very complex software. Clearly this is a human problem as much as a software problem, but delivering software that's easy to understand, easy to configure and comes in a secure configuration out of the box would certainly help.

2.8. Key Security Standards

With a significant role in portal development, we will continue with a brief overview of security standards which a portal developer needs to know.

2.8.1. SSL and TLS

Created by Netscape, SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are higher-level encryption protocols that are used to assure data integrity and confidentiality between two points. Also, they can be used for mutual authentication when both parties have digital certificates. Although, the both are very similar (because TLS is based on SSL) and in most cases we simply refer to both protocols as SSL, they will not interoperate (there are subtle differences).

In an easy-going language, SSL with HTTP are called HTTPS Sessions (a process providing confidential web transmission). A portal developer consider that a SSL session can protect confidentiality and data integrity between the user and the portal, but also between the portal and its next communication point. [12]

2.8.2. XML Encryption

Being used to encrypt elements of an XML document, XML Encryption is a W3C standard that operates XML documents and XML element-level confidentiality. It can be used with key exchange algorithms and public key encryption in order to encrypt documents to different parties. A great advantage of XML encryption (unlike SSL, which is decrypted at each point) is that it can be used in solutions with multiple network nodes between the portal and the data source.

2.8.3. XML Signature

XML Signature is a W3C standard that assures the message integrity and non-repudiation of XML documents. Any part of an XML document can be digitally signed – becoming self-validating when it has a public key. XML Signature is based on a public key technology in which the hash of a message is cryptographically signed; this provides integrity and non-repudiation. When portal communicates with Web services, XML Signature has an important role. Because of self-validating, user's credentials can be put on SOAP messages beyond the portal. [14]

2.8.4. SAML

Security Assertion Markup Language (SAML), is an OASIS standard which is used for pass authentication and authorization information between different parts. In a portal environment, a portal can "declare" that it authenticated a user, having in the same time some certain security credentials. A SAML assertion can be digitally signed using XML Signature. It is good to know that SAML can solve significant challenges in Web services security, because signed SAML can travel between different platforms and organizations. Anyone trusting the signer will trust the credential. SAML is an important standard, and many open-source toolkits are available. [12]

3. Grid construction – General principles

This section briefly highlights some of the general principles that underlie the construction of the Grid. In particular, the idealized design features that are required by a Grid to provide users with a seamless computing environment are discussed. Four main aspects characterize a Grid:

- **Multiple administrative domains and autonomy**. Grid resources are geographically distributed across multiple administrative domains and owned by different organizations. The autonomy of resource owners needs to be honored along with their local resource management and usage policies.
- **Heterogeneity**. A Grid involves a multiplicity of resources that are heterogeneous in nature and will encompass a vast range of technologies.
- **Scalability**. A Grid might grow from a few integrated resources to millions. This raises the problem of potential performance degradation as the size of Grids increases. Consequently, apps. that require a large number of geographically located resources must be designed to be latency and bandwidth tolerant.
- **Dynamicity or adaptability**. In a Grid, resource failure is the rule rather than the exception. In fact, with so many resources in a Grid, the probability of some resource failing is high. Resource managers or applications must tailor their behavior dynamically and use the available resources and services efficiently and effectively. [3]

The components that are necessary to form a Grid are as follows:

- **Grid fabric**. This consists of all the globally distributed resources that are accessible from anywhere on the Internet. These resources could be computers (such as PCs or Symmetric Multi- Processors) running a variety of operating systems (such as UNIX or Windows), storage devices, databases, and special scientific instruments such as a radio telescope or particular heat sensor.
- **Core Grid middleware**. This offers core services such as remote process management, co-allocation of resources,

storage access, information registration and discovery, security, and aspects of Quality of Service (QoS) such as resource reservation and trading.

• **User-level Grid middleware**. This includes application development environments, programming tools, and resource brokers for managing resources and scheduling application tasks for execution on global resources.

