Global Information Assurance Certification Paper

Auditing Borland's J2EE Application Server: An Auditor's Perspective

by Brenton Camac
for GSNA Practical Assignment (v2.1)

March, 2004

Abstract

This paper documents an independent audit of an in-production business system.

The focus of the audit is confined to the system's J2EE AppServer component only. The AppServer in this system is Borland's Enterprise Server, version 5.2.

The purpose of this audit is to assess the risk facing the system at this layer (the AppServer layer) and provide the organization's management with a plan to resolve that risk in an effective way.

The audit activity was divided into four parts.

Part I – The Audit Research. The audit's subject, scope and purpose were defined in detail. Within the boundary of these parameters a risk analysis was undertaken to identify the business assets of the system and their value to the organziation. Vulnerabilities specific to the AppServer part of the system which could threaten these assets were then identified by a threat analysis.

Part II – The Audit Ruler. Audit controls based on the threats identified by Part I were devised. These controls allow the system to measured to determine its degree of exposure to threats identified in Part I.

Part III – The Audit Evidence. The system was audited against the ruler defined in Part II. This identified specific vulnerabilities in the system that are exposed and could be exploited. (Note. Not all of the collected audit evidence is documented in this part, but only those items which are referred to later in Part IV.)

Part IV – The Audit Report. The findings of the audit are documented and summarised. The risks facing the system, the recommended remedial actions that should be taken, and the costs associated with the remedial actions are all presented. An executive summary of these findings is also provided for management.

Executive Summary

On February 9 and 10, 2004, an audit of the system was conducted to determine the amount of risk it faced from security threats.

The audit's scope was limited to consider risks posed to the system at the AppServer layer only. This is but one layer of many in the system; other layers include the network and operating system. Of all the layers comprising the system, the AppServer is the one with the least amount of security related resources and wisdom currently available; primarily due to its being the least mature technology in the system's infrastructure stack. Consequently, determining the risk facing the system at this layer is more difficult than at other layers and therefore the reason for considering the AppServer layer only during the audit.

The audit measured the risk facing the system by evaluating it against security controls (see Part III). These controls were developed during the preparatory phases of the audit (see Part II) and were based upon threats identified by a threat analysis also performed as part of the audit (see Part I). The threat analysis examined the organization's use of the system as a business tool from a business perspective and identified the system's primary assets to be the data it manages. The most serious threats to this asset are a breach of its confidentiality or integrity. Such threats have the potential to impact the business of the organization by causing operational losses, loss of reputation, loss of competitive standing in the marketplace, regulatory penalties and fiduciary liabilities.

For example, disclosure of the data to unauthorized individuals (a breach of confidentiality) could result in regulatory fines for failure to protect the privacy of the individuals described in the data; result in revocation of special business licenses issued by the government necessary for operating in this particular market; and the loss of confidence from the public and business partners caused by a damaged public image / reputation. Likewise, if data integrity was compromised this could result in immediate operational losses including lost revenue as the application is offline while the data's integrity is restored (scrubbed of unauthorized changes), plus the cost of performing data verification, which would be incurred by the organization.

Hence, costs to the organization resulting from breached security can be significant. The magnitude of the costs in some cases would be compartible with the revenue generated by the system for the business. Therefore, some of these threats could have catastrophic impact on the organization's business.

2.1.1.1 The Degree of Compliance

The level of compliance found during the audit is shown below. The system's compliance with these controls indicates the degree to which the system is exposed to security exploits and therefore at risk.

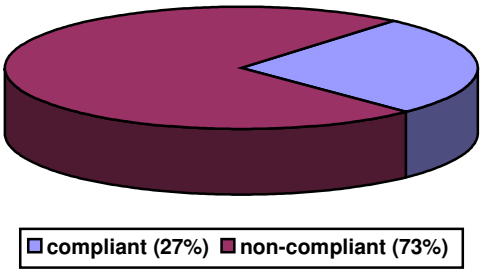

Figure 1. System's Compliance with Technical Controls (15 total)

Technical controls are concerned with the AppServer product and its configuration.

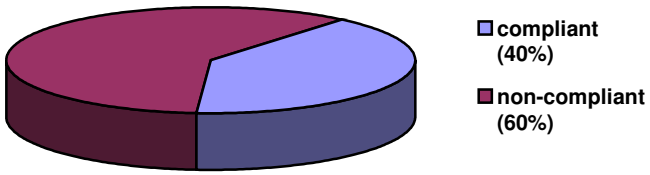

Figure 2. System's Compliance with Operational Controls (5 total)

Operational controls are concerned with the personnel, procedures and practices employed to support the AppServer in this environment.

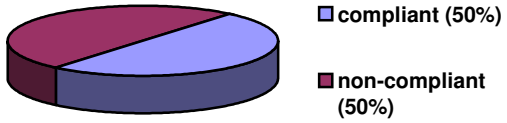

Figure 3. System's Compliance with Management Controls (5 total)

Management controls are concerned with the needs of the system being represented and supported at the management level of the organization.

2.1.1.2 The Cost to Remedy

The estimated cost to bring the system into compliance is divided into three disjoint categories: critical risks, non-critical risks, and on-going costs:

Category	Estimated Effort	Approximate Cost
Critical risks	11 days	$3,800
Non-critical risks	22 days	$7,800
On-going costs to maintain security	10 hours / week	$453 / week

2.1.1.3 Recommendations

It is recommended that critical risks be addressed immediately. These are instances where the system is currently exposed to threats, which if exploited could have significant impact on the assets of the system. The critical risks are identified by five technical controls and one operational control.

Once the critical risks have been resolved, non-critical risks should be addressed. These are instances where the system is also exposed to threats but the level of risk to the system is lower. The non-critical risks are identified by six technical controls, two operational controls and one management control.

To ensure the system remains secure, ongoing effort must be devoted to the security of the system. For greatest efficiency, that effort should be guided by organizational procedures, which will have been developed as part of the non-critical remedies. This ensures that the security of the system is maintained as it evolves over time in response to new requirements and new threats.

Part I

Research in Audit, Measurement Practice And Control

Table of Contents

1 Subject of the Audit

The system to be audited is a J2EE[1] Application Server, specifically Borland's Enterprise Server (BES) version 5.2.

This J2EE application server is used to host a custom J2EE application that performs core, day to day business functions for an organization. The business functions reference and manipulate confidential data, which resides in a database within the organization. The data is confidential and proprietary to the organization and therefore maintaining its confidentiality and integrity is of great concern to the organization.

Users of the application are employees of the organization who interact with it through a web-based interface hosted by the AppServer's internal web server. The majority of users are located on the internal network, but some are external and access it via the Internet. Administrators of the system access the AppServer from the internal network using BES admin tools that communicate using IIOP[2].

A separate LDAP server on the internal network hosts authentication and authorization information about users of the AppServer. Also, a separate RDBMS server on the internal network hosts the data used by the application.

The AppServer is hosted on a machine running Linux (RedHat 7.2) located within the organization's DMZ network. The DMZ network is bridged to the Internet via an external firewall and bridged to the internal network via an internal firewall.

The relationship of the system to these elements of the organization's IT environment is shown below (see Figure 4).

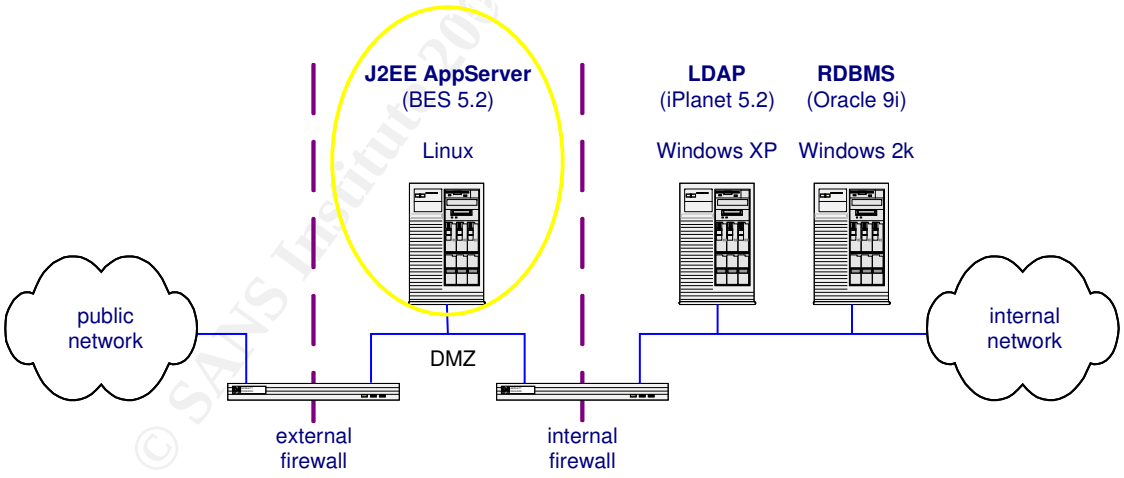

Figure 4. Context of the System being audited

[1] J2EE is the Java 2 Enterprise Edition of the Java programming environment.
[2] IIOP is the Internet InterORB Protocol defined by the Object Management Group (OMG) to allow communication between CORBA ORBs.

1.1 A Primer on J2EE Application Servers

J2EE Application Servers are a standardized platform for hosting applications. The platform extends the basic Java 2 Standard Edition platform (J2SE) by adding various services and integrating them together. The services mandated for the J2EE platform and available to applications include:

- a Transaction service based on the Java Transaction API (JTA)
- a Naming Service based on the Java Naming Directory Interface API (JNDI)
- a Security Service based on the Java Authentication and Authorization Service (JAAS)
- a Database Connectivity based on the Java Database Connection API (JDBC)
- a Messaging service based on the Java Messaging API (JMS)
- and others.

This provides a richer, more standardized environment for large-scale, Java-based enterprise business applications.

Furthermore, J2EE AppServers are generally considered part of an organization's IT infrastructure rather than a part of any application. Therefore, their configuration and maintenance are typically the responsibility of the administration staff rather than the application development staff.

More details about the AppServer concept are explained in the J2EE FAQ[3] produced by Sun Microsystems.

The AppServer used by the system being audited is Borland's Enterprise Server (BES) version 5.2.

2 Scope of the Audit

The scope of this audit is limited to the J2EE AppServer component of the system.

Other elements comprising the system include the application hosted by the AppServer, the operating system of the machine hosting the AppServer, the underlying communication network, and the RDBMS and LDAP servers used by the AppServer and ultimately the application. Each of these could be the subject of their own security audit.

Choosing to audit the AppServer alone allows for a more in-depth analysis and layered approach to be taken with the system's security (see The Security Mind[4]). The tight internal cohesion amongst services of the AppServer combined with the well-defined, loose external coupling of it to other elements of the system make it a natural unit of study within the system's architeture.

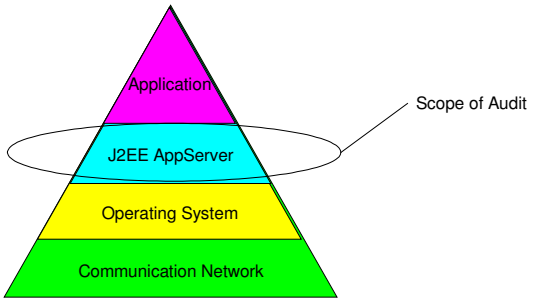

Figure 5. Audit's scope in the context of other elements

[3] Sun. "The J2EE FAQ"
[4] Day, p89-92. "Inside the Security Mind: Making the Tough Decisions"

To specifically delineate what is and what is not within the scope of this audit, the following items are mentioned.

Within Scope:
The configurations of BES's (J2EE) services used by the application are within the scope of this audit. For instance, the Java security policy applied to hosted J2EE components is within scope. However the correctness of those service's implementations is not. This choice is partly to keep the size of the audit manageable and partly because such issues are common to all uses of BES. Focusing on areas specific to this application produces results that are particular to this organization and therefore of greater value.

Also within the scope of this audit is the administration functionality of BES. For instance, access control to the administration domain is considered.

Also within the scope of this audit are the operational activities performed by the admin staff to maintain BES, as well as management's support of those activities.

Finally, the audit scope is restricted to the in-production incarnation of the system as it existed on February 9 and 10, 2004. Like most enterprise systems, this one has and will continue to evolve over time. But this report is static. Therefore the audit must address the system as it existed at a particular period in time.

Beyond Scope:
As mentioned above, not within scope are the operating system and communications network services. Therefore, issues such as unauthorized access to the host or holes in the firewall configuration are not considered. These issues should be the subject of a separate audit.

Also outside of the audit's scope is the application itself. Fortunately, the boundary between application and AppServer is well defined in the J2EE architecture: an application is that functionality deployed with EAR, WAR, or JAR files. Hence, possible flaws in the implementation of the application are not considered; such as SQL injection vulnerabilities, etc. The security of the application should likewise be the subject of a separate audit.

3 Purpose of the Audit

The purpose of the audit is to assess the level of risk facing the system.

The system was placed into production only recently and has not yet been analyzed to determine its exposure to risk. Yet this system executes critical day-to-day business operations for the organization and the information it accesses within the organization is highly confidential and proprietary. A breach of this confidentiality, or loss of its integrity, or loss of system availability could have a serious impact on the organization's business. Therefore it is prudent to determine what level of risk is being faced by such a high value system.

The audit will assess the level of risk and communicate this to management by the following approach:

1. Identify the business assets of the system
2. Identify the potential impact to the organization's business if the security of these assets were compromised
3. Research the role played by the AppServer within the system
4. Identify vulnerabilities of the AppServer which could be used to threaten the system's business assets
5. Develop security controls to prevent or mitigate such vulnerabilities from being exploited
6. Develop a ruler (a checklist of items) to formally and objectively determine the AppServer's exposure to such threats

7. Report the outcome of this assessment, identifying root causes of problems and recommending remedies in a prioritized and actionable plan that includes estimated costs

This information will inform management as to what risks the system faces, what is the relative importance of each risk, what actions can be taken to remedy those risks, how much it might costs to implement those remedies.

Based on this information, management will be able to make better informed decisions about the costs of providing and maintaining the system's security and thus the cost of protecting the business assets of the system.

4 Assessment of Risk

Knowing what risks the system faces is important and will become the basis upon which action will be developed to reduce the risk. An audit is one way to assess the risk.

4.1 General Risks to the System

Before undertaking an analytical assessment of risk facing the system, it is worth noting some peculiar (i.e. special) factors about J2EE AppServers that contribute risk to the system.

The newness of J2EE AppServers
The AppServer is a relatively new technology compared with other elements of the system. Consequently, AppServer technology tends to be more concerned with development issues over production issues. For example, most AppServer product installations are optimized by default for development environments rather than production ones. Hence security is usually disabled, most other services enabled (such as web servers, etc), and default accounts provided. But the goals of a development environment which center around ease of use are generally at odds with the goals of a production environment which are more concerned with managing control.

Furthermore, since AppServers are a relatively new technology, a common body of knowledge on how to harden them for production use has not yet developed to the same degree has other technologies of the system such as operating systems (see for example YASSP[5] and Titan[6]).

Hence, AppServers introduce a considerable amount of risk into the system when initially installed and require substantial individual effort and expertise from the administration staff to reduce.

The newness of J2EE technology also means that the J2EE specification and hence vendor's products are still evolving/maturing. As these implementations change, often substantially from one release to another, this will cause change to occur at this layer in the system. Compounding this is change resulting from deployment of new applications to the AppServer, which require the AppServer's configuration to be modified to suit the demands of each new application. This change to a system in production leads to greater potential for introducing vulnerabilities or unintentionally exposing previously secured vulnerabilities. This increases the risk to the system.

The general purpose nature of J2EE AppServers
The general-purpose nature of J2EE AppServers means that many functions that previously were implemented directly in an application are now configured within an AppServer. Examples include naming

[5] Chouanard. "YASSP: Yet Another Solaris Security Package"
[6] Powell. "Titan Security Toolkit"

services, access control frameworks and policies, database connection management, etc. As the AppServer is now part of the system infrastructure, it typically falls under the responsibility of the administration staff. Thus administration staff must become aware of what these services do in order to configure them correctly and maintain their secure state. But these services are generalizations taken from the programming domain and so it is unreasonable to expect administration staff to be already trained in these areas. So if administration staff is not trained on this new technology it can result in this new layer of the system becoming neglected from a security perspective which would increase the risk facing the system.

The proximity of J2EE AppServers to core business assets
The J2EE AppServer is often the highest layer of infrastructure in a system. It is thus the innermost layer of defense around the application still under the control of the administrator. Yet, for the reasons above, it maybe under-protected from threats.

Being a layer close to the business operations and business resources means there is the least amount of abstraction and generalization employed. Hence if access is gained at this layer then it is access to the business data and operations themselves. Therefore, this layer is a natural target for attack and thus faces extra risk in this regard.

4.2 An Analytical Approach

The foregoing was a brief survey of some of the factors particular to AppServers that can increase the risk to the system. In this section, a more analytical approach is taken to assess the risks facing the system, which will be the basis of subsequent work in this audit.

4.2.1 Terminology

The terminology used in this analysis builds upon the concepts from a presentation titled "Risk Assessment and Management" by Jeff Kimmelman[7].

A summary of this terminology is shown in Figure 6.

Threat = a Vulnerability of the system which can be Exploited by a Threat Agent

Attack = an attempt to Exploit a Vulnerability by a Threat Agent

Successful Attack = the Exploit of an Exposed Vulnerability by a Threat Agent

Likelihood = the probability of a Vulnerability being exploited by a Threat Agent

Risk = the Likelihood and impact of a threat on a business asset of the system

Figure 6. Terminology used in this report for Risk Assessment

A vulnerability is something in the system that can be exploited. Remember that the definition of system in this audit extends beyond the technology to include operational and management matters as well.

[7] Kimmelman. "Risk Assessment and Management"

A vulnerability maybe caused by a flaw or weakness in the system.
A flaw is some imperfection in the system. It may exist in the design, implementation or execution of the system. A flaw may be concealed or exposed, known or unknown. An example of a flaw is leaving default user accounts installed on a production system. This would be a flaw in the runtime configuration chosen for the system.

A weakness is some aspect of the system that is insufficient to resist an expected attack. A key point here is that the weakness is measured and defined with respect to the force of an attack. So, for instance, sending passwords between a user and the system using an 40-bit RC2 export grade SSL cipher may be sufficient (i.e. strong enough) to protect its confidentiality against a casual attack, but would not be strong enough to resist a more determined attack.

Flaws and weakness are not always exploitable. There may exist flaws and weakness that do not pose any threat to a system's assets. However, a vulnerability is always exploitable – i.e. it can be used to threaten a system's assets.

An exploit is a methodology (or recipe) for taking advantage of one or more vulnerabilities. Exploits are repeatable and always yield the same result each time they are applied to the same vulnerability. An exploit is the basis for, but not the same as an attack.

An attack is the prosecution of an exploit by a threat agent. An attack may or may not be successful, depending upon whether the vulnerability is exposed to a threat agent. If the vulnerability is exposed to a threat agent then an attack will be successful, if not then it won't. This assumes that the threat agent prosecutes the exploit flawlessly – i.e. the ideal case.

Notice that the definitions thus far have not mentioned probability. This aspect will now be considered.

The exposure of a vulnerability is binary: either the vulnerability is exposed or it is not. Therefore its exploitability is also binary. If the vulnerability is exposed, then by definition, it can be exploited. However the certainty that a threat agent will exploit a vulnerability is not binary. Over a given period of time (in the future) there is only a likelihood of the event occurring. Only over an infinite amount of time could it be considered certain to occur.

Contributing factors to this probability include

- The size of the threat agent population. If there are many threat agents capable of exploiting a vulnerability, then the probability that one of them will exploit it is higher than if there are fewer threat agents.

- The period of time of exposure. If the period of exposure is short, then the agent has less opportunity - perhaps not even sufficient opportunity - to exploit the vulnerability. Whereas the longer the vulnerability is exposed the more likely it becomes that a threat agent will exploit it.

If a threat agent exploits a vulnerability, then it will have an effect on the system that will generally result in some impact to the organization's business operations. The impact may be on one or more of the following aspects of the business (these categories are also from the presentation given by Jeff Kimmelman).

Operational: Lost time, production or delivery
Reputation: Loss of customer or consumer confidence
Competitive: Reduction of market advantage
Regulatory: Legal liability
Fiduciary: Fiduciary liability

The degree of impact to the business and the likelihood of it occurring are referred to as the risk facing the system. This definition of risk is broader than the typical 'probability' use of the term. Here it is used in

the business sense; a risk specifies what asset is threaten, what value that asset has to the business, and what is the likelihood of it being affected by the threat. This definition attempts to quantify the likelihood of actual losses during a given period of time.

4.2.2 Risk Assessment Methodology

Thus, to assess the risk of the system is to :

- identify the assets of the system,
- estimate the business value of those assets
- identify what vulnerabilities of the system could threaten those assets,
- estimate the likelihood of each vulnerability being exploited

This analysis will provide some measure of potential risk facing the system. But it will take an audit of the system to determine the degree to which vulnerabilities identified here are exposed by the system.

4.2.3 Assets of the System and their Value

The business assets of this system are:

- the **data** of the system, and
- the **application** which accesses and manipulates the data

Of these two assets, the organization considers the data to be the more important asset.

The system's data contains financial and other personal information about individuals entrusted to the organization that is used by the organization in its business. (Details of what that business is are not included in this report to protect the confidentiality of the organization).

The privacy of the data is such that laws exist to govern its access and use. In addition, the organization must be licensed by the government for them to conduct business using this information. Those licenses are concerned with preserving the confidentiality of this information.

If the confidentiality of the data were breached, the organization could face substantial fines from the government and even the loss of their license. This would have a regulatory and fiduciary impact on their business.

Furthermore, if their customers and/or business partners learnt of the breach then this could cause them to loose confidence in the organization. This would impact the reputation and competitiveness of the business.

Of these possible consequences, the organization considers the regulatory and fiduciary impacts to be the most serious.

Likewise, the integrity of the data is important to the organization for similar reasons. However a loss of integrity would not have the same impact as loss of confidentiality.

Finally, the availability of the system is important as this is used for critical, day-to-day operations of the business, which contribute directly to the organization's revenue. Hence a loss of system availability would cause an operation impact to the business. However the potential loss of revenue caused by a typical outage in operations is estimated by the organization to be substantially less than losses that might arise from a breach of data confidentiality or integrity.

Thus, the organization's business relies on maintaining the confidentiality of the data and to a lesser extent its integrity. It is impacted by system availability, but this is not considered as important as the other two aspects.

Valuing these assets and estimating the potential cost to the organization of their loss is not undertaken by this audit. Rather, the relative importance of these assets as mentioned above will be used to direct the audit's priority.

4.2.4 Vulnerabilities which could threaten the identified assets

Having identified the principal assets of the system, the risk analysis will now focus on identifying what vulnerabilities of the system could threaten those assets if exploited.