• **Grid applications and portals.** Grid applications are typically developed using Grid-enabled languages and utilities such as HPC++ or MPI. An example application, such as parameter simulation or a grand-challenge problem, would require computational power, access to remote data sets, and may need to interact with scientific instruments. Grid portals offer Web-enabled application services, where users can submit and collect results for their jobs on remote resources through the Web.

In attempting to facilitate the collaboration of multiple organizations running diverse autonomous heterogeneous resources, a number of basic principles should be followed so that the Grid environment:

• does not interfere with the existing site administration or autonomy;

• does not compromise existing security of users or remote sites;

• does not need to replace existing operating systems, network protocols, or services;

• allows remote sites to join or leave the environment whenever they choose;

• does not mandate the programming paradigms, languages, tools, or libraries that a user wants;

• provides a reliable and fault tolerant infrastructure with no single point of failure;

• provides support for heterogeneous components;

• uses standards, and existing technologies, and is able to interact with legacy applications;

• provides appropriate synchronization and component program linkage. [3]

As one would expect, a Grid environment must be able to interoperate with a whole spectrum of current and emerging hardware and software technologies. An obvious analogy is the Web. Users of the Web do not care if the server they are accessing is on a UNIX or Windows platform. From the client browser's point of view, they 'just' want their requests to Web services handled quickly and efficiently. In the same way, a user of a Grid does not want to be bothered with details of its underlying hardware and software infrastructure. A user is really only interested in submitting their application to the appropriate resources and getting correct results back in a timely fashion. An ideal Grid environment will therefore provide access to the available resources in a seamless manner such that physical discontinuities, such as the differences between platforms, network protocols, and administrative boundaries become completely transparent. In essence, the Grid middleware turns a radically heterogeneous environment into a virtual homogeneous one.

The following are the main design features required by a Grid environment.

• Administrative hierarchy. An administrative hierarchy is the way that each Grid environment divides itself up to cope with a potentially global extent. The administrative hierarchy determines how administrative information flows through the Grid.

• Communication services. The communication needs of applications using a Grid environment are diverse, ranging from reliable point-to-point to unreliable multicast communications.

The communications infrastructure needs to support protocols that are used for bulk-data transport, streaming data, group communications, and those used by distributed objects.

• The network services used also provide the Grid with important QoS parameters such as latency, bandwidth, reliability, fault-tolerance, and jitter control.

• Information services. A Grid is a dynamic environment where the location and types of services available are constantly changing. A major goal is to make all resources accessible to any process in the system, without regard to the relative location of the resource user. It is necessary to provide mechanisms to enable a rich environment in which information is readily obtained by requesting services. The Grid information (registration and directory) services components provide the mechanisms for registering and obtaining information about the Grid structure, resources, services, and status.

• Naming services. In a Grid, like in any distributed system, names are used to refer to a wide variety of objects such as computers, services, or data objects. The naming service provides a uniform name space across the complete Grid environment. Typical naming services are provided by the international X.500 naming scheme or DNS, the Internet's scheme.

• Distributed file systems and caching. Distributed applications, more often than not, require access to files distributed among many servers. A distributed file system is therefore a key component in a distributed system. From an applications point of view it is important that a distributed file system can provide a uniform global namespace, support a range of file I/O protocols, require little or no program modification, and provide means that enable performance optimizations to be implemented, such as the usage of caches. [3]

• Security and authorization. Any distributed system involves all four aspects of security: confidentiality, integrity, authentication, and accountability. Security within a Grid environment is a complex issue requiring diverse resources autonomously administered to interact in a manner that does not impact the usability of the resources or introduces security holes/lapses in individual systems or the environments as a whole. A security infrastructure is the key to the success or failure of a Grid environment.

• System status and fault tolerance. To provide a reliable and robust environment it is important that a means of monitoring resources and applications is provided. To accomplish this task, tools that monitor resources and application need to be deployed.