A two-fold approach is taken to identify such vulnerabilities. First, existing resources are sought which may provide this information. Ideally, these resources will document common vulnerabilities for the AppServer and would include checklists or practices for securing the product. The second step will be to supplement this information by studying the AppServer's architecture and its use by this particular system.

4.3 Current State of Practice

Based on these assets of the system and the categories of vulnerabilities identified for them, a search was conducted to determine if there are pre-existing resources which could be used to assist with the audit activity. Specifically, if there were resources that identify and describe vulnerabilities that are relevant to this audit and if so how they can be secured in the Borland Enterprise Server product.

This search began with the focus on BES-specific resources. The focus was then progressively broadened to include more generic material. The sources used in the search and the results obtained are detailed below.

4.3.1 BES-specific Resources

The following searches were made to locate existing BES checklists or other resources that would assist in securing or auditing BES.

Search term: {"Borland Enterprise Server" Security Audit} submitted to Google. Matches found: 102.
Reviewed all candidates, but did not find the information sought.

Since Borland Software Corporation was also known as Inprise Corp for a few years (circa 2000), its Application Server was abbreviated as IAS. This spelling variant was therefore searched for as well, eventhough it corresponds to the previous version of the product (i.e. version 4.x).

Search term: {+Inprise App Server Security Audit} submitted to Google. Matches found: ~308.
The highest ranked 60 candidates were inspected, but did not provide the information sought.

Since no BES specific audit resources appear to exist, the search was refocused to find information about BES security vulnerabilities that could be used to develop audit materials.

Search term: {security vulnerability site:borland.com} submitted to Goolge. Matches found: 6.
This search was restricted to the vendor's site. Most of these matches were either to product documentation addressing product patches, or to articles that were not relevant to this particular audit.

Search term: {security OR vulnerability group:borland.public.bes.*} submitted to Goolge. Matches found: ~280.
This search was restricted to newsgroups setup for the discussion of this product (borland.public.bes.*). The majority of these matches came from the groups borland.public.bes.appserver.security and borland.public.bes.appserver.ejb which are both highly relevant to this subject. Each match's synopsis was reviewed but none indicated any relevance.

Reviewed the BES 5.2 Security Service FAQ[8]
Much of this information applies to deprecated functionality and how to use the new functionality. This was mostly for developers wishing to customize the security aspects of BES.

Reviewed the Tech Note: "How to secure a WEB application"[9]
This is aimed at configuring authentication and authorization services of the web server within BES. It provided useful background information on the implementation but no information directly useful for the audit.

Searched the vulnerability database Secunia[10].
Performed subqueries by vendor = Borland, also by vendor = Visigenics (the company which developed the underlying ORB used in the AppServer), and by product = Borland Enterprise Server / Enterprise Server / AppServer. No records were found.

Searched the vulnerability database X-Force[11], keyword = Borland. Matches found: 8.
These matches were for other products developed by Borland and thus not applicable here.

Searched the vulnerability database SecurityFocus[12], vendor = Borland/Inprise. Matches found: 10.
These matches were for other products developed by Borland and thus not applicable here.

4.3.2 Resources for BES's embedded services

The search activity was refocused to find audit resources that may have been developed specifically for one of BES's embedded services. As a J2EE AppServer, BES follows the J2EE architecture and provides several standard services for its hosted applications. Some of these are implemented by products that exist in their own right – such as Apache and Tomcat. Other aspects of the architecture merely incorporate Java specifications – such as JAAS (Java's Authentication and Authorization service). Hence a search was undertaken against these components and specifications.

Many online resources were found using search terms such as {JAAS protection}, {Tomcat vulnerability threats}, {Apache vulnerability threats}, etc. The main types of resources discovered by these searches were product documentation / specifications, and online articles introducing basic concepts (i.e. tutorials). This was not the type of information that was being sought.

Several books were reviewed including the following Hacking Exposed[13], Java Server Programming[14], Java Security[15], Java Security Handbook[16]. However, none of these contained resources which would be directly applicable to an audit.

Search term: {Application Server audit checklist site:sans.org} submitted to Google. Matches found:~237.
A search of the SANS website identified several audit resources. These included a brief paper title "Web Application Checklist"[17] which provided some 26 control items to counter typical web application vulnerabilities. However, these are for the application level, which is outside the scope of this audit. Other resources found by the search were for auditing operating systems or network elements (firewalls, routers,

[8] Borland. "Borland Enterprise Server Security Service FAQ"
[9] Borland. "Securing Web Access in BES 5.2"
[10] http://www.secunia.com.
[11] http://xforce.iss.net
[12] http://www.securityfocus.com/bid/vendor
[13] Taylor. "Hacking Exposed: Java and J2EE"
[14] Allamaraju. "Professional Java Server Programming, J2EE 1.3 Edition"
[15] Oaks. "Java Security: Writing and Deploying Secure Applications"
[16] Jaworski. "Java Security Handbook"
[17] Naidu. "Web Application Checklist"

etc). These too are all outside the scope of this audit. However, the framework of these audit resources can be applied to this particular audit.

4.3.3 Resources for J2EE Application Servers in general

Since BES is a type of J2EE AppServer, and one of many, the focus was broadened to search for audit resources that may exists for other vendor's AppServers. Eventhough those resources may not be directly applicable, the concepts might be transferable.

The leading commercial J2EE AppServers are Weblogic from BEA, WebSphere from IBM, and JBoss from JBoss Group. Searches of their vendor's websites were undertaken.

Search term: {audit security threat vulnerability site:jboss.org} submitted to Google. Matches found: 0.
No materials found to review.

Search term: {audit security threat vulnerability site:bea.com} submitted to Google. Matches found: 6.
This search matched a relevant section of Weblogic's product documentation titled "Securing a Production Environment"[18]. That document describes 40 recommended security actions aimed at reducing risk in a WebLogic installation. Many of the recommendations apply to the host, network, database and application levels; all of which are outside the scope of this audit although some of the recommendations classified under Applications could be reclassified as AppServer. There were 12 actions that applied to the Weblogic Security Service.
Some of these and some of the actions under the Application heading were generic enough to be applicable to BES – such as

- not installing uncompiled JSPs and other source code on the production machine,
- not using the Servlet servlet,
- if applications contain untrusted code then use the Java Security Manager and its security policy files.

However, none of these recommendations were relevant enough to this particular system to be included in this audit.

Search term: {audit security threat vulnerability websphere site:ibm.com} submitted to Google. Matches found: 44.
No resources were found that related to securing IBM's AppServer.

4.3.4 Resources pertaining to Processes and Procedures

The research described above implicitly focused on the technical aspects of securing J2EE AppServers. However, a secure system should not rely on technical controls alone. Operational and management controls must also be used, even if only to support the technical controls. Thus, further research was undertaken to identify any resources of a process/procedure nature that might be applicable to this particular audit.

Reviewed CobiT[19]
The section of Cobit on Control Objectives provided information on a framework, principles and instances of controls useful at the operational and management levels. Cobit's numerous controls and its framework was used to guide the development of non-technical controls for this audit.

Another source of management and operational controls reviewed was the book "Security Architecture: Design, Deployment and Operations"[20]. Chapter 4 "Applying Policies to Derive the Requirements" details

[18] BEA Systems. "Securing a Production Environment"
[19] IT Governance Institute. "CobiT Control Objectives"
[20] King. "Security Architecture: Design, Deployment and Operations"

the development of such controls. This also was used to guide the development of non-technical controls for this audit.

4.3.5 Summary of search results

The search found very few directly applicable pre-existing audit resources. The technical resources were only generally applicable and most addressed elements outside the scope of this audit's focus (e.g. application or host security). Some operational resources were useful (i.e. the Cobit standard) and will be used to guide the development of the audit ruler (see Part II).

Since no pre-existing audit resources were found that could be directly used for this particular system, a development of them from first principles is undertaken. This involves analyzing the BES AppServer to understand in what ways the system's assets were being accessed and what particular vulnerabilities might be found in those areas that could threaten those assets.

5 Threat Analysis

Since no pre-existing audit resources were found (see previous section) that could be directly used with this particular system, they are developed here from first principles. This involves analyzing the BES AppServer to understand in what ways the data of the system is accessed and what vulnerabilities may exist in the system which could threaten those methods.

The number of ways that data can be accessed in this system, both directly and indirectly, is several. This analysis begins with considering direct access to the data, then proceeds to consider indirect methods, and then considers supporting activities that maintain the security of the system.

Direct access to the data of this system is through a JDBC database connection managed by the AppServer. Therefore, this analysis considers how AppServer-managed JDBC connections are protected against unauthorized use, including how the database login credentials used to establish those connections are protected.

Indirect access to the data is via the application hosted by the AppServer. Since the application has access to the data, access to the application is a way to indirectly access the data. Therefore vulnerabilities which allow unauthorized access to the application will also be considered.

Access to the application and the JDBC connections is controlled by the AppServer through its security mechanisms and configuration. Hence if these access mechanisms are circumvented or the configuration altered, then unauthorized access to the application or the JDBC connections could be obtained. Therefore vulnerabilities which allow these types of exploits will also be considered.

Finally, the threat analysis considers the domains of operational and management activities since vulnerabilities in these spheres could also threaten the assets of the system.

5.1 Access via JDBC Connections

The data of this system is located and managed by a Relational Database Management System located on a separate host outside the DMZ (on the internal network). Access to it is through JDBC connections that are managed by the AppServer and made available to applications.

The JDBC connections are established using login credentials (a username and password) assigned to the application. Since there is no other authentication mechanism used apart from the login credentials, an

unauthorized party with access to these credentials could gain access to the data. Thus protecting access to these credentials is crucial to maintaining the confidentiality and integrity of the application's data.

To identify vulnerabilities of the system's configuration that may allow unauthorized access to those credentials, the locations of where the credentials are found within the system – both at rest and in transit – are first identified. These locations are then analyzed to identify means of access. The access paths are studied to identify possible vulnerabilities. How these vulnerabilities might be exploited and the likelihood of doing so will be examined.

By convention[21], J2EE platforms store database login credentials not in the application but separately in a datasource object (defined by javax.sql.DataSource) which is registered in the J2EE Server's JNDI naming service from which J2EE components request database connections. (This convention is also followed by the application.).

For the BES J2EE AppServer, datasource objects are specified using either a DAR (Datasource Archive) file or as part of an EAR (Enterprise Archive) file[22]. Hence, these files contain database login credentials. These files are deployed to a Partition (the J2EE container) which stores them in its file system (in the ear and dar directories of <BES>/var/servers/<svr>/partitions/<partition>) to allow the information to persist between incarnations of the Partition process. Thus one location where the credentials can be found in the system at rest is in the DAR/EAR files of the Partition on disk.

When the BES Partition process is launched it reads this information from the archive file, then instantiates the specified datasource objects and registers them in the JNDI serial://datasources naming context. These serialized datasource objects are used by the application and the system to initiate JDBC connections, and hence also contain the login credentials. Thus, another location where the credentials can be found in the system at rest is in serialized datasource objects within the JNDI naming service of a running Partition.

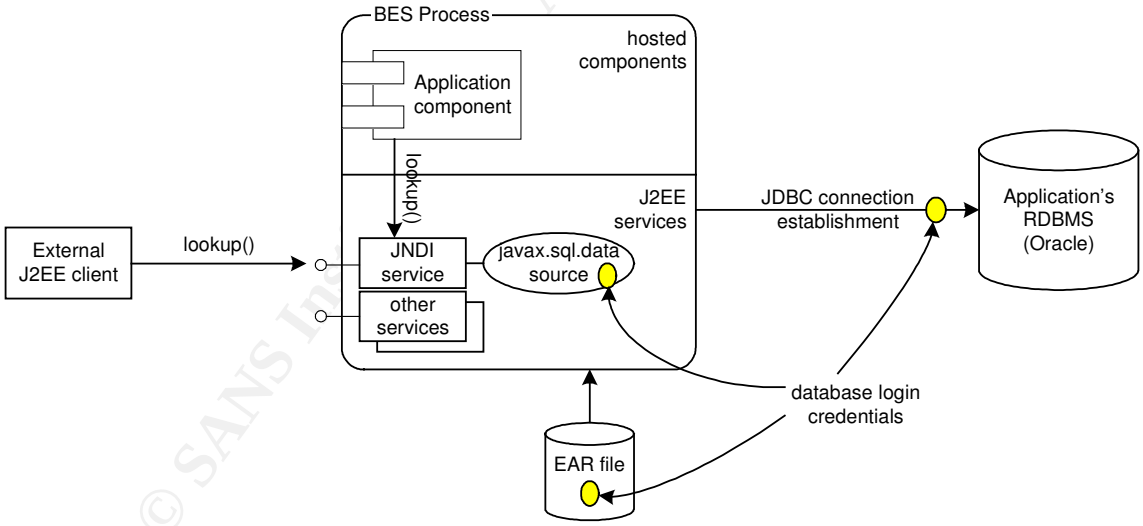

Figure 7. Locations of database login credentials

When the application needs to access data in the database it will first obtain the appropriate datasource object by performing a lookup on the JNDI naming service, then request a new connection from the

[21] Allamaraju, p200. "Professional Java Server Programming, J2EE 1.3 edition"
[22] Borland. "Borland Enterprise Server 5.2 documentation"

datasource. The Datasource may, at this point, initiate a connection to the database, or it may return an existing unused connection from its pool of connections if these have already been established. Initially, therefore, a database connection will be established and this involves providing the login credentials to the database. Thus, a third location where the credentials can be found in the system is in transit between the Partition process and the DBMS system on another host.

These three locations of the login credentials in the system are depicted in Figure 7.

Having identified where the login credentials are found within the system (both at rest and in transit), these sites are now analyzed to identify possible means of access and the likelihood of them being exploited.

To access the credentials stored within a DAR or EAR file requires read access to the file. If this is obtained then the contents of the file can be extracted and must then be interpreted. Interpreting the contents is not difficult since the EAR and DAR files follow a simple file format mandated by J2EE[23] (the Java archive format). Within the archive, the credentials appear as properties of a datasource that is specified in an XML file – named jndi-definitions.xml. This file's schema is also well defined; indeed, being an XML file it is also human readable. So extracting the login credentials from the jndi-definitions.xml file is easily accomplished. Thus extracting its database login credentials is possible if read access to a DAR or EAR file is obtained.

To access the credentials configured within the serialized datasource object involves retrieving a copy of the object from the JNDI naming service. Retrieval is achieved by invoking the lookup() method[24]. A lookup() call can originate either from within the J2EE container (from a deployed component or class) or from an external client via CORBA. If it is internal then this requires first deploying a component to the Partition which typically requires BES administrator privileges. Irrespective the source of the call, the lookup() method returns a serialized datasource object which can be queried for login credentials using the getUser() and getPassword() operations. Thus, if network access to the Partition's JNDI name service is obtained (by which an external client can invoke lookup()), or if an unauthorized J2EE component is loaded into the J2EE container, extracting these login credentials is possible.

A related but distinct vulnerability is that the datasource could be retrieved and used to establish a connection with the datasource even without extracting the login credentials. Similar to the previous vulnerability, this requires access to the JNDI service of BES, but unlike the previous vulnerability, this does not require extracting the database credentials in order to establish a connection. Note that the JDBC connection identifies the database as an endpoint on the network, and therefore could be used from hosts other than BES's.

To access the credentials while in transit between BES and the DBMS requires monitoring network traffic and that the traffic not be encrypted. The TCP network configuration between these two hosts means that such monitoring could be done using a Man-in-the-middle attack, such as can be achieved using dsniff[25]. This type of exploit is moderately difficult to execute without causing disruption to the network and thus announcing its presence. Alternatively, the network stacks of either machine could be monitored using tools such as ngrep[26]. However, these require utilities to be installed which can likewise affect normal operations particularly with respect to performance and reliability. If such a tap can be made to the communication stream it could then be examined for login credentials which are sent during JDBC connection establishment. To be able to identify the credentials in the stream and use them requires that it not be encrypted. The Oracle JDBC Type-4 driver used in this application can support encrypted communications but by default this is not enabled. Accessing credentials using this approach is much more complex than the other two approaches mentioned above.

[23] Allamaraju, p35. "Professional Java Server Programming, J2EE 1.3 edition"
[24] Sun. "Looking up an Object"
[25] Russel. "Penetration Testing with dsniff"
[26] Eyler. "Using ngrep"

5.1.1 Threat Summary

Three possible access paths to database login credentials in the system were discussed above. These are summarized in the following table.

Threat	Vulnerability	Exploit	Exposure	Likelihood
T1.1	Credentials are stored in datasource definition within DAR/EAR files	Read the datasource definition section of the DAR/EAR file. Extract credentials from datasource. Establish own database connection using credentials.	Requires file access to the DAR/EAR file.	High
T1.2	Credentials are configured in the Datasource loaded in JNDI naming service	Retrieve datasource object from JNDI naming service as a hosted J2EE component. Extract credentials from datasource. Establish own database connection using credentials.	Requires annexation of an already deployed component, or deployment of an unauthorized component.	Med
T1.3		Retrieve datasource object from JNDI naming svc as an external client to Partition process. Extract credentials from datasource. Establish own database connection using credentials.	Requires network access to Partition's JNDI name service.	High
T1.4	Connections to Database are available from Datasource object registered in JNDI naming service.	Access JNDI naming service from within Partition (i.e. as a hosted J2EE component) and request new connection (from pool) using Datasource.	Requires annexation of an already deployed component, or deployment of an authorized component.	Med
T1.5		Access JNDI naming service from outside Partition process and request new connection (from pool) using Datasource.	Requires network connectivity to Partition's NamingSvc socket.	Med
T1.6	Credentials are sent across network during connection establishment	Monitor connection traffic to observe credentials in transit. Establish own database connection using credentials.	Requires access to decipherable network traffic (a relative term, depending upon agent's resources).	Low

Table 1. Summary of Threat T-1

Each of the exploits identified requires the threat agent to have already gained access either to the AppServer host or network. That is, these vulnerabilities are accessible only to threat agents who have compromised the underlying host and network protection domains or to threat agents who may exist already within these domains (viz. the internal network or DMZ as the case maybe). The likelihood of this condition is not considered since it depends on the security of those layers and they are beyond the scope of this audit. Nevertheless, this likelihood is the same for each individual exploit in T1.

However, factors which vary between individual exploits in the set does cause the relative likelihood of them to vary with respect to each other. Such factors include the complexity of an exploit, the minimum time required to perpetrate an exploit, what auxiliary utilities are required in the environment to support the attack, etc. These variances were considered during the threat analysis above and are the basis for the relative differences in likelihood between each exploit recorded in Table 1.

The impact to the organization's business operations of obtaining database login credentials is considered to be significant. Unauthorized access to this application's data would result in its confidentiality and

integrity being compromised. This has the potential to cause great impact to the business's reputation and regulatory standing to the extent that it could threaten their ability to continue in business.

5.2 Access via the Application

The previous section analyzed how direct access to the database could be obtained. Yet even if all those access methods where protected, the system's data would still be accessible via the application. Therefore, in this section, access to application is considered because it is a path for accessing the system's data. Note, because the scope of this audit does not extend to the application, vulnerabilities of that level (such as SQL injection[27]) are not considered.

Access to the application's functionality is protected by access controls declared by the application and implemented by the J2EE AppServer. This approach is specified by the J2EE architecture[28].

For example, a web-based J2EE application can declare that certain web resources (e.g. JSPs or servlets) are to be protected by access controls and that the authentication control to be used is HTTP Form-based authentication. In this case the J2EE application supplies the required login and error JSP pages, but it is the J2EE AppServer which performs the necessary HTTP actions of redirecting callers to the login.jsp if they are not yet authenticated, as well as authenticating callers against the desired authentication mechanism using the credentials they supplied.

Hence there are two separate and distinct parts to access control in J2EE applications: its specification by the application and its realization by the AppServer. The specification by the application is concerned with issues such as: are the authorization privileges sufficient for the application requirements? are all of the confidential resources correctly identified as to be protected by access controls? etc. These issues can not be answered by the AppServer administrator but only by the application developer. Therefore, such application-level issues are beyond the scope of this audit and not addressed here. The other part of access control is the realization of the application-requested controls by the J2EE AppServer, which concerns issues such as: are the mechanisms it offers to applications correctly implemented? are its operations being performed in ways which preserve the confidentiality and integrity assumed by the application? etc. These issues are within the scope of this audit and are addressed here.

The application being audited uses HTTP Form-based authentication to solicit user credentials, and LDAP to authenticate and authorize them. If these credentials were to be obtained by unauthorized parties, then access to the functionality of the application as a valid user could be achieved and thus access to the system's data.

To identify vulnerabilities of the system's configuration that may allow unauthorized access to those credentials, the locations of where the credentials are founds within the system – both at rest and in transit - are first identified. These locations are then analyzed to identify methods of access. Potential access methods identify vulnerabilities of the system's configuration. How these vulnerabilities might be exploited and the difficulty of doing so is also examined.

When a user attempts to access the application, they are presented with an HTTP Login form to which they enter a username and password. At this point, the credentials exist on the user's platform and could potentially be captured by a keylogger or by physically observing the keys typed (shoulder surfing). The security of the user's platform is beyond the scope of this audit and so is not analyzed for such vulnerabilities.

[27] McDonald. "SQL Injection: Modes of attack, defense, and why it matters"
[28] Allamaraju, p881-888. "Professional Java Server Programming, J2EE 1.3 edition"

Once the user enters the credentials the user's browser submits them to the URL identified in the login form. Thus, another place where user credentials can be found is in transit between the browser and the BES host.

Once BES receives the credentials, they are routed to the LDAP server for authentication as per the access control configuration of the application. Thus, another place where user credentials can be found is in transit between BES and LDAP on a different host.

If the supplied credentials were invalid, then the user's browser is redirected back to the same login page to try again. Although there are no valid credentials revealed in this instance, it does give the user another chance to authenticate.

The sources, sinks and paths for the login credentials to the application are depicted below in Figure 8.

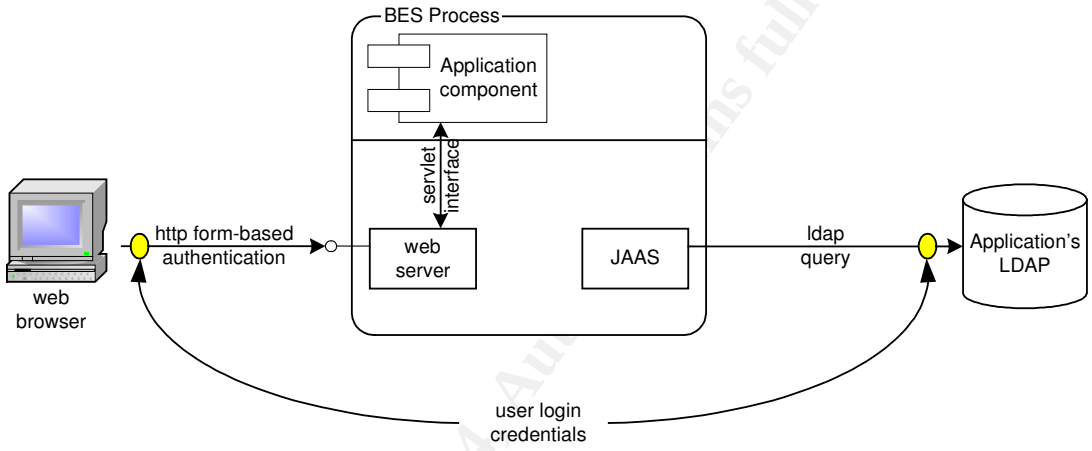

Figure 8. Locations of application login credentials

Having identified the locations within the system where application login credentials can be found (both at rest and in transit), these are now analyzed to identify possible methods of access and the likelihood of them being exploited.

Intercepting the credentials sent by the browser to the web server can be achieved by monitoring the network communication and analyzing the traffic. This can be done either at the network level or higher if an HTTP proxy is present. Locations where the communication stream could be tapped include the user's network endpoint, the web server's endpoint, or the path between them. Although each of these locations is outside the scope of this audit, the encoding used by the AppServer for the communication stream is within scope since it is under the control of the AppServer's configuration. Therefore, if the communication stream is viewable by a 3rd party, then this has exposed a vulnerability of the AppServer: that credentials exchanged with it are not kept confidential from external parties. Therefore, if access to the stream has been obtained and if the stream is not encrypted, extracting the credentials from it is not difficult.

The user credentials also appear in transit from the AppServer to the LDAP server during the authentication process. Here also the credentials are exposed to interception. However, in this case the interception points are entirely internal to the organization. As in the previous exploit, the confidentiality and integrity of the stream is the responsibility of the AppServer even though it travels through elements of the system that are beyond the scope of this audit. Therefore, if this stream is viewable by a 3rd party, then it has exposed a vulnerability of the AppServer configuration. The communication with the LDAP server uses the LDAP protocol, which allows the identification of these credentials in the stream. In fact, some tools

such as dsniff[29] have native support for the LDAP protocol. Therefore, if access to this stream has been obtained and if the stream is not encrypted, extracting the credentials from it is not difficult.

Another means for obtaining access to the credentials is by searching the application's authentication domain space. Since the application uses a username/password scheme, a search would involve submitting username/password combinations in an attempt to determine a valid combination. The username space is typically based on the names of users of the system, and as such it is neither random nor evenly distributed. Usernames can easily be inferred from staff names of the organization, which can be obtained by various means. Likewise, the password space is theoretically large, but in practice usually contains values that are neither evenly distributed nor particularly random. Passwords devised by humans are notoriously non-random[30]. Furthermore many applications don't require users to change their passwords. All of these factors allow assumptions to be made that reduce the size of the space that must be searched and thus increase the effectiveness of the attack. Furthermore tools to assist with this search activity are available[31] or can be easily developed. Thus if access to the authentication process of the application is gained by a threat agent, then methods exist for exploiting this vulnerability.

Finally, valid user credentials for the application maybe derived from well-known user accounts if those have not been removed from the system. Such identities are installed by default during the product installation of BES. If these identities and domains are not removed then these user accounts could be used to successfully authenticate against the system. One of the authentication domains defined by default in BES uses the host's authentication service. Consequently any account on the BES host or recognized by that host could be used to successfully authenticate with the system. The general knowledge of these accounts means that this vulnerability is easily exploitable. However, unless these roles are mapped to valid values in the application, the user would not be able to access secured resources. For this reason, the likelihood of this exploit threatening the application is not as high as it would otherwise be.

5.2.1 Threat Summary

Four possible methods for obtaining application login credentials were identified. These are summarized in Table 2.

Threat	Vulnerability	Exploit	Exposure	Likelihood
T2.1	User credentials are exchanged between browser and BES	Monitor user's network to observe credentials in transit.	Requires access to user's remote network.	Med
		Monitor intermediate network paths and proxies to observe credentials in transit.	Requires access to intermediate network and proxies	Low
		Monitor web server's network to observe credentials in transit.	Requires access to web server host and/or network	Low
T2.2	User credentials are exchanged between BES and LDAP	Monitor connection traffic to observe credentials in transit.	Requires access to network traffic and traffic be decipherable (a relative term, depending upon agent's resources).	Low

[29] As of version 1.7, dsniff has the ability to parse the LDAP protocol.
[30] McAuliffe. "Computer Passwords Reveal Worker's Secrets"
[31] Brutus

T2.3	Application uses username/password authentication scheme	Search the authentication space for valid combinations (username / password pairs) by repeatedly attempting to login to the application either with enumerated lists (brute-force) or dictionary lists.	Requires access to the application login page via the Internet	High
T2.4	Default user accounts are installed during installation of BES.	Use these well-known login credentials to authenticate against application.	Requires access to the application login page via the Internet	Med

Table 2. Summary of Threat T-2

It should be noted that some of these exploits (T2.1c and T2.2) require a degree of access to either the host or network which would not ordinarily be expected in a production environment. That is, these vulnerabilities were exposed only to threat agents who had compromised underlying host and network protection domains or to threat agents who may exist already within those domains. It is beyond the scope of this audit to examine the potential for that.

However, the remaining threats were based on vulnerabilities that could be exploited by threat agents from the external network. Furthermore, since the complexity of these exploits were less than the others, it would be reasonable to assume that internal threat agents would likely prefer these exploits.

Therefore the overall likelihood of this general threat (T2) relative to other threats identified in this analysis is considered high because these exploits are relatively easily to execute (existing tools are available) and primarily because the threat agent population is larger due to the vulnerabilities being accessible from the Internet.

5.3 Access via Compromising the AppServer's Configuration

The previous two sections identified access to the application's data through the application indirectly and through JDBC connections directly. Those access methods in turn could be protected by security mechanisms of the AppServer. Examples of these mechanisms include the JAAS framework, JAAS LoginModules and their configurations, SSL cipher suites and their configurations, etc.

Yet if these controls are not themselves protected against unauthorized reconfiguration, then they could be altered to weaken or remove the protection they provide. This would allow unauthorized access to the application and/or its data. Hence, the protection of these configurations is important.

Since the configuration of these mechanisms is both particular to the application and part of the J2EE AppServer, it is considered within the scope of this audit. However, the soundness of the mechanisms themselves (e.g. the correctness of a vendor supplied JAAS LoginModule implementation or of a vendor supplied SSL cipher suite) is beyond the audit scope because it is not particular to the application or even this organization. Auditing those aspects could be done independently of this application.

To identify vulnerabilities and exploits that might allow unauthorized reconfiguration to occur, the methods for accessing the system's configuration are first identified. These methods are then analyzed to identify potential vulnerabilities and exploits. Factors affecting the likelihood of such threats are also discussed.

5.3.1 AppServer security configuration

The configuration of BES's security mechanisms is defined by various properties that are stored in files within its installation directory. These files are read by BES processes at start-up.

Thus, direct access of these files is one method of accessing the configuration.

Another method of accessing the configuration is via the Partition process. The Partition is the J2EE container functionality of the BES architecture and hosts J2EE application components. The Partition process must access these configuration files to obtain its runtime configuration when starting. Therefore as a minimum, it has read access to the files. Being a container for application components also means that application defined functionality can execute with this process. Depending upon the configuration of the Partition (as will be discussed) the hosted application components may also have access to the configuration files from within the Partition.

Another process of BES that accesses these configuration files is IAS. This process acts as a local proxy to carry out BES administration activities including starting and stopping other BES processes. It also can update the configuration when changes to it are requested from a remote BES Console. For example, when an administrator uses the BES Console to change a BES server property, that change is communicated to IAS that makes the change to the local file. In this way, IAS implements the local presence of remote administration commands.

These three identified methods of accessing the system's configuration are shown in Figure 9.

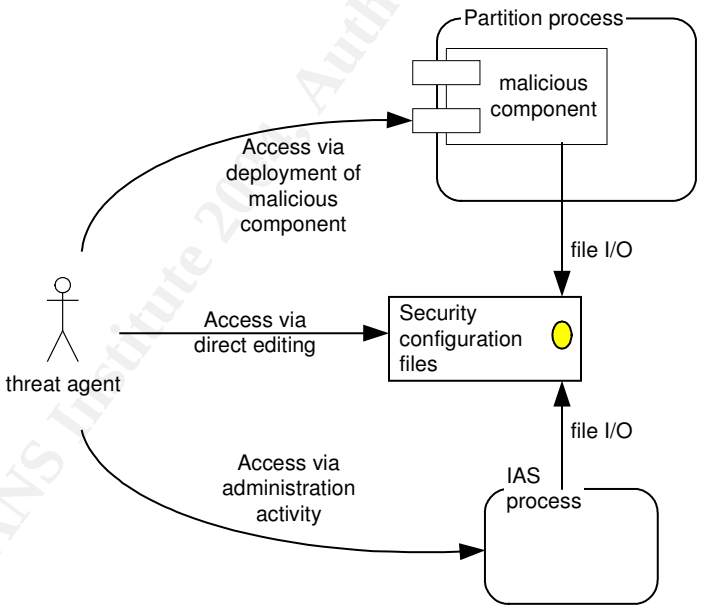

Figure 9. Access paths to BES Security Configuration Files

Having identified the methods by which the system's configuration can be accessed, these are now analyzed to identify possible exploits and factors that affect the likelihood of them being exploited.

Being able to modify the system configuration directly by editing the configuration files requires write access to the files by the threat agent. This is controlled by the operating system privileges in effect for those files. If the permissions include users and groups who are not necessary, then this creates a vulnerability in the configuration. If a threat agent gained access to a configuration file, changes could be

easily made since the file follows a human readable format. Therefore, if this vulnerability is exposed, its exploit is very simple.

Restricting permissions on the configuration files to allow only access by the BES processes would not however eliminate the potential for unauthorized access. Unauthorized access could still occur indirectly through either the IAS process or the Partition process, which would still have the necessary operating system privileges.

In the case of the partition process the potential exploit is that an application component hosted within the partition will attempt to access the files through Java IO operations. If this attempt is not prevented by the Partition, then the operations will access the file with the operating system privileges of the process (i.e. the Partition). The Partition can control the resources accessible by hosted components through the Java security policy of the Partition. The Partition executes a Java Virtual Machine that hosts the application components, and therefore the JVM can be used to restrict resource access to particular Java classes only. However, this requires that the Partition's Java security policy be set correctly. This exploit also requires that a malicious component be deployed to the Partition, which is an activity that may be prevented by other controls. Therefore this exploit is only moderately complex but does require deployment capability to the Partition as well as an inadequate Java security policy on the Partition.

Similarly, if the IAS process were directed by a threat agent to change a configuration file then the resulting file operation would be performed by the IAS process. To guard against this, the IAS process requires callers to first authenticate against the BES Administration realm.

By default, the installed BES administration realm is configured to be a small set of well-known user accounts and passwords (the ServerRealm and userdb.jds authentication database). If these default settings are not changed after installation then one method for gaining unauthorized access to IAS is to use these well-known accounts. Since the usernames and passwords of these default accounts is generally known this would be a vulnerability. Also, since IAS can be contacted via the network this increases the locations from where this attack could be launched. This by itself would increase the likelihood of it being exploited.

Other means of gaining valid authentication credentials to IAS include searching its authentication space for valid combinations (password enumeration). Also, since this AppServer's administration domain is configured to authenticate against the internal LDAP server, a password search could be launched against that server directly. There is also the possibility of monitoring IAS's communication with its authentication provider (LDAP in this case) to observe valid authentication credentials in transit. The later would require some amount of development (programming) since the UI provided by the BES Console is not HTML-based hence existing web-based password cracking utilities can not be used as is. However, the amount of effort required is only modest. Since this exploit could be performed from anywhere on the network which has access to the IAS process which is located on the BES host, the likelihood of this exploit is moderate because of its relative ease and the potential threat agent population.

5.3.2 Threat Summary

Five possible methods for gaining access to the AppServer's configuration were identified. These are summarized in Table 3.

Each of the exploits identified requires the threat agent to have already gained access either to the AppServer host or network. That is, these vulnerabilities are accessible only to threat agents who have compromised the underlying host and network protection domains or to threat agents who may exist already within these domains (viz. the internal network or DMZ as the case maybe). The likelihood of this situation was not considered since it depends on the security of those layers, which are beyond the scope of this audit. However its effect on each individual exploit in T3 is the same.

Threat	Vulnerability	Exploit	Exposure	Likelihood
T3.1	Security configuration for AppServer is stored in operating system files	Directly edit the contents of files to alter security configuration	Requires write privileges (in OS domain) to these files	Low
T3.2	J2EE components hosted within Partition have Partition's OS privileges when performing file operations	Malicious component is deployed to Partition and accesses security configuration files from within the process (under the processes' privileges).	Requires Partition process be run with write permissions for these configuration files. And requires malicious component be loaded in Partition.	Med
T3.3	Sample user accounts are installed with BES.	Authenticate to BES admin realm using default BES accounts and change security config via Console.	Requires default authentication realm to be present.	High
T3.4	Authentication to BES uses username/password scheme	Search the authentication space for valid combinations (username / password pairs) by repeatedly attempting to login to BES either with enumerated lists (brute-force) or dictionary lists.	Requires a utility to automate the process.	Med
T3.5	Administrator credentials are exchanged between BES and LDAP	Monitor connection traffic to observe credentials in transit. Then use these credentials to gain access to IAS and instruct it to modify the security configuration files.	Requires access to network traffic and traffic be decipherable (a relative term, depending upon agent's resources).	Low

Table 3. Summary of Threat T-3

The relative likelihood of exploits within the set does vary however because of by factors that differ between exploits. Such factors include the complexity of an exploit, the minimum time required to perpetrate an exploit, what auxiliary utilities are required in the environment to support the attack, etc. These variances were considered during the threat analysis above and are the basis for the relative differences in likelihood between each exploit recorded in Table 3.

The effect of unauthorized access to the configuration could be to weaken or remove security controls, which would allow unauthorized access to the application and its data. This in turn would have a significant impact to business operations.

5.4 Non-technical Threats

In addition to the technical threats identified above are threats of a non-technical nature. Non-technical threats arise from vulnerabilities that exist in the practices and procedures of the organization as it interacts with and supports the system. Although such threats may not be malicious, they can nevertheless have significant impact upon the system's security.

For example, having never attempted to restore the system from a backup (an operational concern), or not having commitment from senior management for sufficient personnel to monitor the security of the system (a management concern) can threaten the assets of the system as technical vulnerabilities do.

Following the same formula used above, vulnerabilities in the system that might expose the assets to significant risk are examined. The exploits identified are events which would cause the vulnerability to be realized but are often not intentional acts as is usually the case with technical vulnerabilities.

5.5 Operational Risks

Many of the technical threats above can result in the system's configuration and/or its data being modified. In the extreme case, the modifications may corrupt the configuration or data so extensively as to destroy them. Two possible responses to this event are to either restore the system from a backup or in the case of the product, to reinstall it and apply baselined configuration changes. If the organization does not have these capabilities then the system is exposed to the system potentially losing data and/or being offline for a significant period of time. It is therefore important the organization have the operational capability to perform these tasks. This concern is raised as threat T4.1.

Being able to restore the system's configuration presupposes that the version that was in use is known and is the version that will be restored. If it is not known, then the restoration process may lead to a different system configuration than what was in production. This could introduce unexpected behavior to the application and even undo changes that (re)expose previously protected security vulnerabilities. All of which have impact upon the system and can raise the amount of risk facing it. Hence it is important that changes to the system's configuration are tracked and reflected in the restoration copies (baselines or backups) as well. This requires that such operational activities follow a predefined process that can be audited for correctness and sufficiency. (See Cobit DS9). This concern is raised as threat T4.2.

As new vulnerabilities are discovered in systems (not just the ones of this organization), patches and updates are developed and released by the vendor. If these updates address newly discovered vulnerabilities then they need to be applied to the system to protect against the newly discovered vulnerability from being exploited. But there are also patches that may have little or no relevance to the security of the system. Since updating a system involves significant administration effort, non-critical updates should be queued behind more critical work. Also, being informed promptly of patch availability, evaluating its impact to the system, and rolling it out into production according to a schedule are important operational activities that support the security of the system and should be performed regularly and under the guidance of a policy to ensure consistency. This goal is closely aligned with the Cobit DS13 (Manage Operations) requirement. In the absence of such operational maturity, the secure state of the system can decay and/or the operations effort spent supporting it maybe not be effective. This concern is raised as threat T4.3.

As it is the organization's intention to continue using this system well into the future, and given the number and diversity of vulnerabilities identified so far, the likelihood of a security incident occurring (i.e. a vulnerability being exploited) is high. In the absence of an incident response process to follow, these events may go undetected or if detected not dealt with effectively when they arise. This could have a direct and immediate impact on the system, as well as causing the system to remain vulnerable in the future if the incident's root cause is not identified. Having the ability to manage problems and incidents relating to security in a formalized way is the goal of Cobit's DS10 requirement (Manage Problems and Incidents). This concern is raised as threat T4.4.

Some aspects of the system inherently rely on physical controls for protection. For example, backups of the data and configuration of the system are stored on portable media. If physical access to that media is obtained then the data could be taken or copies made of it and the copies taken. This would breach the data's confidentiality. Therefore, it is important to manage the security of the facilities in so far as they are

involved in the security of the system. This is the goal of Cobit DS12 (Manage Facilities). This concern is raised as threat T4.5.

These threats to the security of the system of an operational nature are summarized in Table 4.

Threat	Vulnerability	Exploit	Exposure	Likeli hood
T4.1	System can not be restored from baseline	System configuration or data is damaged beyond repair.	Applies to all elements of system configuration and all data of the application	Med
T4.2	Changes to system's configuration are not tracked	Changes are made to configuration without recording what was changed, by whom, when and for what purpose.	Limited to individuals who have access (authorized or not) to the BES host	Med
T4.3	Regular maintenance activities not scheduled	Newly discovered vulnerability is exploited for which a patch existed. Or, system needs to be restored from backup.	Applies to all facets of system	Med
T4.4	System is not monitored for security incidents	Any incident involving the system.	Applies to all facets of system	High
T4.5	Physical access to system's backup media	Backup media is taken (or switched) and copied for offline analysis.	Requires physical access to backup media.	Med

Table 4. Summary of Threat T-4

5.6 Management Risks

Performing the operational activities of securing and maintaining the security of the system requires software and hardware resources, and properly trained personnel. Since it is the organization's management which budgets for and allocates these, it is important that management be informed of these needs. These issues which can threaten the security of the system are categorized under a third domain – management (c.f. technical and operational domains).

Ideally security of the system is the responsibility of a particular position within management. Ideally that position is also given the authority to effect changes as required. The absence of such a position can have an adverse impact on the ongoing security of the system. This concern is raised as threat T5.1

In addition, personnel are often involved in many of the operational procedures mentioned in the previous section. And eventhough those procedures maybe documented, they often assume the reader (the operator) has a certain minimum understanding of the system already as well as the skills needed to execute the actions. Often, when the administrator writes the procedure they assume a certain level of understanding and skill. If another person needed to perform the procedure does not posses this knowledge or skill then they may not be able to follow the procedure. This vulnerability could impact the system if an operational procedure (e.g. restoring from backup or handing an incident) became necessary but the primary administrator was not available to perform it. Reasons for the unavailability of the administrator range from unavoidable accidents (the proverbial "hit by a bus"), to natural staff turnover (according to the

SAGE/SANS/BigAdmin Annual Salary Survey of 2002[32] the mean length of stay for an administrator with an employer is 4.32 years). This concern is raised as threat T5.2

These threats to the security of the system that can be traced to management issues are summarized in Table 5.

Threat	Vulnerability	Exploit	Exposure	Likeli hood
T5.1	Responsibility of system's security is not assigned to an individual within the organization	Changes to the system that require management approval will not be enacted. Or, security needs of the system change which require re-evaluation by management by will not get approval for this activity.	Applicable to all facets of the system	Low
T5.2	Operations can only be restored by one individual in the organization	System needs to be restored but the individual is not unavailable.	Applicable to all facets of the system	High

Table 5. Summary of Threat T-5

5.7 Summary of Threat Analysis

This section undertook a threat analysis of the system based on first principles. That is, it considered the assets of the system: the confidentiality and integrity of the application' data, and then proceeded to discover vulnerabilities which if exploited could have an adverse impact on them.

This analysis began with a narrow focus on how the database login credentials might be obtained, which lead to a category of threats (T1). The focus then progressively broadened to include how other elements of the system could indirectly access the data (threat category T2), then to how those protection measures are themselves protected by the administration functions of the system (threat category T3). Finally, non-technical matters of the system were examined; specifically operational and management issues (vulnerabilities) which could also impact the security of the system (threat categories T4 and T5 respectively).

[32] Kolstad, p25. "SAGE/SANS/BigAdmin 2002 Annual Salary Survey"

6 Summary of Part I

Part I of this audit focused on defining the subject, scope and purpose of the audit.

After identifying the system to be audited, an assessment of the risks facing it was undertaken. To help in this regard a risk assessment terminology was introduced. The risk analysis approach used was asset-driven whereby the business value of the system's assets directed the focus of the audit to consider threats which would cause impact to the system's assets and thus the organization's business.

After identifying the assets of the system (the data of the system and more specifically its confidentiality and integrity), an extensive search was performed to locate audit resources which might assist with auditing the system in this respect. Although very little directly applicable information was found, several resources contributed information of a general-purpose nature to this task.

To supplement this information and provide a basis for the remainder of the audit, a threat analysis of the system was undertaken. The analysis considered how the system's assets were accessed and revealed several vulnerabilities. These vulnerabilities combined with possible exploits formed threats, which if realized would place the system at risk.

Part II

The Audit Ruler

Table of Contents

Overview of Part II

Part II documents security controls that were developed in response to the threats identified in Part I.

Each control is classified as one the following types[33]:

- Preventative – deter and avoid undesirable events from taking place
- Detective – identify undesirable events that have occurred
- Corrective – correct undesirable events that have occurred
- Deterrent – discourage security violations
- Recovery – restore resources and capabilities
- Compensating – provide alternatives to other controls

These are listed in order of preference: preventative controls being most preferred. This attribute is noted for each control.

The goal of a control is to counter a threat. Both the goal and the threat referenced are noted for each control.

Each control also defines compliance criteria that systems can be measuring against. If the system is not compliant with the control, then this implies that the vulnerability associated with the corresponding threat is exposed by the system and thus could be exploited.

Each control also includes a description of how to measure the system's compliance (i.e. perform a compliance test). Some of these tests are stimulus/response tests, which are preferred over inspection-type tests. The objectiveness of each test is also noted. Objective tests are independently verifiable and repeatable and are thus preferred. However, some situations do not allow for objective tests and these are noted when they occur.

[33] Harris, p159. "CISSP Certification Exam Guide"

1 Technical Controls

Addressing Threat T1

A summary of the controls addressing threat T1 - the threat of gaining access to the database by obtaining database credentials assigned to BES.