• Resource management and scheduling. The management of processor time, memory, network, storage, and other components in a Grid is clearly very important. The overall aim is to efficiently and effectively schedule the applications that need to utilize the available resources in the Grid computing environment. From a user's point of view, resource management and scheduling should be transparent; their interaction with it being confined to a manipulating mechanism for submitting their application. It is important in a Grid that a resource management and scheduling service can interact with those that may be installed locally.

• Computational economy and resource trading. As a Grid is constructed by coupling resources distributed across various organizations and administrative domains that may be owned by different organizations, it is essential to support mechanisms and policies that help in regulate resource supply and demand [1], [2]. An economic approach is one means of managing resources in a complex and decentralized manner. This approach provides incentives for resource owners, and users to be part of the Grid and develop and using strategies that help maximize their objectives.

• Programming tools and paradigms. Grid applications (multi-disciplinary apps.) couple resources that cannot be replicated at a single site even or may be globally located for other practical reasons. A Grid should include interfaces, APIs, utilities, and tools to provide a rich development environment. Common scientific languages such as C, C++, and Fortran should be available, as should application-level interfaces such as MPI and PVM. A variety of programming paradigms should be supported, such as message passing or distributed shared memory. In addition, a suite of numerical and other commonly used libraries should be available.

- User and administrative GUI. The interfaces to the services and resources available should be intuitive and easy to use. In addition, they should work on a range of different platforms and operating systems. They also need to take advantage of Web technologies to offer a view of portal supercomputing. The Web-centric approach to access supercomputing resources should enable users to access any resource from anywhere over any platform at any time. That means, the users should be allowed to submit their jobs to computational resources through a Web interface from any of the accessible platforms such as PCs, laptops, or Personal Digital Assistant, thus supporting the ubiquitous access to the Grid. The provision of access to scientific applications through the Web (e.g. RWCPs parallel protein information analysis system [16]) leads to the creation of science portals. [3]

4. Conclusion

Nowadays, security is a hot topic and it is obvious that data needs to be protected. Taking into consideration that architects, portal administrators and, of course, developers are faced with a variety of factors when planning for portal application security, it is very important do not forget the significant role of security aspects and required standards in portal development.

According to the American Heritage Dictionary, a portal is “a doorway, entrance, or gate, especially one that is large and imposing”. The intent behind such structures is really one of security, to allow the welcome visitors through, while keeping unwelcome intruders out. From a technological perspective, a portal is something that provides a convenient entry point to resources, applications or content located elsewhere. Early *Web portals* were typically web sites with search engines or indexes to other content on the World Wide Web.

Since all of the content accessible through these web portals was publicly available anyway, everyone was welcomed in and security was barely a concern. In Grid computing, the resources of interest are not websites, but data and computational resources, services and applications. Thus, the goal of a *Grid portal* is to provide a convenient entry point to these Grid resources, typically via a Web-based front-end. While many Grid portals expose relatively general purpose functionality like launching jobs for remote execution or retrieving remotely-stored data, they can also include application specific interfaces customized for a particular domain.

Security gains prominence in Grid portals largely because of the nature of the Grid resources they expose. Many Grids link together powerful clusters of computational power and large scale data stores containing confidential, classified or proprietary information. A compromised Grid portal could allow an attacker to harness these powerful computational resources to launch a large scale attack elsewhere on the Internet or to gain user access to probe for privilege escalation or root compromise, for example.

Generally speaking, “*We’ve done tremendous work to secure computers but nothing to secure the human operating system. To change human behaviour, you need to educate and train employees, not just once a year but continuously. Like you continually patch computers and applications, you’re continually training and patching human operating systems.*” ([5], pp.1)