Ref	Control Objective	Threat Ref.	Compliance Measure	Test for Compliance
C1.	Restrict read access to DAR and EAR files	T1.1	DAR and EAR file permissions will deny the 'other' group write access.	Use operating system calls (find) to determine permissions of all ear/dar files in the BES file system. Test is stimulus/response. Test is objective.
C2.	Restrict access to JNDI service	T1.2	External agents can not retrieve datasource objects from the JNDI naming service	Use an external program that connects to JNDI and retrieves a datasource. Test is stimulus/response. Test is objective.
C3.	Keep DB credentials confidential while in transit	T1.4	Credentials in transit are not decipherable to parties other than BES (Partition) and the Database.	Use known tokens as login credentials and observe the communication stream between BES and Database during connection establishment for presence of tokens. Test is stimulus/response. Test is objective.
C4.	Restrict connections to DB to BES only	T1.1, T1.3, T1.4	Requests to establish a new connection with the database (originating other than from BES) are refused	Attempt to establish a new database connection other than from BES. Test is stimulus/response. Test is objective.
C5.	Restrict permissions granted to BES JDBC Connection	T1.1, T1.3, T1.4	Database permissions granted to application do not include schema manipulation or DB administration functions.	Login to Database using credentials supplied to the application and attempt a schema modifying command (such as creating a table). Test is stimulus/response. Test is objective.

Table 6. Summary of Controls addressing Threat T-1

Control 1

Source: original work

Control Objective: Restrict file access to deployed DAR/EAR files to the BES process group only.

Risk Addressed: Addresses threat identified by T1.1: database login credentials assigned to the application and managed by the BES process are embedded within deployed DAR and EAR files. Having read access to these files allows the credentials to be extracted, which can subsequently be used to gain access to the database hosting the application's data (as the user assigned to BES). This leads to the confidentiality of the application's data being compromised.

Compliance Criteria: File permissions on all .dar and .ear files do not allow read access by the 'other' group. This compliance metric is binary.

Type of Control: Preventative

A Test for Compliance: Search for all files named *.dar or *.ear which are readable by the 'other' group in the BES server directory (<BES-server>). This can be achieved with the command:

```
find <BES-server-dir> -type f -name '*.[de]ar' -perm +4 -ls
```

This lists files in the <BES-server-dir> directory that are readable by 'other' (perm=4). If no files are matched (listed) by the command, then the system is compliant.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: The above test is not a stimulus / response test, but is easily scriptable for automatic testing. However, the following manual procedure is a stimulus / response test. The stimulus is to attempt to read a deployed DAR or EAR file. Specifically, to locate the <prop-name> tag that declares a password. The response (which indicates non-compliance) is to be able to locate such an XML tag in the jndi-definitions of that archive. The procedure to execute the stimulus/response test is:

1. Login as a user who is not a member of the BES process group.
2. Extract the jndi-definitions.xml file from the DAR or EAR archive.
3. Search for the XML tag "<prop-name>password</prop-name>" within the jndi-definitions.xml file.
4. If this tag can be located, then it demonstrates that the value of the password (which is identified by the tag - <prop-value>) can likewise be extracted from the DAR or EAR archive. In which case, the system is not compliant.

Control 2

Source: original work

Control Objective: Restrict access to JNDI service.

Risk Addressed: This control addresses the threat identified by T1.2: database login credentials assigned to the application and managed by BES are saved as properties of DataSource objects that are serialized in JNDI. Retrieving such datasource objects from JNDI and querying these properties will return the login credentials, which would allow unauthorized access to the database. This leads the confidentiality of the application's data being compromised.

Compliance Criteria: Unauthorized external agents can not retrieve datasource objects from BES's JNDI serial naming context. This compliance metric is binary.

Type of Control: Preventative.

A Test for Compliance: Run an external program which connects to BES's JNDI naming service and retrieves Datasources configured in the serial:// naming context. Then, query these retrieved datasources for username and password properties. The source code for such a program is provided in Appendix A.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to request a Datasource from the BES JNDI naming service. The response (which indicates non-compliance) is to receive the requested datasource from JNDI.

Control 3

Source: original work

Control Objective: Keep database login credentials confidential while in transit.

Risk Addressed: Addresses threat identified by T1.4: database login credentials assigned to the application and managed by BES are sent to the database process during connection establishment in order to identify and authenticate the caller – BES. If the communication between BES and the database is not confidential, then these credentials could be obtained by a third-party and used to gain unauthorized access to the database, thus accessing the application's data (as the user assigned to BES). This would lead to the application's data confidentiality being compromised.

Compliance Criteria: Database login credentials sent from BES to Database during connection establishment are not decipherable by a 3rd party. This compliance metric is binary but depends upon the definition of decipherable.

Type of Control: Preventative.

A Test for Compliance: The goal of this test is to monitor communications between BES and the Database, to see whether a known token (the password) can be observed in transit. If such a token is observed, then it demonstrates that the credentials are observable by a 3rd party. (Note. This compliance measure depends upon the capabilities of the observing 3rd party – specifically, their ability to decipher the stream. In this test that capability is interpreted weakly; meaning that the 3rd party is only able to decipher unencrypted streams. In reality, the 3rd party maybe able to decipher weakly encrypted streams as well.)

Compliance can be tested for as follows.

1. Define a new datasource object whose URL property identifies the database of interest but whose login credentials are invalid (yet unique) and package this into a DAR file. (The DAR file developed for Control-2 Test can be reused for this test by altering its URL, username and password properties as required.). Developing a DAR file can be done easily with the BES Deployment Descriptor Editor[34].
2. Deploy the DAR file (the datasource) to the BES Partition. Deployment is discussed in BES documentation[35].
3. Monitor the communication stream between BES and Database. Monitoring the communication stream can be done via Man-in-the-middle type tools such as dsniff[36] or via tools that monitor the network stack of a host (either the BES or Database host) such as ngrep[37].
4. Retrieve the datasource from JNDI and use it to open a new JDBC connection. This is a common task for a J2EE-component (see J2EE programming[38] for details). This action will result in the datasource opening a TCP connection with the database and then sending the login credentials for access control purposes.

---

[34] BES User's Guide. Chapter "Using the Deployment Descriptor Editor"
[35] BES User's Guide. Chapter "iastool command-line utility"
[36] Song. "dsniff"
[37] Eyler. "ngrep"
[38] Sun. "Looking up an Object"

5. Search the data of the communication stream to see whether the unique token (the password) was sent in plain-text to the database.
6. If the unique token (the password) was observed in the communication stream, then the test demonstrates that the login credentials were sent in plain-text.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to cause BES to initiate a database connection (i.e. send known and unique credentials to the database). The response (which indicates non-compliance) is to observe those known and unique credentials in transit to the database.

Control 4

Source: original work

Control Objective: Restrict database connection origins to BES process only.

Risk Addressed: Addresses threats identified by T1.1/3/4. In those threats, login credentials are obtained by various means and used to establish connections with the database other than from BES (only T1.2 used connections established by BES). Since normal operations do not require these connections to originate other than from BES, this control can be put in place to provide an extra layer of defense, preventing database connections from being established even when the credentials may have been disclosed.

Compliance Criteria: Attempts to create database connections to the database from other than the BES process are denied. This compliance metric is binary.

Type of Control: Preventative.

A Test for Compliance: There are at least three different sources from where a database connection could be made from in this particular environment: i) a host in the internal network, ii) a host in the DMZ other than the one running BES, iii) and the host running BES in the DMZ. This test is based on the second approach as that allows the test to be run from the more likely staging area for an attack (the DMZ) but doesn't require installing any software on production machines, which could become resources for threat agents.

Compliance can be tested for as follows.

1. Select an SQL query tool that can establish a connection (preferably a JDBC one since that is what BES uses) to a database. One such tool is Database Pilot[39].
2. Configure the tool to connect to the database using the URL supplied to BES. Valid username and password credentials can be used to demonstrate access is possible, however invalid credentials can also be used since the purpose of the test is to determine whether a network connection to the database can be established.
3. Attempt to establish a connection to the database.
4. If the connection attempt is successful, the database will prompt for the user to authenticate. This will demonstrate that the database connections are not being restricted as specified in this Control.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to request a new connection to the database. The response (indicating non-compliance) is that the connection with the database is established.

[39] DatabasePilot is a utility of the JBuilder IDE from Borland Software Corp.

Control 5

Source: original work

Control Objective: Restrict database permissions granted to BES JDBC Connections

Risk Addressed: Addresses threats identified by T1.1/3/4. In those threats, login credentials are obtained by various means and used to establish connections with the database other than from BES (only T1.2 used connections established by BES). Since normal operations do not require the application to be able to alter the database schema or create new user accounts in the database or other such DB administration activities, having such privileges associated with these credentials means excessive privilege would also be granted to unauthorized parties. Reducing the privileges to only those required by the application will reduce the impact to the data should unauthorized parties obtain the credentials.

Compliance Criteria: Database privileges should not include schema manipulation privileges.

Type of Control: Preventative.

A Test for Compliance: One test for compliance would be to attempt to create a new table in the database. This is a non-destructive test. (Of more concern would be whether the user is allowed to drop or alter tables. However that has too much potential to cause damage to the system to be acceptable in a production environment.)

Compliance can be tested for as follows.

1. Login to the database using the same credentials that are assigned to BES (as configured in its Datasource objects).
2. Create a table. For example, executing the following SQL will create a table named audit_table with one column (c1) its primary key: "create TABLE audit_table (c1 VARCHAR (10) PRIMARY KEY)".
3. If the table was created then the system is non-compliant.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to request a change to the database schema (creating a new table). The response (indicating non-compliance) is that a new table is created.

Addressing Threat T2

A summary of the controls addressing threat T2 - the threat of gaining access to application's data via unauthorized access to application.

Ref.	Control Objective	Threat Ref.	Compliance Measure	Test for Compliance
C6.	Keep user credentials confidential while in transit between browser-BES.	T2.1	Encrypt communication traffic between browser and BES webserver.	Submit known tokens as authn credentials and inspect communications stream for presence of them. Test is stimulus/response. Test is objective.
C7.	Keep user credentials confidential while in transit between BES and LDAP.	T2.2	Encrypt communication traffic between BES and LDAP authentication server.	Submit known tokens as authn credentials and inspect communications stream for presence of them. Test is stimulus/response. Test is objective.
C8.	Protect application against password discovery by enumeration	T2.3	System accepts no more than one authentication request per 3 seconds.	Use programmatic or scripting utility (such as Brutus) to submit high volume of authn requests. Test is stimulus/response. Test is objective.
C9.	Remove sample security configurations installed with BES in application	T2.4	Sample user accounts are not valid for login. Host-based accounts are not valid for login.	Login attempts to application using default user accounts or host-based account are unsuccessful. Test is stimulus/response. Test is objective.

Table 7. Summary of Controls addressing Threat T-2

Control 6

Source: original work

Control Objective: Keep user credentials confidential while in transit between browser and BES

Risk Addressed: This control addresses threat T2.1 where login credentials that are exchanged between the browser and BES webserver could be observed, captured and used to gain unauthorized access to the application. Therefore, this control requires that all information exchanged between the browser and BES webserver not be decipherable by a 3rd party.

Compliance Criteria: Login credentials exchanged between user's browser and BES are not decipherable by a 3rd party. This compliance metric is binary but depends upon the definition of decipherable.

Type of Control: Preventative.

A Test for Compliance: A test for compliance with this control is to monitor the communication between a browser and BES webserver to see whether a known token (the password) can be observed in transit. If

the tokens are observed then it demonstrates that the credentials are accessible by a 3rd party. (Note. This compliance measure depends upon the capabilities of the observing 3rd party – specifically, their ability to decipher the stream. In this test that capability is interpreted weakly; meaning that the 3rd party is only able to decipher unencrypted streams. In reality, the 3rd party maybe able to decipher weakly encrypted streams as well.)

Compliance can be tested for as follows.

1. Devise two alphanumeric strings that will be used for a username and password. Since this test does not require the credentials to be valid, use of random strings will ensure the audit evidence does not record confidential information (valid credentials).
2. Begin monitoring communication between the browser and BES webserver. Possible access points include the network stack on the browser's host, the network path between hosts, and the network stack on the BES host. All access points are equally eligible since the entire path between the browser and BES webserver should be encrypted. Probably the simplest approach is to insert an HTTP proxy between the browser and webserver that will require modifying the browser's network settings to direct its traffic through the proxy. An HTTP Proxy such as Achilles[40] will allow the HTTP traffic to be easily observed.
3. From the browser, request the application's login page and enter the tokens selected above for the username and password.
4. Submit the credentials entered in the HTML Login Form to the BES webserver.
5. Inspect the communication stream between the browser and BES webserver for the presence of the base-64 encoded versions of the tokens.

If the tokens were observed in the communication stream, then the test demonstrates that the login credentials were sent unencrypted. Hence they could be intercepted and used by an unauthorized 3rd party.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to send authentication credentials across a vulnerable path. The response (indicating non-compliance) is that the credentials can be observed in transit along the path.

Control 7

Source: original work

Control Objective: Keep user credentials confidential while in transit between BES and LDAP

Risk Addressed: This control addresses threat T2.2 where login credentials that are exchanged between BES and the LDAP authentication server could be observed, captured and used to gain unauthorized access to the application. Therefore, this control requires that all information exchanged between BES and the LDAP server not decipherable by a 3rd party.

Compliance Criteria: Login credentials exchanged between BES and LDAP are not decipherable by a 3rd party. This compliance metric is binary but depends upon the definition of decipherable.

Type of Control: Preventative.

A Test for Compliance: A test for compliance with this control is to monitor communications between BES and the LDAP server to see whether a known token (the password) can be observed in transit. If such a token is observed, then it demonstrates that the credentials are accessible by a 3rd party. (Note. This compliance measure depends upon the capabilities of the observing 3rd party – specifically, their ability to decipher the stream. In this test that capability is interpreted weakly; meaning that the 3rd party is only able

[40] Achilles HTTP Proxy http://packetstormsecurity.nl/filedesc/achilles-0-27.zip.html

to decipher unencrypted streams. In reality, the 3rd party maybe able to decipher weakly encrypted streams as well.)

Compliance can be tested for as follows.

1. Devise two alphanumeric strings that will be used for a username and password. Since this test does not require the credentials to be valid, use of random strings will ensure the audit evidence does not record confidential information (valid credentials).
2. Begin monitoring communication between BES and LDAP provider. There are several possible points to access the communication stream including the BES host's network stack, the network path between hosts, and the network stack on the LDAP host. All access points are equally eligible since the entire path between BES and the LDAP process should be encrypted. Utilities such as ngrep[41] can be used to monitor the network stack of a host, and utilities such as dsniff[42] can be used to monitor the communication stream between the hosts.
3. Login a-new to the application, using the credentials devised in step 1 above.
4. Inspect the communication stream between BES and the LDAP server for the presence of the tokens. Note. Dsniff is able to parse the LDAP protocol to identify authentication credentials. This will help locate the credentials if they are sent over an unencrypted stream.

If the tokens were observed in the communication stream, then the test demonstrates that the login credentials were sent unencrypted and hence could be intercepted and used by an unauthorized 3rd party.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to send authentication credentials across a vulnerable path. The response (indicating non-compliance) is that the credentials can be observed in transit along the path.

Control 8

Source: original work

Control Objective: Protect application against password discovery by enumeration

Risk Addressed: This control addresses threat T2.3 where login credentials might be 'guessed' by a 3rd party and used to gain unauthorized access to the application. For practical purposes, the 3rd party needs to be able to search large blocks of the space within a reasonable amount of time. Therefore, this control requires that the rate at which authentication can be attempted be limited such that it is reasonable for humans but not practical for 'guessing' passwords by brute force.

Compliance Criteria: The system limits authentication attempts to 1 every 3 seconds. At this rate, a full scan of an 8-digit alphanumeric password space (36^8 combinations) would be impractical (2.68x 10^5 years). Yet it does not inconvenience legitimate users performing normal tasks.

Type of Control: Deterrent. This control does not prevent the threat of password cracking. It is possible that a password could be found in one attempt. Also, other assumptions maybe used to reduce the search space and thus reduce the time required (on average) to find a valid username/password combination. Because this is only a deterrent, Control 17 is used in conjunction to detect the event.

A Test for Compliance: A test for compliance with this control is to programmatically submit more than one authentication request per 3-seconds. Since this is a web-based application, utilities such as Brutus[43] can be used. If a rate of more than one authentication request per 3 seconds is accepted by the system, then it is not compliant.

---

[41] Eyler. "ngrep"
[42] Song. "dsniff"
[43] Brutus

Objectiveness: This test is subjective because the rate determined for compliance is a subjective one. That is, this value is believed to balance the needs of users to authenticate and the needs of security staff to thwart attacks. However, the organization may decide that the right balance is achieved at a different value.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to send authentication requests to the application. The response (indicating non-compliance) is that a high rate of requests (more than 1 per 3 seconds) is processed by the application.

Control 9

Source: original work

Control Objective: Don't allow default security configurations (such as accounts and authorization domain declarations) that were installed with the product to remain in production.

Risk Addressed: This control addresses threat T2.4 where known login credentials, known authentication domain configurations, and known authorization domain mappings are installed along with the BES product. If those sample resources are not removed, then they could provide a means for unauthorized access to the application.

Compliance Criteria: The system does not accept these well-known credentials or valid host accounts for authentication to the application.

Type of Control: Preventative.

A Test for Compliance: A test for compliance with this control is to attempt to authenticate against the application using one of the well known accounts or attempt to authenticate using a valid host account.

The sample accounts[44] installed with the product are: admin/admin, jeeves/jeeves, pclare/pclare, borland/borland, and joeshopper/joepass. To test whether the system is still configured to recognize this authentication realm, attempt to login to the application using one of these identities.

Similarly, there is another default authentication realm defined - HostRealm. This authenticates the credentials using the host's own authentication mechanism. Thus, a test to determine whether the system is still configured to recognize this realm is to attempt to login to the application with an identity valid on the BES host.

If the credentials from either of these sources are accepted, then the system still includes the default settings for authentication that the product was installed with and hence is not compliant with this control.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to use authentication credentials installed with the sample authentication database. The response (indicating non-compliance) is that the credentials would be accepted by the application as valid.

---

[44] The list of users was extracted from <BES>/var/servers/<svr>/adm/security/userdb.jds which is installed by default, using the Borland supplied utility userdbadmin, as follows:

prompt> userdbadmin -db jdbc:borland:dslocal:./userdb.jds -user admin -password admin –interactive

>listusers

admin: [user]
jeeves: [user associate]
pclare: [user]
borland: [user]
joeshopper: [client]

The passwords can be tested against this database directly using userdbadmin's test command.

Addressing Threat T3

A summary of the controls addressing threat T3 - the threat of gaining access to application by overriding infrastructure's security configuration setup.

Ref	Control Objective	Threat Ref.	Compliance Measure	Test for Compliance
C10.	Protect config files from being written to by users who are not in the user group assigned to BES.	T3.1	No write permissions for the 'other' group on security configuration files in the BES installation directory.	Use shell commands to append a comment line to a security configuration file, as a user not in the group assigned to BES. Test is stimulus/response. Test is objective.
C11.	Prevent Partition's hosted components from modifying configuration files by using appropriate JVM security policies	T3.2	Java security policy of the Partition's JVM denies J2EE components write privileges to BES's security configuration files.	Run an EJB component within the Partition that attempts to write to a security configuration file. Test is stimulus/response. Test is objective.
C12.	Prevent Partition process from modifying configuration files by using appropriate OS privileges	T3.2	Operating system identity of the Partition process is different from all other BES processes, and has insufficient privilege to write to BES's security configuration files.	Run an EJB component within the Partition that attempts to write to a security configuration file. Test is stimulus/response. Test is objective.
C13.	Remove sample user accounts and security configurations that install with BES	T3.3	Sample user accounts are not valid for login. Host-based accounts are not valid for login.	Login attempts to application using default user accounts or host-based account are unsuccessful. Test is stimulus/response. Test is objective.
C14.	Protect system against password discovery by enumeration	T3.4	System accepts no more than one authentication request per 3 seconds.	Use programmatic or scripting utility (such as Brutus) to submit high volume of authn requests. Test is stimulus/response. Test is objective.
C15.	Protect credentials while in transit between Admin clients (e.g. Console or iastool) and BES, and between BES and LDAP	T3.4	Encrypt communication traffic with BES administration server (IAS), and with LDAP.	Submit known tokens as authn credentials and inspect communications stream for presence of them. Test is stimulus/response. Test is objective.

Table 8. Summary of Controls addressing Threat T-3

Control 10

Source: original work

Control Objective: Protect config files against direct access (e.g. shell access).

Risk Addressed: This control addresses threat T3.1where the system's security configuration could be altered via direct edits to the system's configuration files. Such an exploit requires write access to the configuration files.

Compliance Criteria: The configuration files of the BES server and its partitions should not allow any write access except by users of the group established for the BES system. Those files are enumerated below.

The files containing security configurations for a BES server (<svr>) are the following:

<BES>/var/servers/<svr>/adm/security/java_security.policy
<BES>/var/servers/<svr>/adm/security/management.rolemap
<BES>/var/servers/<svr>/adm/security/management_config.jaas
<BES>/var/servers/<svr>/adm/properties/management_required_roles.properties
<BES>/var/servers/<svr>/adm/properties/management_vbroker.properties
<BES>/var/servers/<svr>/adm/properties/systemDefaults.properties

The files containing security configuration for a BES Partition (<partition>) are the following:

<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/security/config.jaas
<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/security/default.rolemap
<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/security/java_security.policy
<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/security/management_config.jaas
<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/partition.classpath
<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/management_required_roles.properties
<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/management_vbroker.properties
<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/partition.properties
<BES>/var/servers/<svr>/adm/properties/partitions/<partition>/vbroker.properties

Type of Control: Preventative.

A Test for Compliance: Search for all files in the <BES-install>/var/servers/<svr>/adm directory and its subdirectories which are writable by the 'other' group. This can be achieved with the command

```
prompt> cd BES/var/servers/svr/adm
prompt> find . -type f -perm +2 -ls
```

which lists files of the current directory and its subdirectories that are writable by 'other' (perm=2). If the implementation is compliant, then this command will not list (match) any files from the above set.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: The above test was not a stimulus / response test but is easily scripted. However, the following test is a manual stimulus / response test.

1. Login to the host as a user that does not belong to the access group assigned to the BES system. (e.g. audit / audit). This can be determined by using the groups(1) command to return a list of groups to which this user belongs.

2. Attempt to modify one of the above files in a non-destructive way, for example, by adding a comment. The following command will append a dated comment line to the file named *filename*.

   ```
   prompt> echo "#This line added by auditor on " `date` >> filename
   ```

3. View the contents of the same file to see whether the contents were updated. The tail command can be used for this:

```
prompt> tail filename
```

4. If a comment line appears in the file with the creation date of when step 2 was performed, then the system allows unauthorized users to update its security configuration files and thus it is not compliant with this control. A compliant response would be that no such line exists in the file. The acceptance or rejection of step2 by the shell should not be used as the compliance criteria; but rather the contents of file as in step 3 above.

Control 11

Source: original work

Control Objective: Prevent Partition's hosted components from modifying files by using proper JVM security policies.

Risk Addressed: This control addresses threat T3.2 where the system's security configuration could be altered by J2EE components hosted within a Partition. The exploit uses the Partition processes' operating system privileges to access configuration files that might otherwise not permit write access to users outside the BES group.

Compliance Criteria: The Partition's Java security policy for hosted components does not permit write access to any file within the <BES>/adm/security directory, the <BES>/adm/properties directory, or any of those subdirectories

Type of Control: Preventative.

A Test for Compliance: A compliant Partition would have the following entries in its java_security.policy file. (The location of that file is determined by the Partition's server.security.policy property defined in its parition.property file. The default value for this points to the file <BES>/var/servers/<svr>/adm/properties/partitions/<partition>/security/java_security.policy.)

```
grant codeBase "file:${server.instance.root}/partitions/standard/ears/-" {
  permission java.io.FilePermission "<<ALL FILES>>", "";
};

grant codeBase "file:${server.instance.root}/partitions/standard/ejb_jars/-" {
  permission java.io.FilePermission "<<ALL FILES>>", "";
};

grant codeBase "file:${server.instance.root}/partitions/standard/wars/-" {
  permission java.io.FilePermission "<<ALL FILES>>", "";
};

grant codeBase "file:${server.instance.root}/partitions/standard/rars/-" {
  permission java.io.FilePermission "<<ALL FILES>>", "";
};
```

These entries deny file access for all J2EE components.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: The above test is not a stimulus / response, nor is it easily scripted or automated. This is because many of the rules we are interested in are the denial of permission, which can not be explicitly stated in a Java security policy file[45]. To rely on human inspection for control

[45] It seems odd (at least to me) that Java does not support an explicit 'deny' statement in its security policy grammar. In Java, to deny a permission is to not grant it. But with such a flexible policy definition tool,

conformance (particularly with this type of file/grammar) is to be exposed to human error. Hence, the following stimulus / response test is proposed to allow automatically verifiable evidence.

A stimulus / response test for this control is to load a J2EE component into the Partition and have it attempt a non-destructive file operation on one of the identified security configuration files. If the file can be updated, then this would indicate that the system is not compliant w.r.t. this control.

Such a test is performed by the EJB – FileIOTestEJB. This is defined in the Appendix B. It duplicates a specified file and prepends a comment line to the contents of the duplicate it indicating when the operation took place and that is was generated by this test EJB. This action was chosen rather than modifying the original file since that is part of the system's configuration, which if corrupted would impact its operations (in production). Yet, the test needs to demonstrate that the EJB has access to a system configuration file. Hence the duplicate activity was chosen. However, a malicious EJB would modify the configuration file directly.

Control 12

Source: original work

Control Objective: Prevent Partition process from modifying configuration files by using appropriate OS privileges

Risk Addressed: This control addresses threat T3.2 where the system's security configuration could be altered by J2EE components hosted within a Partition. The exploit uses the Partition processes' operating system privileges to access configuration files that might otherwise not permit write access to users outside the BES group. Control-11 addresses the same risk in a different way. There, Java's security policy feature was used to prevent components from accessing underlying operating system resources. In this control, operating system features are used. Combining these two controls would provide a multi-layered defense against the threat.

Compliance Criteria: The Partition's operating system process executes under a different user identity than the rest of the BES processes. The identity has read access to files in the BES file system, but does not have write access to any file within the <BES>/adm/security directory, the <BES>/adm/properties directory, or any of those subdirectories.

Type of Control: Preventative.

A Test for Compliance: A compliant system would have the following configured:

1. A separate user identity exists for the Partition process. For example uid='partition', gid='borland'.
2. The Partition process executable has the setuid bit enabled and the file's owner is that of the identity from step1.
3. No files of the <BES>/adm/security or <BES>/adm/properties directories are writable by the user of step1.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: The above test is not a stimulus / response, nor is it easily scripted or automated. The stimulus / response test devised for Control-11 can also be used here as a stimulus

---

which can easily introduce permissions with sweeping ranges, relying on such an important fact to be implicitly present is a serious concern. I would rather see the policy tool be able to accept deny statements, and if possible flag conflicting permission statements during startup. I would prefer to be able to add statements such as the following to a policy file to explicitly establish the desired configuration:

```
deny codeBase "file:${server.instance.root}/partitions/standard/rars/-" {
  permission java.io.FilePermission "file:${server.instance.root}/adm/security/-",
    "write";
};
```

/response test for this control since the exploits are the same eventhough the vulnerabilities are slightly different.

Control 13

Source: original work

Control Objective: Remove sample user accounts and security configurations that install with BES.

Risk Addressed: This control addresses threat T3.3 where sample user accounts and realm definitions installed with the BES installation are exploited to gain unauthorized access to either BES administration functions or hosted applications. The sample user accounts have well known usernames and passwords that can be used to authenticate to BES as an administrator. Likewise, the sample management_config.jaas configuration file also defines a host-based authentication realm that uses the host's authentication service to authenticate users. Both of these sample configurations should be removed from production systems.

Compliance Criteria: Sample user accounts are not valid for logging into BES administration roles. Also, host-based accounts are not valid for login.

Type of Control: Preventative.

A Test for Compliance: The following two tests determine whether the system is compliant with this control:

1. One or more of the following must be true:

   a) The ServerRealm definition defined in the System's security configuration file <BES>/var/servers/<svr>/adm/security/management_config.jaas by default (see Figure 10) has been removed from the file, or

   b) If the ServerRealm definition is present, then its URL property is altered to no longer reference the default userdb.jds file, or

```
ServerRealm {
  com.borland.security.provider.authn.BasicLoginModule required
    DRIVER=com.borland.datastore.jdbc.DataStoreDriver
    URL="jdbc:borland:dslocal:adm/security/userdb.jds"
    TYPE=BASIC
    LOGINUSERID=admin
    LOGINPASSWORD=admin;
};
```

Figure 10. Sample ServerRealm definition installed with BES

   c) The users of the userdb.jds file have been removed. This can be tested by running the BES utility

```
prompt>userdbadmin -db jdbc:borland:dslocal:./userdb.jds \
       -user admin -password admin listusers
```

2. The HostRealm definition (shown in Figure 11) has been removed from the same file.

```
HostRealm {
  com.borland.security.provider.authn.HostLoginModule required;
};
```

Figure 11. Sample HostRealm definition installed with BES

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: The above test was not a stimulus / response test. A stimulus/response test for compliance would be to attempt to login to BES's administration functions using a sample user account (e.g. admin/admin). The BES command-line tool - iastool - can be used for this purpose as follows:

1. First ensure that the IAS process of BES is running. It is not necessary that a Partition be running, just IAS as it is this process which embodies BES's authentication configuration.
2. Invoke iastool with the command –ping against the BES server (*servername*).

   ```
   prompt> iastool –ping –server servername
   ```

3. The iastool will need to authenticate the caller before performing the action (ping). The first step of the authentication process is for the user to select an authentication realm from among the configured choices.
4. A compliant system will not offer HostRealm as an authentication realm choice.
5. Enter the credentials of a sample account: e.g. admin/admin.
6. If the iastool returns the results of the ping function, then the sample accounts are still configured in the system and hence the system is not compliant. If, however, iastool reports that authentication failed, then these sample accounts are not valid for gaining access to the system and hence the system is compliant.

Control 14

Source: original work

Control Objective: Protect system against password discovery by enumeration

Risk Addressed: This control addresses threat T3.4 where login credentials might be 'guessed' by a 3rd party and used to gain unauthorized access to the system's administration roles. For practical purposes, the 3rd party needs to be able to search large blocks of the space within a reasonable amount of time. Therefore, this control requires that the rate at which authentication can be attempted be limited to a value which is reasonable for humans but not practical for 'guessing' passwords by brute force.

Compliance Criteria: The system limits authentication attempts to 1 every 3 seconds. At this rate, a full scan of an 8-digit alphanumeric password space (36^8 combinations) would be impractical (2.68x 10^5 years). Yet it does not inconvenience legitimate users performing normal tasks.

Type of Control: Mitigating. It does not prevent the threat of password cracking. It is possible that a password could be found in one attempt. Control 17 is used in conjunction with this one to detect exploits.

A Test for Compliance: A test for compliance with this control is to programmatically submit more than one authentication request per 3-seconds. Since there is not a web-based interface to the BES

administration functions, utilities such as Brutus[46] can not be used. However, communication with BES server administration processes is over CORBA's IIOP and uses CORBA's CSIv2 protocol. Therefore, a programmatic client similar to Brutus in purpose could easily be constructed for this test. Alternatively, a (Perl) script could be written to invoke BES's iastool with parameterized values for –user and –password arguments. This tool communicates with BES processes and requires the user first successfully authenticate themself before it performs their commands. If more than one authentication request per 3 seconds is accepted by the system, then the system is not compliant.

Objectiveness: This test is subjective because the rate determined for compliance is a subjective one. That is, the auditor believes that this value reasonably balances the needs of users to authenticate and the needs of security staff to thwart attacks. However, the organization may decide that right balance is achieved at a different value.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to send authentication requests to the application. The response (indicating non-compliance) is that a high rate of requests (more than 1 per 3 seconds) is processed by the application.

Control 15

Source: original work

Control Objective: Keep user credentials confidential while in transit between BES administration clients (e.g. Console or iastool) and BES, and between BES and LDAP

Risk Addressed: This control addresses threat T3.5 where login credentials that are exchanged between BES administration clients and BES, and between BES and the LDAP authentication server could be observed, captured and used to gain unauthorized access to the system in an administrator's role. Therefore, this control requires that all information exchanged with BES and with the LDAP server not be decipherable by a 3rd party.

Compliance Criteria: Login credentials exchanged with BES's IAS administration process and with the LDAP server are not decipherable by a 3rd party.

Type of Control: Preventative.

A Test for Compliance: A test for compliance with this control is to monitor communications between BES utilities and IAS, and between IAS and the LDAP server to see whether a known token (the password) can be observed in transit. If such a token is observed, then it demonstrates that the credentials are accessible by a 3rd party.

Compliance can be tested for as follows.

1. Devise two alphanumeric strings that will be used for a username and password. Since this test does not require the credentials to be valid, use of random strings will ensure the audit evidence does not record confidential information (valid credentials).
2. Begin monitoring communication with IAS, and between IAS and LDAP provider. There are several possible points to access the communication stream including iastool's host network stack, the network stack of the BES server running the IAS process, the network path between hosts, and the network stack on the LDAP host. All access points are equally eligible since the entire path between the endpoints should be encrypted. Utilities such as ngrep[47] can be used to monitor the network stack of a host, and utilities such as dsniff[48] can be used to monitor the communication stream between the hosts.
3. Launch a BES administration client (e.g. the console or iastool). This will prompt the user to authenticate to the BES administration domain. Use the credentials devised in step 1 above for this.

[46] Brutus.
[47] Eyler. "ngrep"
[48] Song. "dsniff"

4. Inspect the communication stream for the presence of the tokens. Note. Dsniff is able to parse the LDAP protocol to identify authentication credentials. This will help locate the credentials if they are sent over an unencrypted stream.

If the tokens were observed in the communication stream, then the test demonstrates that the login credentials were sent unencrypted and hence could be intercepted and used by an unauthorized 3rd party.

Objectiveness: This test is independently verifiable and repeatable, therefore it is an objective test.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is to send authentication credentials across a vulnerable path. The response (indicating non-compliance) is that the credentials can be observed in transit along the path.

2 Operational Controls

Operational controls compliment and support technical controls. Broadly speaking, technical controls are concerned with the system and its configuration, whereas operational controls are concerned with the personnel who interact with the system. The objective of these controls is to ensure that those interactions promote and sustain a secure operating environment for the system and at a minimum protect the technical controls in place.

The operational controls presented below have been devised specifically to address the primary risks faced by this system. Emphasis has been placed on maintaining data confidentiality and security, as well as availability.

Ref	Control Objective	Threat Ref.	Compliance Measure	Test for Compliance
C16.	Ensure system (particularly data) can be restored properly from a baseline	T4.1	Documented procedure exists to restore entire system configuration and data from a known baseline. Also, a known baseline exists and is available.	Entire system can be restored from a known baseline by following the documented procedure. Test is stimulus/response. Test is subjective.
C17.	Ensure changes to system's config are detectable and result in action	T4.2	Changes are detected automatically. Detection occurs within 4 hours of the event. Notification is automatic.	Introduce a change to the system and observe whether it is detected within 4 hours. Test is stimulus/response. Test is objective.
C18.	Product defects are addressed promptly	T4.3	Critical patches are obtained within 24 hours of release. Newly discovered vulnerabilities in the production environment are addressed within 48 hours.	Simulate release of a new critical patch. Test is not stimulus/response. Test is subjective.
C19.	Login attempts are monitored	T4.4	Login attempts are logged. Logs are reviewed every 24 hours.	Attempt to login to application and observe whether this is recorded in the event log. Test is stimulus/response. Test is subjective.
C20.	Physical access to backup media is restricted	T4.5	Backup media is kept is a safe location which does not permit access by unauthorized individuals	Identify storage location for media and review its access controls. Test is not stimulus/response. Test is subjective.

Table 9. Summary of Controls addressing Threat T-4

Control 16

Source: original work

Control Objective: Ensure system (particularly data) can be restored properly from baseline

Risk Addressed: This control addresses threat T4.1where the system can not be restored from a known baseline. Being able to restore from a baseline is important if it is detected that unauthorized changes have been made. It may also be necessary in situations where a new configuration has been deployed (e.g. a new patch was applied) but failed, although not caught during testing.

Compliance Criteria: Documented procedure exists to restore entire system configuration and data from a known baseline. Also, a known baseline exists and is available.

Type of Control: Preventative.

A Test for Compliance: A test for compliance is to restore the entire system from a known baseline by following the documented procedure. The agent executing the procedure should be a staff member of the organization. The resulting system should be tested to ensure its functionality is sound.

Objectiveness: This test is a subjective one because it relies to some extent on the ability of the person who is following the procedures. It could be that the procedures are ambiguous or vague, but correctly interpreted or even compensated for by an experienced candidate. Eventhough the resulting system could be objectively tested to determine if it is responding correctly, that would not be a direct test of the success of the restoration activity.

A Stimulus/Response Test: The test is a stimulus / response test. The stimulus is to require the system be restored using the documented procedure, and the response is a system restored (if compliant) or not (if not compliant).

Control 17

Source: original work

Control Objective: Ensure changes to system's config are detectable and result in action

Risk Addressed: This control addresses threat T4.2 in which changes made to the system are not tracked and therefore may not be known. Knowing what changes have been made to a system is important: in order to keep the baseline upto date so that if the system is restored, these changes will be included; to ensure that the proper and safe procedure has been followed for introducing changes to a production system; and to ensure unauthorized changes are not made to the system.

Compliance Criteria: Changes made to the system's configuration, whether by mandated procedure or not, are automatically detected. Details of the change, whether authorized or not, are recorded and include what element was changed, when it was changed, and by whom it was changed (if known). These details are communicated to the designated administration staff within 4 hours of the change occurring.

Type of Control: Detective.

A Test for Compliance: Compliance can be tested by making a non-destructive change to the system configuration, such as adding a comment line to the configuration file <BES>/var/servers/<svr>/adm/security/management_config.jaas. A compliant system will:

1. record that a change was made to the configuration
2. record what was changed (the file) and when it was changed.
3. notify the designated administration staff with this information about the event within 4 hours of its occurrence.

Objectiveness: This test is an objective test because the criteria for compliance are quantifiable and independently verifiable. Note, however that time reporting window specified by the control is set to a value which the organization may consider either to short or too long.

A Stimulus/Response Test: This test is a stimulus/response test. The stimulus is the change made to the file. The compliant response is as outlined above. A non-compliant response is any response that does not include all of the results mentioned above.

Control 18

Source: original work

Control Objective: Product defects are addressed promptly

Risk Addressed: This control addresses threat T4.3 in which regular maintenance activities, if not scheduled, are unlikely to be performed when needed. Maintenance activities include obtaining, testing and applying the latest security patches. This control addresses that aspect of the threat.

Compliance Criteria: Critical patches are obtained within 24 hours of release. Newly discovered vulnerabilities in the production environment are addressed within 48 hours.

Type of Control: Corrective.

A Test for Compliance: Compliance can be tested by verifying the following exist:

1. A procedure exists to obtain the latest patches for the product.
2. The procedure is executed at least every 24 hours.
3. A procedure exists to extract the nature of new vulnerabilities identified by a patch, and evaluate the threat of this vulnerability to the system.
4. A procedure exists to apply the patch, test it in a staging environment, and roll it out to production, and optionally to rollback the last changes made to the production configuration.

Objectiveness: This is a subjective test because judging whether or not the procedure's instructions clearly and sufficiently prescribe activities 3 and 4 above is a subjective judgement.

A Stimulus/Response Test: No stimulus/response test was developed for this control.

Control 19

Source: original work

Control Objective: Login attempts are monitored

Type of Control: Detective. This control does not seek to prevent the password-cracking exploit, but to detect it when it occurs. Controls C8 and C14 are preventative controls that aim to prevent the exploit from being used in a practical way – although they do not absolutely prevent the exploit. Thus this control is used in conjunction with those.

Risk Addressed: This control addresses threat T4.4 in which the system monitoring is inadequate, resulting in security incidents occurring but going unnoticed by the organization. This could lead to security being compromised without the organization's knowledge and therefore without response.

Compliance Criteria: Login attempts (successful and unsuccessful) to both BES and application are logged. Logs are reviewed every 24 hours.

Type of Control: Detective. This control does not prevent unauthorized access. It would be used in conjunction with preventative controls (C6, C7, and C9) and to validate their effectiveness, and in conjunction with deterrent controls (C8 and C14) to detect to unauthorized access attempts.

A Test for Compliance: Compliance can be tested by verifying the following exist:

1. Authentication attempts (both successful and unsuccessful) to BES administration functions and Application functions are logged.
2. A procedure exists to review the logs at least every 24 hours for activity that might indicate unauthorized access was obtained.

Objectiveness: This is a subjective test because judging whether or not the procedure's instructions clearly and sufficiently prescribe activity 2 above is a subjective judgement.

A Stimulus/Response Test: A stimulus/response test for this control is to attempt to login to either BES or the application, and then determine whether this event was logged. Further, the organization should then be able to explain whether this event was an unauthorized access or not.

Control 20

Source: original work

Control Objective: Physical access to backup media is restricted

Risk Addressed: This control addresses an element of threat T4.5 in which physical access to the system's backup could be exploited to obtain access to the system's data and its configuration. In particular, this control addresses the vulnerability of the media at rest (i.e. in storage). Other related vulnerabilities such as when the media is being transported would be the subject of other controls (which are not included here).

Compliance Criteria: Backup media is kept in a safe location that does not permit access by unauthorized individuals.

Type of Control: Preventative.

A Test for Compliance: Compliance can be tested by verifying the following:

1. Confirming that all backups are subject to this policy: i.e. that all backups taken of the system either exist in the secure storage area or have been properly disposed of.
2. Confirming that access to the secure storage area is not granted to unauthorized individuals.

Objectiveness: This is a subjective test because the first criteria can not be independently verified.

A Stimulus/Response Test: A stimulus/response test for the second criteria involves attempting to access the secure storage area containing the backup media. A compliant environment will not permit access to an unauthorized individual.

3 Management Controls

Management controls are used to ensure that the security needs of the system are being reported to management and that management is addressing those needs.

Ref	Control Objective	Threat Ref.	Compliance Measure	Test for Compliance
C21.	Ensure that the security of the system can be maintained by receiving necessary management approval for those activities.	T5.1	Position exists within organization responsible for security of system and has sufficient resources and personnel to maintain the system's security.	Verify that management position exists and has the responsibility and budget required. Test is not stimulus/response. Test is subjective.
C22.	Ensure skills needed to perform essential system operations reside with more than one individual within the organization.	T5.2	At least two individuals exist within the organization who are capable of interpreting and performing the instructions of the restoration procedure.	System can be restored by the individuals by following the restoration document (see control C16) Test is stimulus/response. Test is objective.

Table 10. Summary of Controls addressing Threat T-5

Control 21

Source: "Security Architecture: Design, Deployment and Operations"

Control Objective: Ensure that the security of the system can be maintained by receiving necessary management approval and budget for those activities.

Risk Addressed: This control addresses threat T5.1 in which the system's security is not represented adequately at the management level. Consequently, the system may not receive the resources and personnel necessary to maintain the security of the system.

Compliance Criteria: A position exists within the organization that is responsible for the security of the system and has sufficient resources and personnel to maintain the security of the system.

Type of Control: Corrective.

A Test for Compliance: Compliance with this control can be tested by verifying the following:

1. A management position exists within the organization that is responsible and accountable to management for maintaining the security of the system.
2. The manager of the system has been given sufficient resources and personnel for the ongoing maintenance of the system's security.

Objectiveness: This is a subjective test because verifying the second criteria is a subjective judgement.

A Stimulus/Response Test: There is no stimulus/response test designed for this control.

Control 22

Source: original work

Control Objective: Ensure skills needed to perform essential system operations reside in more than one individual within the organization.

Risk Addressed: This control addresses threat T5.2 in which there exists only one individual within the organization with the familiarity and training to correctly restore the operations of the system. Although the restoration procedure maybe correctly documented (see control C16), the instructions within that document may assume a certain level of technical knowledge or even familiarity with the system. Should no other such individual exist within the organization, then restoring the system would be delayed possibly affecting the system's availability and perhaps security.

Compliance Criteria: At least two individuals exist within the organization capable of interpreting and performing the instructions of the restoration procedure.

Type of Control: Recovery.

A Test for Compliance: Compliance with this control can be tested by verifying the following:

1. Two individuals of the organization have been designated for this role.
2. Both individuals are able to successfully restore the system by following the restoration procedure document.
3. Both individuals are able to explain the current system configuration, including where it differs from default values and what the difference entails.

Objectiveness: This is an objective test because the results can be independently verifying.

A Stimulus/Response Test: A stimulus/response test for this control would be to have each individual separately restore the system using the restore document and then test the functionality of the system. If the organization is compliant with this control then the response will be a functional system that behaves the same as the current in-production version.