References

[1] Buyya R, Abramson D, Giddy J. Economy driven resource management architecture for computational power grids. *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2010)*, Las Vegas, NV, 2010. CSREA Press: Athens, GA
[2] Buyya R. Economic-based distributed resource management and scheduling for Grid computing. *PhD Thesis*, Monash University, Melbourne, Australia, April 2010.
[3] David Del Vecchio, Victor Hazlewood and Marty Humphrey, "Evaluating Grid Portal Security" *Department of Computer Science, University of Virginia*, San Diego, 2009
[4] Hazen A.Weber, "Role-Based Access Control: The NIST Solution", *SANS Inst.*, 08
[5] Lance Spitzner, "Target: The Human", *Information Security Magazine*, May 2011
[6] Mark Baker, Rajkumar Buyya and Domenico Laforenza, "Grids and Grid technologies for wide-area distributed computing", *School of Computer Science, University of Portsmouth*, Mercantile House, Portsmouth, U.K. and *Grid Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering*, The University of Melbourne, Melbourne, Australia
[7] M. Velicanu, D. Litan, L. Copcea (Teohari), M. Teohari, A.M. Mocanu (Virgolici), I. Surugiu, O. Raduta, "Ways to Increase the Efficiency of Information Systems", *The Proc. of the 10th WSEAS Internat. Conf. on artificial Intelligence, Knowledge Engineering and Databases, Cambridge, UK*, 2011.
[8] M. Velicanu, D. Litan, I. Surugiu, O. Raduta, A.M. Mocanu (Virgolici), "Information Technology Standards – a Viable Solution to Reach the Performance", *International Conference on TECHNOLOGY POLICY and LAW (TPL '11), Brasov, RO*, 2011.
[9] D. Litan, L. Copcea (Teohari), M. Teohari, A.M. Mocanu (Virgolici), I. Surugiu, O. Raduta, "Information Systems Integration, a New Trend in Business", *APPLICATIONS of COMPUTER ENGINEERING (ACE '11), Canary Islands, ES*, 2011.
[10] National Institute of Standards and Technology, Computer Security Resource Center, "An Introduction to Role-Based Access Control, in *ITL Computer Security Bulletin.*, Dec. 2010.
[11] Ovidiu Raduta, Adrian Munteanu, "Business Intelligence Solutions - Security Components", *The 10th International Conference on Informatics in Economy*, 2011
[12] W. Clay Richardson, Donald Avondolio, Joe Vitale, Peter Len, Kevin T. Smith, "Professional Portal Development with Open Source Tools: Java™ Portlet API, Lucene, James, Slide", *Wiley Technology Publishing*, 2009
[13] http://www.sans.org
[14] www.rsa.com
[15] http://csrc.nist.gov/publications
[16] http://www.rwcp.or.jp/papia/
[17]http://www.owasp.org/documentation/topten.html
[18]http://en.wikipedia.org/wiki/Web_porta

Ovidiu Răduță has graduated the Academy of Economic Studies (Bucharest, Romania), Faculty of Cybernetics, Statistics and Economic Informatics in 2006. He holds a Master diploma in Informatics Security (Master Thesis: IT Software in banks. Security Issues) from 2008 and currently, he is a Ph.D. Candidate in Economic Informatics with his Doctor's Degree Thesis: Bank System's Process Optimizing. In present, he is ISTQB – Advanced Test Analyst certified and he works as Senior Test Analyst with 3+ years testing experience in Raiffeisen Bank Romania (6+ years banking projects experience). His research activity can be observed in many international proceedings (papers ISI proceedings). His scientific fields of interest include: Test management, Test Techniques, Databases processes, Middleware Products, Information Systems and Economics.

Adrian Munteanu has graduated the Academy of Economic Studies (Bucharest, Romania), Cybernetics, Statistics and Economic Informatics in 2001. Currently, he is a Ph.D. Candidate in Economic Informatics with his Doctor's Degree Thesis: DataWarehouses - Business Support. In present, he is Advanced Resolution Engineer with 12+ years experience in database and Enterprise solutions field at Oracle Corporation. His research activity can be observed in many international proceedings (papers ISI proceedings) published by now. His scientific fields of interest include: Business Intelligence, Datawarehouse Modelling and Enterprise Resource Planning implementation.