4 Appendix A – Java Source code for Test Case 2

start of JNDIBrowserTest.java

```
package com.camacit.MQRaudit.tc02;

/**
 * Title: MQR-Audit TestCase02
 * Description: A JNDI Client which browses the serial://datasources context
 * for a specified datasource entry, then retrieves the login credentials of
 * that.  The extracted credentials are compared against the credentials
 * supplied to the object by DsPassword() and DsUsername().  This client was
 * developed to work with Borland Enterprise (J2EE) Server 5.x, but is easily
 * ported to other platforms.
 * Copyright: Copyright (c) 2004
 * Company: Camac IT Ltd
 * @author Brenton Camac
 * @version 1.0
 */

import javax.naming.*;
import javax.naming.NamingEnumeration;
import com.borland.javax.sql.DataSourceProperties;
import com.inprise.j2ee.jndi.CtxFactory;

public class JNDIBrowserTest {

  public static void main(String[] args) {
    JNDIBrowserTest testObj = new JNDIBrowserTest();

    // initialize test client with the JNDI name of the datasource to test against
    testObj.setDsDriverName("Audit/ds1Driver");

    // initialize test client with login credentials expected from datasource
    testObj.setDsUsername("audituser-3D4y7RS9");
    testObj.setDsPassword("auditpass-3O1b77sA");

    // execute test...
    if (testObj.executeTest()) {
      System.out.println("Successful in retrieving datasource credentials from JNDI Naming Service");
    }
    else {
      System.out.println("Unsuccessful in retrieving datasource credentials from JNDI Naming Service");
    }
  }

  /**
   * Retrieve the datasource from JNDI specified by DsDriverName property,
   * extract login credentials from it and compare against the
   * DsUsername and DsPassword properties of this object.
   * @return true if the two sets of credentials are identical, false otherwise
   */
  public boolean executeTest() {
    Context c = getDataSourcesContext();
```

```
  if (c == null)
    return false;

  DataSourceProperties p = getDataSourceProperties(c);
  if (p == null)
    return false;

  return testCredentials(p);
}

public JNDIBrowserTest() {
}

public void setDsDriverName(String s) {
  dsDriverName = s;
}

public void setDsUsername(String s) {
  dsUsername = s;
}

public void setDsPassword(String s) {
  dsPassword = s;
}

private Context getDataSourcesContext() {
  try {
    InitialContext ic = new InitialContext();
    // retrieve bindings of the 'serial://' context
    NamingEnumeration e = ic.listBindings("serial://");
    Binding b;

    while (e.hasMore()) {
      // get next binding from serial:// context
      b = (javax.naming.Binding) e.next();

      // hasMore() can return true even if its an empty enumeration ?!?
      if (b == null) {
        break;
      }

      // find datasources subcontext
      if (b.getName().equals("datasources")) {
        // return the naming context for 'serial://datasources'
        return (Context) b.getObject();
      }
    }
    System.out.println("Unable to list bindings of serial:// naming context.");
    return null;
  }
  catch (NamingException ex) {
    System.err.println("Unable to list bindings of serial:// naming context.");
    System.err.println(ex);
    return null;
  }
}
```

```
 private DataSourceProperties getDataSourceProperties(Context c) {
  try {
   return (com.borland.javax.sql.DataSourceProperties) c.lookup(dsDriverName);
  }
  catch (NamingException ex) {
   System.err.println("Datasource Driver: " + dsDriverName +
                      ", not found in serial://datasources");
   return null;
  }
 }

 private boolean testCredentials(DataSourceProperties p) {
  return (dsUsername.equals(p.getUser()) && dsPassword.equals(p.getPassword()));
 }

 private InitialContext ic;
 private String dsDriverName;
 private String dsUsername;
 private String dsPassword;

}
```

end of JNDIBrowserTest.java

The following is the jndi-defnitions.xml file which must be part of the DAR file deployed to the Naming Service. The highlighted properties are those which were expected by the test client above.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jndi-definitions PUBLIC "-//Borland Corporation//DTD JndiDefinitions//EN"
"http://www.borland.com/devsupport/appserver/dtds/jndi-definitions.dtd">
<jndi-definitions>
  <visitransact-datasource>
    <jndi-name>serial://datasources/Audit/ds1</jndi-name>
    <driver-datasource-jndiname>serial://datasources/Audit/ds1Driver</driver-datasource-jndiname>
    <property>
      <prop-name>connectionType</prop-name>
      <prop-type>Enumerated</prop-type>
      <prop-value>Direct</prop-value>
    </property>
    <property>
      <prop-name>maxPoolSize</prop-name>
      <prop-type>Integer</prop-type>
      <prop-value>2</prop-value>
    </property>
  </visitransact-datasource>
  <driver-datasource>
    <jndi-name>serial://datasources/Audit/ds1Driver</jndi-name>
    <datasource-class-name>com.borland.javax.sql.JdbcConnectionFactory</datasource-class-name>
    <log-writer>False</log-writer>
    <property>
      <prop-name>user</prop-name>
      <prop-type>String</prop-type>
      <prop-value>audituser-3D4y7RS9</prop-value>
    </property>
    <property>
      <prop-name>password</prop-name>
      <prop-type>String</prop-type>
      <prop-value>auditpass-3O1b77sA</prop-value>
    </property>
    <property>
      <prop-name>url</prop-name>
      <prop-type>String</prop-type>
      <prop-value>jdbc:borland:dslocal:audit.jds</prop-value>
    </property>
  </driver-datasource>
</jndi-definitions>
```

5 Appendix B – Java source code for Test Case 11

Source code for EJB Client (FileIOTestEJBClient.java):

start of FileIOTestEJBClient.java

```
package com.camacit.MQRaudit.tc11.ejb;

/**
 * Title: MQR-Audit TestCase11
 * Description: A J2EE component that copies the contents of a specified file
 * into a new file of the same name with a .audit suffix appended.  A comment
 * is inserted as the first line of the new file indicating when the test was
 * performed.
 * Copyright: Copyright (c) 2004
 * Company: Camac IT Ltd
 * @author Brenton Camac
 * @version 1.0
 */

import javax.naming.*;
import javax.rmi.PortableRemoteObject;
import java.rmi.*;

public class FileIOTestEJBClient {

 //Main method

 public static void main(String[] args) {
  if (args.length != 1) {
   System.err.println("Usage: <program>; sourceFileName");
  }

  FileIOTestEJBClient client = new FileIOTestEJBClient();
  // Use the client object to call one of the Home interface wrappers
  // above, to create a Remote interface reference to the bean.
  // If the return value is of the Remote interface type, you can use it
  // to access the remote interface methods.  You can also just use the
  // client object to call the Remote interface wrappers.
  FileIOTestEJB remoteIF = client.create();

  try {
   System.out.println("Setting sourceFilename to " + args[0]);
   remoteIF.setSourceFilename(args[0]);
  }
  catch (RemoteException ex) {
   System.err.println("Remote exeception encountered while executing ejb.setSourceFilename()");
   System.err.println(ex);
   return;
  }

  try {
   System.out.println("Executing test: EJB will duplicate the specified source file.");
   remoteIF.executeTest();
   System.out.println("Test executed at " + new java.util.Date().toString());
```

```
  }
  catch (RemoteException ex) {
    System.err.println("Remote exeception encountered while executing ejb.executeTest()");
    System.err.println(ex);
    return;
  }
}



//Construct the EJB test client
public FileIOTestEJBClient() {
  initialize();
}

public void initialize() {
  long startTime = 0;
  if (logging) {
    log("Initializing bean access.");
    startTime = System.currentTimeMillis();
  }

  try {
    //get naming context
    Context context = new InitialContext();

    //look up jndi name
    Object ref = context.lookup("FileIOTestEJB");

    //look up jndi name and cast to Home interface
    fileIOTestEJBHome = (FileIOTestEJBHome) PortableRemoteObject.narrow(ref,
        FileIOTestEJBHome.class);

    if (logging) {
      long endTime = System.currentTimeMillis();
      log("Succeeded initializing bean access through Home interface.");
      log("Execution time: " + (endTime - startTime) + " ms.");
    }
  }
  catch (Exception e) {
    if (logging) {
      log("Failed initializing bean access.");
    }
    e.printStackTrace();
  }
}

//-------------------------------------------------------------------------
// Methods that use Home interface methods to generate a Remote interface reference
//-------------------------------------------------------------------------

public FileIOTestEJB create() {
  long startTime = 0;
  if (logging) {
    log("Calling create()");
    startTime = System.currentTimeMillis();
```

```
  }
  try {
    fileIOTestEJB = fileIOTestEJBHome.create();
    if (logging) {
      long endTime = System.currentTimeMillis();
      log("Succeeded: create()");
      log("Execution time: " + (endTime - startTime) + " ms.");
    }
  }
  catch (Exception e) {
    if (logging) {
      log("Failed: create()");
    }
    e.printStackTrace();
  }

  if (logging) {
    log("Return value from create(): " + fileIOTestEJB + ".");
  }
  return fileIOTestEJB;
}

//---------------------------------------------------------------------
// Methods that use Remote interface methods to access data through the bean
//---------------------------------------------------------------------

public void setSourceFilename(String s) {
  if (fileIOTestEJB == null) {
    System.out.println("Error in setSourceFilename(): " + ERROR_NULL_REMOTE);
    return;
  }

  long startTime = 0;
  if (logging) {
    log("Calling setSourceFilename(" + s + ")");
    startTime = System.currentTimeMillis();
  }

  try {
    fileIOTestEJB.setSourceFilename(s);
    if (logging) {
      long endTime = System.currentTimeMillis();
      log("Succeeded: setSourceFilename(" + s + ")");
      log("Execution time: " + (endTime - startTime) + " ms.");
    }
  }
  catch (Exception e) {
    if (logging) {
      log("Failed: setSourceFilename(" + s + ")");
    }
    e.printStackTrace();
  }
}

public void executeTest() {
  if (fileIOTestEJB == null) {
```

```
    System.out.println("Error in executeTest(): " + ERROR_NULL_REMOTE);
    return;
   }

   long startTime = 0;
   if (logging) {
    log("Calling executeTest()");
    startTime = System.currentTimeMillis();
   }

   try {
    fileIOTestEJB.executeTest();
    if (logging) {
     long endTime = System.currentTimeMillis();
     log("Succeeded: executeTest()");
     log("Execution time: " + (endTime - startTime) + " ms.");
    }
   }
   catch (Exception e) {
    if (logging) {
     log("Failed: executeTest()");
    }
    e.printStackTrace();
   }
  }

  public void executeRemoteCallsWithDefaultArguments() {
   if (fileIOTestEJB == null) {
    System.out.println("Error in executeRemoteCallsWithDefaultArguments(): " +
                       ERROR_NULL_REMOTE);
    return;
   }
   setSourceFilename("");
   executeTest();
  }

  //--------------------------------------------------------------------------
  // Utility Methods
  //--------------------------------------------------------------------------

  private void log(String message) {
   if (message == null) {
    System.out.println("-- null");
    return;
   }
   if (message.length() &GT; MAX_OUTPUT_LINE_LENGTH) {
    System.out.println("-- " + message.substring(0, MAX_OUTPUT_LINE_LENGTH) +  " ...");
   }
   else {
    System.out.println("-- " + message);
   }
  }

  private static final String ERROR_NULL_REMOTE = "Remote interface reference is null.";
  private static final int MAX_OUTPUT_LINE_LENGTH = 100;
  private boolean logging = true;
```

```
  private FileIOTestEJBHome fileIOTestEJBHome = null;
  private FileIOTestEJB fileIOTestEJB = null;

}
```

end of FileIOTestEJBClient.java

Source code of EJB Bean implementation (FileIOTestEJBBean.java):

start of FileIOTestEJBBean.java

```
package com.camacit.MQRaudit.tc11.ejb;

import javax.ejb.*;
import com.camacit.MQRaudit.tc11.FileIOTest;

public class FileIOTestEJBBean implements SessionBean {
  SessionContext sessionContext;
  private FileIOTest testObj = new FileIOTest();

  // ejb functions
  public void ejbCreate() throws CreateException {
  }
  public void ejbRemove() {
  }
  public void ejbActivate() {
  }
  public void ejbPassivate() {
  }
  public void setSessionContext(SessionContext sessionContext) {
    this.sessionContext = sessionContext;
  }

  // bean specific functions
  public void setSourceFilename(String s) {
    System.out.println("FileIOTest: setting sourcefile name to " + s);
    testObj.setSourceFilename(s);
  }

  public void executeTest() {
    System.out.println("FileIOTest: executingTest");
    testObj.executeTest();
  }
}
```

end of FileIOTestEJBBean.java

Source code of EJB Bean implementation (FileIOTest.java):

start of FileIOTest.java

```
package com.camacit.MQRaudit.tc11;

/**
 * Title: MQR-Audit TestCase11
 * Description: An object which copies contents of a file and prepends
 * a comment which includes a timestamp.
 * Copyright: Copyright (c) 2004
```

```
 * Company: Camac IT Ltd
 * @author Brenton Camac
 * @version 1.0
 */

import java.io.*;
import java.util.Date;

public class FileIOTest {

 /**
  * Main method used for stand-alone testing.  Purpose of this test, however,
  * is to be run from within a J2EE Container.
  * @param args name of source file
  */
 public static void main(String[] args) {
   FileIOTest testObj = new FileIOTest();

   if (args.length != 1) {
     System.err.println("Usage: <program>; sourceFileName");
   }

   testObj.setSourceFilename(args[0]);
   testObj.executeTest();

   System.out.println("Test created new file " + testObj.getTargetFilename());
 }

 public void FileIOTest() {
   init();
 }

 private void init() {
   // generate target file name based on source file name
   targetFilename = sourceFilename + ".audit";
 }

 public void setSourceFilename(String s) {
   sourceFilename = s;
   init(); // to regenerate targetFilename
 }

 String getSourceFilename() {
   return sourceFilename;
 }

 String getTargetFilename() {
   return targetFilename;
 }

 /**
  * Utility function to copy the contents of open input stream to open output
  * stream.  Closes input stream when finished.  Optionally closes output
  * stream if instructed to.
  * @param is open input stream
  * @param os open output stream
```

```
 * @param close whether to close output stream when finished.
 * @throws IOException
 */
private static void copyFile(InputStream is, OutputStream os, boolean close) throws
    IOException {
  int b; // storage byte to transfer data between streams
  while ( (b = is.read()) != -1) {
    os.write(b);
  }
  is.close();
  if (close) {
    os.close();
  }
}

/**
 * Perform FileIO Audit Test.  Copy the contents of a file into another
 * innocuous file whose name is the source file name with ".audit" appended.
 * The output file will contain an extra line (the first line) which is a
 * comment indicating when the file was created.  This demonstrates that the
 * program had access to file contents.
 */
public void executeTest() {
  try {
    // open source file
    BufferedInputStream inStream = new BufferedInputStream(new
       FileInputStream(sourceFilename));

    // open target file (will clobber existing target if it exists)
    BufferedOutputStream outStream = new BufferedOutputStream(new
       FileOutputStream(targetFilename));

    // generate a control line based on current time
    byte[] controlLine = new String("// File duplicated from "+sourceFilename+
                            " at " +  new Date().toString() +
                            " by CamacIT's FileIO Audit Test (#11)\n").getBytes();

    // insert control line at first line of target file as a comment
    outStream.write(controlLine);

    // copy all of source file into target file
    copyFile(inStream, outStream, true);
  }
  catch (IOException ex) {
    System.err.println(ex);
  }
}

 private String sourceFilename = "../../adm/security/java_security.policy";    // default source file
 private String targetFilename;
}
```

end of FileIOTest.java

Part III

The Audit Evidence

Table of Contents

Overview of Part III

This is Part III of the Audit Report. It examines how the audit evidence was collected and the results obtained for a selection of the audit's controls.

The following 10 tests were selected from the audit results because they represent a wide-cross section of the types of controls used. Nine of them were stimulus/response tests, the exception being the last one, which is a management control.

Also examined in this section is the risk that remains after the initial audit.

Finally, the system's auditability is also briefly discussed.

1 The Audit Evidence

Test 1: Access to EAR/DAR files

The purpose of this test is to determine whether the EAR/DAR files that encapsulate datasource definitions for the AppServer are accessible by users in general (i.e. users belonging to the 'other' group).

As per the test instructions of Control C1, a search for all files with the name suffix '.dar' or '.ear' and readable by the 'other' group was conducted from the AppServer's root directory (/opt/BES/var/servers/SPOCK). The command used and its results appear in Figure 12.

```
[audit@SPOCK T1]$ find /opt/BES/var/servers/SPOCK/ -type f -name '*.[de]ar' \
  -perm +4 -ls
2769074 9516 -rw-rw-r--   1 borland  borland   9724763 Feb  9 15:19
/opt/BES/var/servers/SPOCK/partitions/standard/ears/Cats.ear
[audit@SPOCK T1]$
```

Figure 12. Searching for EAR/DAR files readable by 'other' group

The command returned one file: Cats.ear, which is an EAR file. This result indicates that the Cats.ear file is readable by the 'other' group. Since members of that user group are not members of the group setup for the BES server, this test demonstrates that the system is not compliant with control C1.

For completeness, the inverse test (searching for EAR/DAR files that are not readable by other) was also performed to check what DAR and EAR files denied read privileges for 'other' – i.e. which files were compliant. That command and its results are shown in Figure 13.

```
[audit@SPOCK T1]$ find /opt/BES/var/servers/SPOCK/ -type f -name '*.[de]ar' \
  ! -perm +4 -ls
[audit@SPOCK T1]$
```

Figure 13. Searching for EAR/DAR files not readable by 'other' group

No files where returned from this command, indicating that there are no .ear or .dar files in this installation of BES which were compliant with this control – i.e. that have been configured with read permission denied to 'other' group.

In addition, the stimulus/response test of C1 is also shown. This test attempts to read the embedded jndi-definitions.xml file of a DAR or EAR file, and locate the XML tags that declare the password properties of a datasource. The command used for this test and its results are shown in Figure 14.

```
[audit@SPOCK T1]$ id
uid=502(audit) gid=502(audit) groups=502(audit)

[audit@SPOCK T1]$ unzip -p \
  /opt/BES/var/servers/SPOCK/partitions/standard/ears/Cats.ear \
  'META-INF/jndi-definitions.xml' | grep 'password'
            <prop-name>password</prop-name>

[audit@SPOCK T1]$
```

Figure 14. Stimulus/Response test gaining access to EAR/DAR file

The following is brief explanation of how what this command is accomplishing. The EAR file format is a Java archive (mandated by the J2EE specification[49]) and so can be read with the unzip utility. In this command, unzip is instructed to send its output to stdout (-p option) and only to read the jndi-definitions.xml file from the META-INF directory. It is the jndi-definitions.xml file that contains datasource configuration information, including the database login credentials. The command then pipes the contents of this file to grep so it can be searched for the keyword 'password' in the stream. The password appears in jndi-definitions.xml as a property, with a property-name of password, property-type of String, and property-value of the password value itself. This command matches part of the password property (the property name XML tag) and prints that line to stdout. The actual password value would be found in a corresponding <prop-value> tag. However, that tag is not searched for in this test because it would result in the confidential database login password being extracted and logged in the audit evidence. Being able to locate the prop-name tag of the password property is sufficient to demonstrate that the password value could likewise be extracted.

This stimulus/response test showed that a user who does not belong to the BES process group could read a deployed EAR file and extract the database login credentials supplied to the application for accessing the database. This result demonstrates (and reinforces conclusion arrived at from the other test) that the system is not compliant with control C1.

Test 2: Access to JNDI Service

The purpose of this test is to determine whether datasources registered with the JNDI naming service can be retrieved and whether their database credentials extracted.

This test does not use the datasource definition used by the application since extracting its database credentials will cause them to appear in the audit evidence. Hence, a separate DAR file (MQRauditTC02.dar) is used instead. The jndi-definitions.xml file of that DAR file is documented in Appendix A of Part II.

The deployment of MQRauditTC02.dar to BES was done by the administrator (according to their Operational Guidelines) using the iastool utility. This pre-test activity is shown in Figure 15 (the username and password values for administrative access to BES have been 'blanked' out with the @ character).

```
[audit@SPOCK T2]$ id
uid=502(audit) gid=502(audit) groups=502(audit)

[audit@SPOCK T2]$ iastool -deploy -jars MQRauditTC02.dar -server SPOCK -partition \
  standard
Deploying modules to partition
```

[49] Allamaraju, p35. "Professional Java Server Programming J2EE 1.3 Edition"

```
Creating an ORB on port 19806...

Please enter security credentials to be used for server discovery
0. Abort
1. ServerRealm
2. HostRealm
Choose realm for authentication: 1
Username: @@@@@@@@@@
Password: @@@@@@@@@@
Generating stubs for MQRauditTC02.dar
Warning: No generated stub files were found.
Stubs successfully generated
Merging archive contents: _bes_stubs.jar
Merging archive contents: MQRauditTC02.dar
Verifying /opt/BES/tmp/iastool/tmpbes_18101/MQRauditTC02.dar
Verifying jndi definitions module MQRauditTC02.dar [role=deployer]
0 errors, 0 warnings
Deploying modules to partition standard
Deploying MQRauditTC02.dar to bes://SPOCK/standard
Enabling modules in partition standard
Deployment completed
Deployed successfully

[audit@SPOCK T2]$

(NB. @@@ = obfuscation of reported evidence by auditor to maintain confidentiality)
```

Figure 15. Deploying a DAR file to the system for subsequent testing

As per the test instructions of Control C2, an attempt was made to retrieve the datasource of MQRauditTC02.dar from the JNDI service by using an external unauthorized client – JNDIBrowserTest.

In the first run of the test, the DAR file was disabled in BES and therefore not loaded into JNDI service. This was done in order to baseline the failure case: i.e. to be able to recognize the failure results when they occur. The test was run under the user 'audit' who is not a member of the BES group. The test and results are shown in Figure 16.

```
[audit@SPOCK T2]$ id
uid=502(audit) gid=502(audit) groups=502(audit)

[audit@SPOCK T2]$ vbj -VBJclasspath test02Client.jar -VBJprop vbroker.agent.port=19741 \
  com.camacit.MQRaudit.tc02.JNDIBrowserTest
Datasource Driver: Audit/ds1Driver, not found in serial://datasources
Unsuccessful in retrieving datasource credentials from JNDI Naming Service

[audit@SPOCK T2]$
```

Figure 16. Retrieving a datasource from JNDI

In the second run of the test, the DAR file was enabled in BES and therefore loaded into its JNDI service. The test and results are shown in Figure 17.

```
[audit@SPOCK T2]$ id
uid=502(audit) gid=502(audit) groups=502(audit)

[audit@SPOCK T2]$ vbj -VBJclasspath test02Client.jar -VBJprop vbroker.agent.port=19741 \
  com.camacit.MQRaudit.tc02.JNDIBrowserTest
Successful in retrieving datasource credentials from JNDI Naming Service

[audit@SPOCK T2]$
```

Figure 17. Retrieving a datasource from JNDI

The results of the second run show that the client was able to contact the JNDI naming service and retrieve a datasource object, then extract from that datasource definition the login credentials used for establishing a JDBC connection with the database.

In a compliant system, an unauthorized client should not be able to retrieve datasources from the JNDI service. If the system were compliant then the second test run should have returned the same result as the first. However, the results of the two were different and the second test run successfully retrieved the credentials from the datasource registered with JNDI. Therefore the system is not compliant with Control C-2.

Test 4: Database connection origins

The purpose of this test is to determine whether connections to the database can be established from processes other than BES. This is an important test because unauthorized individuals who have gained possession of the database login credentials must establish a connection with the database in order to gain access to the data. If a connection can not be made (eventhough the credentials have been obtained), then access to the data will have been prevented.

As per the test instructions of Control C-4, this test is run from a host within the DMZ. The host used was the auditor's laptop since it already has a utility (DBPilot) which can connect to databases, and thus doesn't require installing software on the production hosts.

The instructions of the test were followed.

1. An SQL query tool was selected that can establish JDBC connections with the application's type of database (Oracle). The tool selected was DBPilot; a utility program that is included with Borland's JBuilder v9 Java IDE. To use this tool with Oracle RDBMS requires adding Oracle's JDBC thin driver (classes12.zip) to JBuilder's path (<JB>/lib/ext).

2. The tool was configured to connect to the database by using the URL that is configured for BES. The URL supplied to BES (jdbc:oracle:thin:@192.168.1.102:1521:CATSP) was obtained from the system administrator. Although, since the system failed compliance tests C1 and C2, datasources of the system are readable by any party, and therefore this information could have also been obtained from there directly.

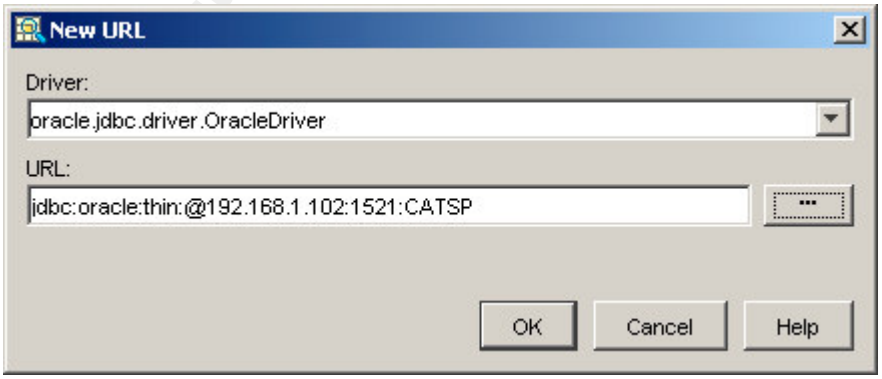

Figure 18. Defining the JDBC URL of the application's database

The tool was also configured with the same login credentials as supplied to the application. Again, these were entered by the administrator, but could have been obtained directly from the datasources of the non-compliant system.

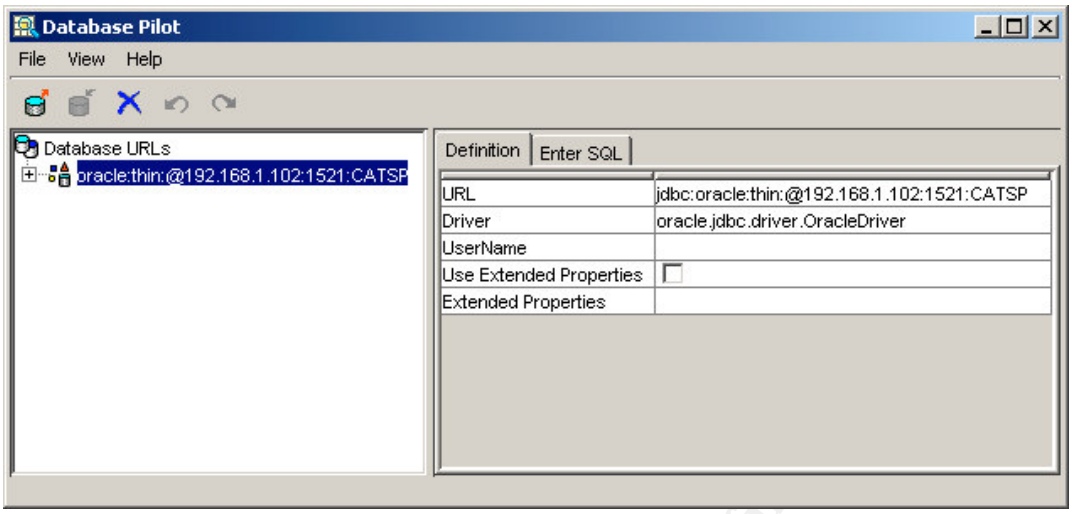

Figure 19. Entering other properties of the datasource

The credentials entered are not shown here in the audit evidence, but were added using via a dialog window (see Figure 20).

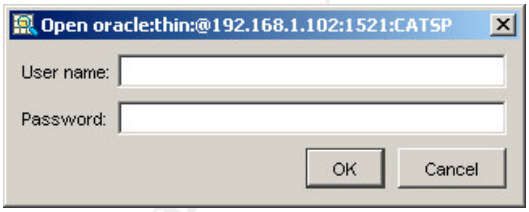

Figure 20. Supplying the database login credentials

3. An attempt was made to establish a connection to the database.

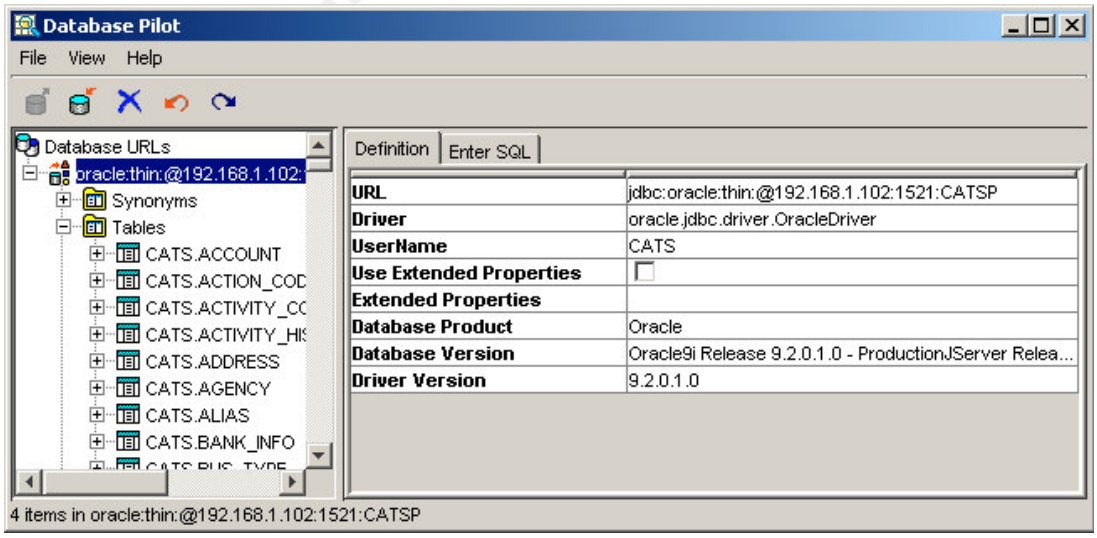

Figure 21. Access to database's schema granted

4. As shown by Figure 21, the connection attempt was successful. This is demonstrated by DBPilot successfully querying the database for its schema.

If a system were compliant with Control C4 then connections to the database from processes other than BES would not be permitted. The preceding test showed that this was not the case; that connections could be established. Hence the system is not compliant with control C-4.

Test 8: Password discovery by enumeration

The purpose of this test is to determine whether application login credentials can be 'guessed' by a threat agent. Control C-8 seeks to limit the rate at which authentication attempts against the application can be made, so as to render password 'guessing' by brute force impractical. This approach deters the threat but does not prevent it. The test is to gauge the effectiveness of that deterrent.

As per the test instructions of Control C-8, a programmatic utility is used to submit authentication requests to the application's web server interface at a rate greater than one request per 3 seconds (the compliance criteria).

The tool selected for this test was Brutus[50]; a well-known tool useful in such activities. The test was run from the auditor's laptop within the DMZ.

Brutus was configured with the particulars for this application:
- use HTTP Form based authentication,
- form target is 192.168.1.3:8080/j_security_check,
- form submit method is HTTP POST,
- form's username field is j_username,
- form's password field is j_password,
- submitted values are encoded.

This information was obtained by reviewing the HTML sent to the client (browser) when a user navigates to the login page of the application.

Next, Brutus was configured for the particulars of this test:
- number of concurrent connections = 4
- timeout 10 seconds.

Since the goal of this test is not to guess the password, but to test the throughput rate of authentication requests, Brutus was configured to use the Brute Force method of password selection; i.e. it calculates a list of word permutations. A specific user ID was selected for all password submissions so that this login activity could be easily identified in the system's logs.

The test was started at 21:50 on February 09, 2004. It was allowed to run for 44 seconds before being explicitly disengaged at the Brutus console (stop command). During that time it made 5580 password guesses. This was a rate of approximately 125 password guesses per second, or 7500 per minute, or 450,000 per hour.

[50] Brutus

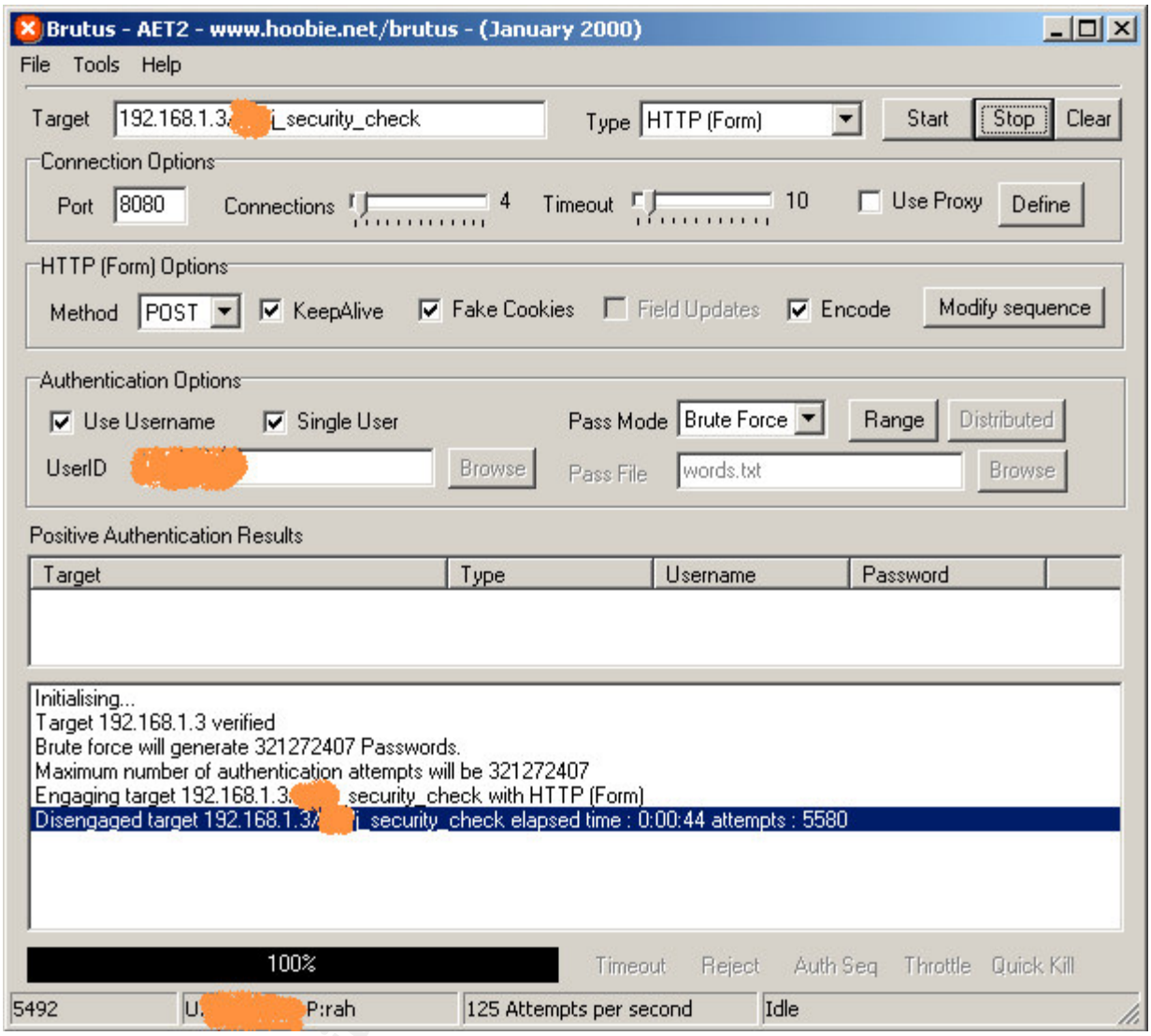

Figure 22. Configuration of Brutus for login to the application

If a system is compliant with Control C-8, then the maximum authentication rate will be limited to 1 per 3 seconds (which equals 14 attempts during the test period). The test recorded 5580 actual password guess, which is a sustained throughput rate of at least 125 per second or 375 authentication attempts per 3 seconds. This rate exceeds number of password guesses per 3-second interval limit set by the control (375 to 1). Therefore the system is not compliant with this control.

Test 11: Modification of files by hosted components

The purpose of this test is to determine whether application components hosted within the BES server (specifically the Partition) have write access to configuration files of the BES server.
As per the test instructions of Control C-11, an inspection is made of the partition's java_security.policy file.

The location of the java_security.policy used by a partition is specified by the server.security.policy property in the Partition's partition.properites file. Thus, the first step is to obtain this value for the application's Partition, 'standard'. This was achieved using a grep command, see Figure 23.

```
[audit@SPOCK T11]$ grep "server.security.policy" \
/opt/BES/var/servers/SPOCK/adm/properties/partitions/standard/partition.properties
server.security.policy=security/java_security.policy

[audit@SPOCK T11]$
```

Figure 23. Determining server security policy file location of a Partition

Having determined the location of the java_security.policy file being used by the Partition, the file was then viewed. This was done using the cat command as shown in Figure 24.

```
[audit@SPOCK T11]$ cat \
/opt/BES/var/servers/SPOCK/adm/properties/partitiions/standard/security/java_security.pol
icy
// java_security.policy
//
// Out of the box Java Security Manager Policy file
//
// It's highly likely that, in production, this file should be customized.
//

/*
 * Out of the box grant AllPermission.
 * This should be removed when tuning the server's security for production.
 * The other permissions in this file present some examples of possible permissions.
 */
grant {
  permission java.security.AllPermission;
};

(continued below)
```

Figure 24. Retrieving the java_security.policy for the Partition

Notice that the first grant statement grants AllPermission to all entities. Such broad permissions maybe useful for development-time purposes, but is not recommended for production systems - as the comment in the file makes mention of.

Since Java calculates the permissions granted to a code source as the union of all applicable permissions, the effect of having the grant statement of Figure 24 in the java_security.policy file is to grant all permissions to all hosted components loaded by the Partition. This fact alone indicates that the system is not compliant with control C-11. However, for completeness, a further examination of the policy file is made to determine the specific permissions granted to the hosted components of the Partition: the ears, ejb_jars, wars, and rars. These are shown in Figure 25.

```
(non relevant codeBase modules removed from listing)

/*
 * grants permissions for local J2EE Modules (in addition to these properties
 * local J2EE Modules are restricted from calling exitVM)
 */
grant codeBase "file:${server.instance.root}/partitions/standard/ears/-" {
  permission java.lang.RuntimePermission "accessClassInPackage.sun.net.www";
  permission java.lang.RuntimePermission "accessClassInPackage.sun.net.smtp";
  permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete, execute";
  permission java.io.SerializablePermission "enableSubclassImplementation";
  permission java.util.PropertyPermission "*", "read, write";
  permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
  permission java.lang.RuntimePermission "accessDeclaredMembers";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.lang.RuntimePermission "modifyThreadGroup";
```

```
  permission java.lang.RuntimePermission "readFileDescriptor";
  permission java.lang.RuntimePermission "writeFileDescriptor";
  permission java.lang.RuntimePermission "loadLibrary.*";
  permission java.lang.RuntimePermission "queuePrintJob";
  permission java.net.SocketPermission "*", "accept, connect, listen, resolve";
};
grant codeBase "file:${server.instance.root}/partitions/standard/ejb_jars/-" {
  permission java.lang.RuntimePermission "accessClassInPackage.sun.net.www";
  permission java.lang.RuntimePermission "accessClassInPackage.sun.net.smtp";
  permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete, execute";
  permission java.io.SerializablePermission "enableSubclassImplementation";
  permission java.util.PropertyPermission "*", "read, write";
  permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
  permission java.lang.RuntimePermission "accessDeclaredMembers";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.lang.RuntimePermission "modifyThreadGroup";
  permission java.lang.RuntimePermission "readFileDescriptor";
  permission java.lang.RuntimePermission "writeFileDescriptor";
  permission java.lang.RuntimePermission "loadLibrary.*";
  permission java.lang.RuntimePermission "queuePrintJob";
  permission java.net.SocketPermission "*", "accept, connect, listen, resolve";
};
grant codeBase "file:${server.instance.root}/partitions/standard/wars/-" {
  permission java.lang.RuntimePermission "accessClassInPackage.sun.net.www";
  permission java.lang.RuntimePermission "accessClassInPackage.sun.net.smtp";
  permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete, execute";
  permission java.io.SerializablePermission "enableSubclassImplementation";
  permission java.util.PropertyPermission "*", "read, write";
  permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
  permission java.lang.RuntimePermission "accessDeclaredMembers";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.lang.RuntimePermission "modifyThreadGroup";
  permission java.lang.RuntimePermission "readFileDescriptor";
  permission java.lang.RuntimePermission "writeFileDescriptor";
  permission java.lang.RuntimePermission "loadLibrary.*";
  permission java.lang.RuntimePermission "queuePrintJob";
  permission java.net.SocketPermission "*", "accept, connect, listen, resolve";
};
grant codeBase "file:${server.instance.root}/partitions/standard/rars/-" {
  permission java.lang.RuntimePermission "accessClassInPackage.sun.net.www";
  permission java.lang.RuntimePermission "accessClassInPackage.sun.net.smtp";
  permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete, execute";
  permission java.io.SerializablePermission "enableSubclassImplementation";
  permission java.util.PropertyPermission "*", "read, write";
  permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
  permission java.lang.RuntimePermission "accessDeclaredMembers";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.lang.RuntimePermission "modifyThreadGroup";
  permission java.lang.RuntimePermission "readFileDescriptor";
  permission java.lang.RuntimePermission "writeFileDescriptor";
  permission java.lang.RuntimePermission "loadLibrary.*";
  permission java.lang.RuntimePermission "queuePrintJob";
  permission java.net.SocketPermission "*", "accept, connect, listen, resolve";
};
```

Figure 25. The java_security.policy file (cont.)

The above entries follow the same pattern: granting application components write permissions to all files. Thus if any of these components attempted to write to a security file of BES the JVM would not prevent it.

In a compliant system the J2EE components would not be granted write access to the BES security configuration files. (In fact, a production J2EE server should not be granting any file permissions on any files to any hosted components.). Therefore, based on the above evidence, the system is not compliant with this control.

Notwithstanding non-compliance, the stimulus/response test of C-11 was also performed. This test involves loading an EJB (FileIOTestEJB) which attempts to copy a file using Java's file I/O operations.

This activity is initiated by an external client (FileIOTestEJBClient). In this test, the EJB was directed to modify the java_security.policy file. The commands used for this test and the results are shown in Figure 26.

```
[audit@SPOCK T11]$ vbj -VBJclasspath test11Client.jar -VBJprop vbroker.agent.port=19741 \
  com.camacit.MQRaudit.tc11.ejb.FileIOTestEJBClient \
  ../../adm/properties/partitions/standard/security/java_security.policy
-- Initializing bean access.
-- Succeeded initializing bean access through Home interface.
-- Execution time: 5032 ms.
-- Calling create()
-- Succeeded: create()
-- Execution time: 41 ms.
-- Return value from create():
Stub[repository_id=RMI:com.camacit.MQRaudit.tc11.ejb.FileIOTestEJB:00000 ...
Setting sourceFilename to ../../adm/security/java_security.policy
Executing test: EJB will duplicate the specified source file.
Test executed at Mon Feb 09 20:52:14 PST 2004
```

Figure 26. Executing the T11 test client

The test client indicates that its request to the EJB was successful. Therefore, the file which should have been generated by the EJB (original file with .audit suffix) is searched for, see Figure 27.

```
[audit@SPOCK T11]$ ls -l \
  /opt/BES/var/servers/SPOCK/adm/security/java_security.policy.audit
-rw-rw-r--    1 borland  borland       223 Feb  9 20:52
/opt/BES/var/servers/SPOCK/adm/security/java_security.policy.audit

[audit@SPOCK T11]$ head !$
head /opt/BES/var/servers/SPOCK/adm/security/java_security.policy.audit
// File duplicated from ../../adm/security/java_security.policy at Tue Feb 10 04:52:14
GMT 2004 by CamacIT's FileIO Audit Test (#11)
grant codeBase "file:${server.root}/lib/*" {
  permission java.security.AllPermission;
};

[audit@SPOCK T11]$
```

Figure 27. Inspecting the results of the FileIOTestEJB operation

The expected file - java_security.policy.audit - was found. So its contents were viewed. This shows that the contents were created when the EJB ran (Feb 09 20:52 PST / Feb 10 04:52 GMT).

In a compliant system, the EJB's request to create the duplicate file would have been denied by the JVM and would raise a security exception. Whereas in this test, the EJB was able to successfully create the file. Therefore, this system is not compliant with the Control C-11.

Test 13: Default user accounts and security configurations

The purpose of this test is to determine whether sample user accounts and default security configurations defined when BES is installed are still present in the system's configuration. Specifically, this test will determine whether knowledge about those settings can be exploited to gain unauthorized administration access to the BES server.

As per the test instructions of Control C-13, the following aspects of the configuration are checked.

1. a) The configuration file management_config.jaas is inspected to determine if the ServerRealm definition (a realm defined during installation) is still present in the System's security. The contents of this file are retrieved using the cat command, see Figure 28.

```
[audit@SPOCK T13]$ cat /opt/BES/var/servers/SPOCK/adm/security/management_config.jaas

ServerRealm {
  com.borland.security.provider.authn.LDAPLoginModule required
        providerurl="ldap://@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@:389"
        initialcontextfactory=com.sun.jndi.ldap.LdapCtxFactory
        searchbase="o=@@@@@@@@@@@@@@@@@"
        usernameattribute=uid
      query.1="ROLE=(&(objectClass=groupOfNames)(uniqueMember={0}))";
};

(NB. @@@ = obfuscation of reported evidence by auditor to maintain confidentiality)
```

Figure 28. JAAS configuration of System's management functions

This shows that the ServerRealm's definition has not been removed from the file, although it has been changed from its installation default.

b) Given that the ServerRealm definition is present, the next step is to determine if its URL property been altered from the default userdb.jds file value. Since the URL for the ServerRealm (and the entire ServerRealm definition) no longer reference the BasicLoginModule or userdb.jds, it is concluded that URL property has indeed been changed.

Since the ServerRealm definition has been replaced there is no need to proceed with part (c) of the first test.

The second test for Control C-13 is to determine whether the HostRealm definition has been removed from the management_config.jaas file. Inspecting the contents of the file (see Figure 28) shows that this definition has been removed from the file.

In a system compliant with control C-13, both tests 1 and 2 above will be affirmative. The audit tests demonstrated this to be the case. Therefore, the system is compliant with control C-13.

Notwithstanding compliance, the stimulus/response test of C-13 was also performed. This test attempts to login to BES's administrative realm using a sample user account. In this instance, the user account admin/admin was used which is one of the default accounts installed with the product. Following the instructions of the test:

1. Ensure that the IAS process of BES is running. Since the management port for this BES server is 19806, a check to see whether its IAS process is running is to query the OSAgent running for that domain. This is done with the following osfind command, see Figure 29.

```
[audit@SPOCK T13]$ osfind -VBJprop vbroker.agent.port=19806

osfind: Found one agent at port 19806
        HOST: SPOCK

osfind: There are no OADs running on in your domain.

osfind: There are no Object Implementations registered with OADs.

osfind: Following are the list of Implementations started manually.
        HOST: SPOCK

                REPOSITORY ID: IDL:inprise.com/util/ncs/FileManager:1.0
                        OBJECT NAME:

                REPOSITORY ID: IDL:borland.com/Enterprise/Server/Agent:1.0
                        OBJECT NAME: SPOCK

[audit@SPOCK T13]$
```

Figure 29. Determining whether the IAS process is running

These results show that two CORBA objects are present in that domain. These objects are hosted by the IAS process. Therefore, IAS is operational. Another test would have been to see whether the IAS process existed in the process table of the operating system. However that approach does not indicate which domain the IAS processes are operating in.

2. Invoke the iastool tool with the –ping option. This is done with a user ID that is installed with the product by default. See Figure 30.

```
[audit@SPOCK T13]$ iastool -ping -server SPOCK
Creating an ORB on port 19806...

Please enter security credentials to be used for server discovery
0. Abort
1. ServerRealm
Choose realm for authentication: 1
Username: admin
Password: admin
Unable to contact server SPOCK in specified management domain: 19806
This could be due to incorrect realm, user, and/or password.

[audit@SPOCK T13]$
```

Figure 30. Invoking operations on the BES server from an unauthorized account

The response from the utility is to reject the supplied credentials. Thus, this user account is no longer recognized by the BES server. Hence this default user account has been removed from the configuration.

In a system compliant with control C-13, the default user account credentials will be rejected. The stimulus/response audit test showed this to be the case. Therefore, the system is compliant with control C-13 according to this test also.

Test 15: Confidentiality of user credentials in transit

The purpose of this test is to determine whether credentials used to authenticate with BES's Administration realm are kept confidential while in transit between the client (i.e. the BES console) and BES (i.e. the IAS process).

As per the test instructions of Control C-15, the following stimulus/response test was performed.

1. Two alphanumeric strings were devised to serve as a username and password. These were auditor3a4c and auditor3a4c.

2. The client was run from the auditor's laptop in the DMZ. That laptop had ngrep[51] installed to monitor the network stack of the laptop. Ngrep was configured to search for the word auditor3a4c in the network traffic (i.e. communication stream). This is shown in Figure 31.

```
Auditor's host> ngrep -w auditor3a4c
interface: \Device\NPF_{8A1AA73B-04E9-41ED-A96E-2796DD2D75A4}
match: ((^auditor3a4c\W)|(\Wauditor3a4c$)|(\Wauditor3a4c\W))
```

Figure 31. Monitor communication stream for a particular word using ngrep

3. The BES Administration console was launched on the auditor's laptop. This prompted for login credentials and an authentication realm. See Figure 32. The credentials were entered and ServerRealm specified.

Figure 32. Login prompt for BES Administration

Clicking the Login button, brought back the following response. See Figure 33.

[51] Eyler. "ngrep"

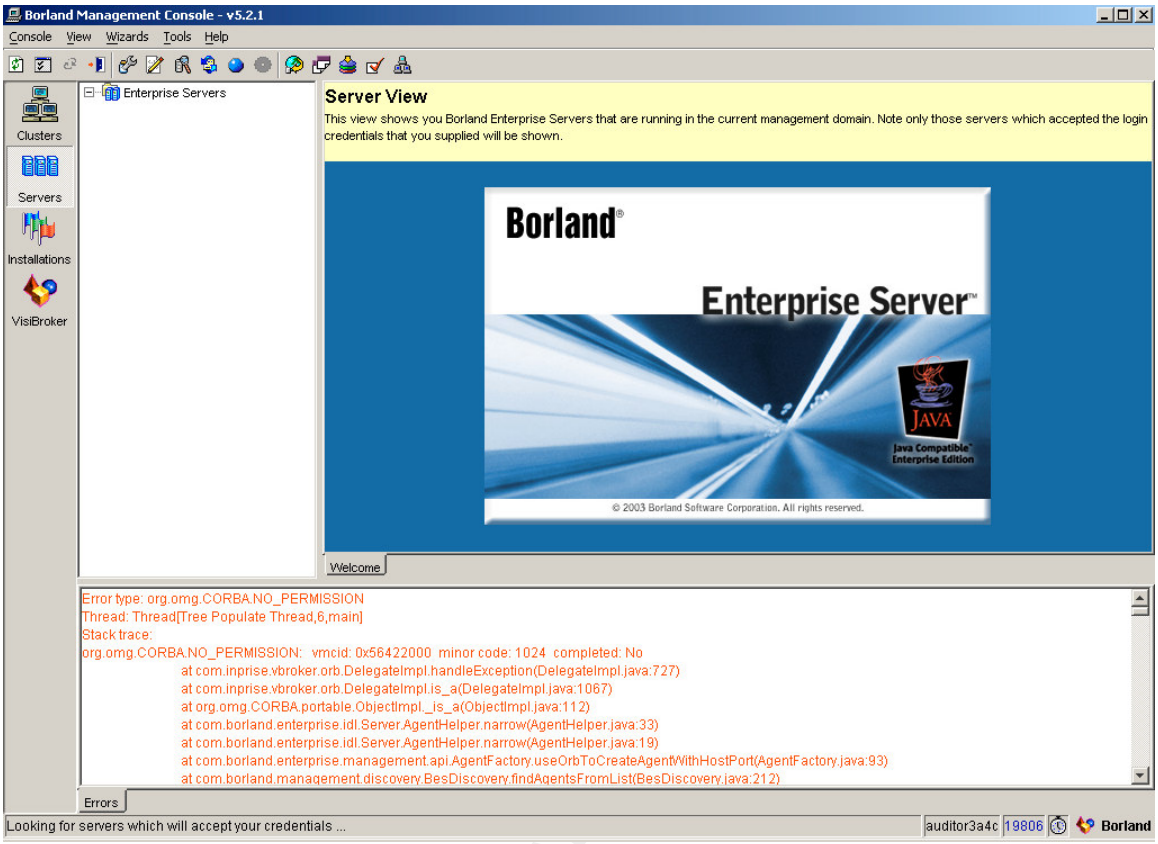

Figure 33. Authentication response to login request

This response demonstrated that credentials were submitted by the Console to BES, but were not accepted as valid. During this time, the following activity was reported by ngrep. See Figure 34.

```
##############################################################################
############################
```

Figure 34. Output from command "`ngrep -w auditor3a4c`"

4. The output from ngrep utility was inspected for the presence of these tokens. The output showed that the keyword “auditor3a4c” was not observed by ngrep in the communication stream.

In a system compliant with control C-15, authentication credentials will be kept confidential while in transit to the IAS process. The stimulus/response audit test demonstrated that credentials exchanged with BES are not observable by 3rd parties and therefore remain confidential between the sending and receiving parties. Therefore, the system is compliant with control C-13.

(Note 1. To confirm that some other issue did not generate the NO_PERMISSION exception, the system administrator logged in using credentials that are valid for the system. The result was a successful login and showed that the authentication process was functioning correctly. See Figure 35).

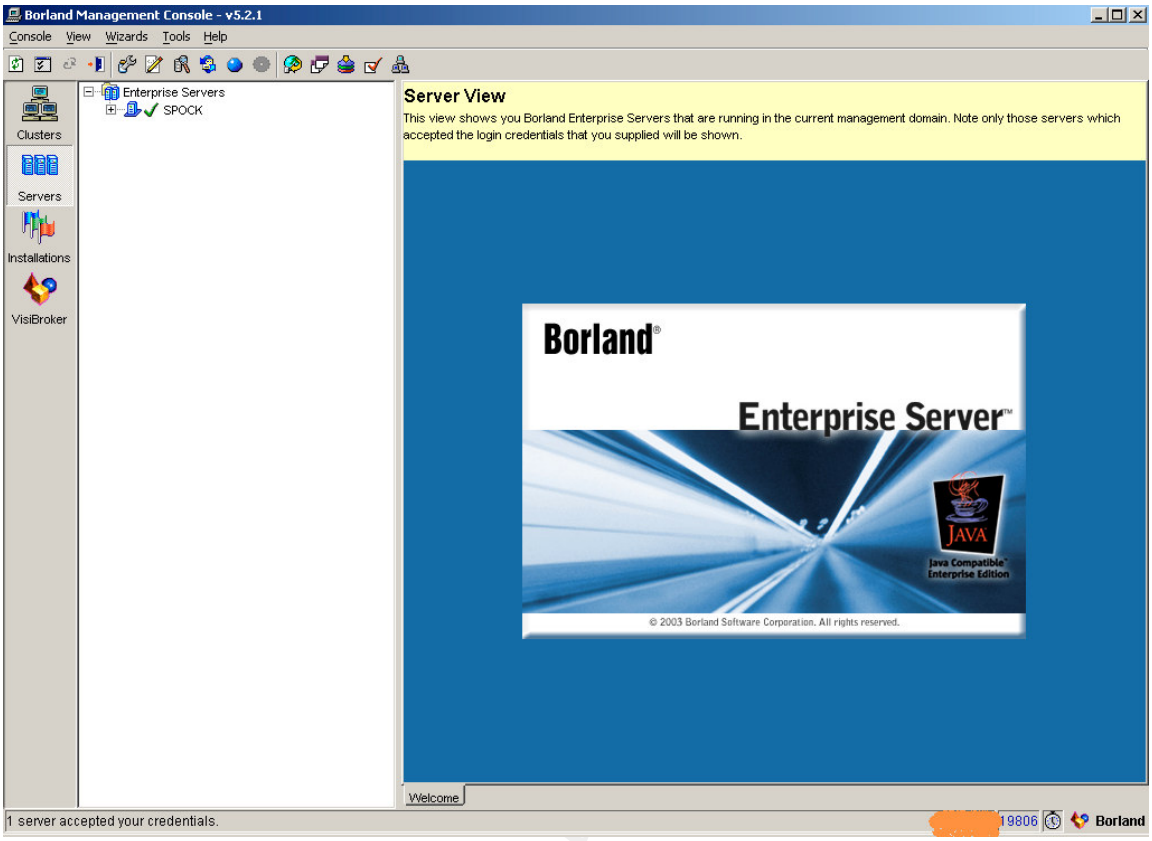

Figure 35. Successful login to BES Administration

(Note 2. To check that ngrep would have reported the keyword if it was sent across the network in plain text, the following test was performed. The keyword was sent to a remote host via the network – in this case, the Netscape HTML browser was used to submit the keyword to an Internet search engine - Google. This action caused the keyword to be encapsulated within an HTTP request and sent plain text across the network to a remote host: www.google.com / 216.239.57.99:80. The activity reported by ngrep during this period of time is shown in Figure 36.

```
T 192.168.1.109:1898 -> 216.239.57.99:80 [AP]
  GET /search?num=30&hl=en&lr=&ie=UTF-8&oe=UTF-8&q=auditor3a4c&btnG=Goog
  le+Search HTTP/1.1..Host: www.google.com..User-Agent: Mozilla/5.0 (Win
  dows; U; Windows NT 5.0; en-US; rv:1.4) Gecko/20030624 Netscape/7.1 (a
  x)..Accept: text/xml,application/xml,application/xhtml+xml,text/html;q
  =0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2
  ,*/*;q=0.1..Accept-Language: en-us,en;q=0.5..Accept-Encoding: gzip,def
  late..Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7..Keep-Alive: 300.
  .Connection: keep-alive..Referer: http://www.google.com/search?num=30&
  hl=en&lr=&ie=UTF-8&oe=UTF-8&q=auditor3a4c&btnG=Google+Search..Cookie:
  PREF=ID=014004ea1f056c93:LD=en:NR=30:TM=1068922151:LM=1070861968:S=Crq
  NeJIj6tIN0rfZ; AdSenseLocaleCookie=en....
```

Figure 36. ngrep output when keyword is sent clear-text

This output demonstrates that ngrep correctly detects the presence of the keyword when sent in plain text. Therefore, it is concluded that the test is functioning correctly and would have detected these tokens had they been exchanged between the BES console and IAS process in plain text.)

Test 17: Detecting changes to the system's configuration

The purpose of this test is to determine whether changes to the system's configuration, particularly its security configuration, are detected during the normal course of administration activities and whether these events are brought to the attention of administration staff in a timely manner.

As per the test instructions of Control C17, the following stimulus/response test was performed.

A non-destructive, non-corruptive change was made to the system configuration using the test EJB FileIOTestEJB (see test 11). This EJB was instructed to modify the file example_java_security.policy in the <BES>/var/servers/SPOCK/adm/security directory. If successful, the EJB will duplicate the file to a file named example_java_security.policy.audit and prepend a comment line to it's contents indicating when this activity was performed. This file is not used by the system, but is a sample file installed with it so as to limit the test's potential for corrupting the system's configuration. Furthermore, the action of duplicating a file rather than modifying an existing one is taken because the change detection system should not limit itself to identifying only existing files but should include files which might be added as well.

The EJB's test client was invoked with the following arguments to duplicate the example_java_security.policy file. See Figure 37.

```
[audit@SPOCK T17]$ vbj -VBJclasspath test11Client.jar -VBJprop vbroker.agent.port=19741 \
 com.camacit.MQRaudit.tc11.ejb.FileIOTest.EJBClient \
 ../../adm/security/example_java_security.policy
-- Initializing bean access.
-- Succeeded initializing bean access through Home interface.
-- Execution time: 3843 ms.
-- Calling create()
-- Succeeded: create()
-- Execution time: 38 ms.
-- Return value from create():
Stub[repository_id=RMI:com.camacit.MQRaudit.tc11.ejb.FileIOTestEJB:00000 ...
Setting sourceFilename to ../../adm/security/example_java_security.policy
Executing test: EJB will duplicate the specified source file.
Test executed at Tue Feb 10 18:19:48 PST 2004
```

Figure 37. Duplicating a system configuration file

The presence of the duplicated file was tested for using the "ls" command. See Figure 38. The first few lines of the new file were viewed using the 'head' command.

```
[audit@SPOCK T17]$ ls -l \
  /opt/BES/var/servers/SPOCK/adm/security/example_java_security.policy.audit
-rw-rw-r--    1 borland  borland      5835 Feb  10 18:19
/opt/BES/var/servers/SPOCK/adm/security/example_java_security.policy.audit

[audit@SPOCK T17]$ head !$
head /opt/BES/var/servers/SPOCK/adm/security/example_java_security.policy.audit
// File duplicated from ../../adm/security/example_java_security.policy Wed Feb 11
02:19:48 GMT 2004 by CamacIT's FileIO Audit Test (#11)
// @(#)jserv.policy    1.7 99/06/04

/*
 * defines the location of the keystore file
 */
keystore "file:${user.keystore}", "JKS";

/*
 * grants permissions for jar files in the lib directory

[audit@SPOCK T17]$
```

Figure 38. Inspecting the results of the FileIOTestEJB operation

The output of the head command and ls command show the new file was created as part of this test. During the stimulus phase of this test the administrator was not informed about the specific activities.

If the system is compliant with Control C-17 then the presence of the new file will be detected within 4 hours.

After 23 hours had elapsed (at Feb 11 17:26 PST), I inquired of the administrator if this new file had been detected. The system administrator reported it had not been detected and that no automated process was in place for this system to detect such changes. Since the change was not detected, the system is not compliant with Control C-17.

Test 19: Monitoring login attempts

The purpose of this test is to determine whether login attempts are monitored and recorded by the system.

As per the test instructions of Control C-19, the following stimulus/response test was performed.

First, attempts were made to login to the application through its web server interface. The first attempt was using invalid credentials and was performed on Tuesday Feb 10 at 18:47 PST. The login attempt was rejected by the application. The second attempt was using valid login credentials (entered by the administrator) and was performed on Tuesday Feb 10 at 18:50 PST. This attempt was accepted by the application.

The event log of the partition was examined from 18:47 PST onwards for login events. There was no entry made at 18:47; the most recent was made at 18:19. This indicates that the unsuccessful login attempt was not recorded. The login attempt made at 18:50 was, however, recorded in the log. See Figure 39.

```
[Tue Feb 10 18:50:26 PST 2004] 00000016,2/10/04 6:50 PM,192.168.001.003,00001896,VBJ-
Application,Thread-15,INFO,Caller [Subject:
[Tue Feb 10 18:50:26 PST 2004]       Principal: AAAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Public Credential: Privileges for
AAAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Private Credential: Password credentials for
AAAAAAAAAAAAA in target realm ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Private Credential:
GSSUP:GSSUP:ServerRealm:AAAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004] ] trying to access /RRRRRRRRRRR/UUUUUUUUUUU.jsp
[Tue Feb 10 18:50:26 PST 2004] 00000017,2/10/04 6:50 PM,192.168.001.003,00001896,VBJ-
Application,Thread-15,INFO,Resource /RRRRRRRRRRR/UUUUUUUUUUU.jsp requires caller to be in
role(s) [XXXXXXXXXX YYYYYYYYYY ZZZZZZZZZZ ]
[Tue Feb 10 18:50:26 PST 2004] 00000018,2/10/04 6:50 PM,192.168.001.003,00001896,VBJ-
Application,Thread-15,INFO,
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] :::::::::::::::::::::::::: ACCESS CONTROL DECISION
::::::::::::::::::::::::::
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] RESOURCE  : [/RRRRRRRRRRR/UUUUUUUUUUU.jsp]
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] SUBJECT   : [Subject:
[Tue Feb 10 18:50:26 PST 2004]       Principal: AAAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Public Credential: Privileges for
AAAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Private Credential: Password credentials for
AAAAAAAAAAAAA in target realm ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Private Credential:
GSSUP:GSSUP:ServerRealm:AAAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004] ]
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] DECISION : GRANTED
[Tue Feb 10 18:50:26 PST 2004]
```

```
[Tue Feb 10 18:50:26 PST 2004] *** ACCESS GRANTED because..
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] *** [Subject:
[Tue Feb 10 18:50:26 PST 2004]       Principal: AAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Public Credential: Privileges for
AAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Private Credential: Password credentials for
AAAAAAAAAAAA in target realm ServerRealm
[Tue Feb 10 18:50:26 PST 2004]       Private Credential:
GSSUP:GSSUP:ServerRealm:AAAAAAAAAAAA@ServerRealm
[Tue Feb 10 18:50:26 PST 2004] ] belongs to role [YYYYYYYYYY] because..
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] *** [Privileges for AAAAAAAAAAAA@ServerRealm] comprise a
set of permissions, which IMPLIES the required permission: RolePermission[YYYYYYYYYY]
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] *** [Privileges for AAAAAAAAAAAA@ServerRealm] comprise a
set of permissions, which DOES NOT IMPLY the required permission:
RolePermission[XXXXXXXXXX]
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] *** Roles required for accessing resource
[/RRRRRRRRRR/UUUUUUUUUU.jsp] are: [XXXXXXXXXX] [YYYYYYYYYY] [ZZZZZZZZZZ]
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004]
[Tue Feb 10 18:50:26 PST 2004] End of Access Control Decission
[Tue Feb 10 18:50:26 PST 2004]
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
[Tue Feb 10 18:50:26 PST 2004]
```

Figure 39. Event log of successful login attempt

Note. Some of the information from the event log has been purposefully obscured in this report in order to protect the confidentiality of certain key elements of the application's security configuration. The legend for those changes is:

AAAA = login user id
RRRR/UUUU = login URL
XXXX, YYYY, ZZZZ = authorization roles of the application

A system compliant with Control C-19 will have logged both successful and unsuccessful login attempts. This exercise demonstrates that the system does not log unsuccessful login attempts. Therefore, on this point the system is not compliant with control C-19.

The second element necessary for compliance with control C-19 requires that a procedure exist to review the logs at least every 24 hours to identify unauthorized attempts to gain access to the application. The system's administrator was asked if the logs are reviewed at least once in every 24-hour period and if such a procedure was defined. He responded that they were not and that no such procedure exists. (Note. Even if such a procedure did exist, the information recorded in the event log is inadequate for such analysis.)

Based on these facts, neither the system nor the organization is compliant with Control C-19.

Test 22: Redundancy of system operations skills

The purpose of this test is to determine whether the skills necessary to operate and maintain the system are exist with more than one person in the organization, and therefore whether the organization has redundancy of skills in this respect.

As per the test instructions of Control C-22, the following three assessments were made.

First, an inquiry was made to determine if there are at least two individuals within the organization that could fulfill the operations role of this system (even if only temporarily). This inquiry was made to the

system administrator and the project sponsor. Both persons indicated that there was no such other individual within the organization.
Since there was no other individual in the organization able to perform this role, the remaining two assessments of the organization according to the control's test instructions were not performed.

An organization compliant with Control C-22 will have two individuals within the organization who are able to successful restore the system according to documented procedures. Since this organization has only one such individual capable of this, it is not compliant with Control C-22.

2 Residual Risk Analysis

The risks which remain in the system after the above compliance verification activity are examined in this section.

The risks that remain in the system can be classified into two categories. First, there are risks which have not been addressed by the audit, eventhough they fall within the defined scope of audit. Second, there are risks that are addressed by the audit, and are shown to be exposed.

On the first subject, to understand why the system faces risks that were not addressed by the audit it is necessary to recall Part I of the audit – the risk assessment. That risk assessment explicitly focused only on risks which had the potential for greatest impact to the assets of the system for this organization. For this organization and system, confidentiality and integrity of the application data was far more important than application availability. By focusing only on high-risk threats the subsequent amount of threat analysis and control development was reduced. This allowed an initial audit to proceed more rapidly with determining the state of the system's security in regards to its greatest perceived risks. The alternative would have been to include all the risks from all categories during the threat analysis, many of which figure much less importantly in the system's security. Yet their inclusion with other threats and controls would significantly lengthen the auditing process and its delivery schedule. Thus, by prioritizing the risks to the system and choosing to audit a subset of the most serious of them, as was the case here, an initial assessment could be performed sooner, which allowed compensating action to secure the system to begin much earlier than would have been the case with a more comprehensive approach. In addition, since the framework used to categorize and prioritize the risks was documented (see Part I), follow-on audits that examine remaining risks can easily continue with risks that this initial audit did not address.

The second source of residual risks facing this system are vulnerabilities identified by the audit which are not protected from exploit in the system. These were identified as non-compliant controls in the audit evidence. Table 11 is a list of those vulnerabilities, together with a brief analysis of the cost to remedy the situation.

Control	System Compliant	Risk	Remedy	Effort estimate (days)
C1	No	High	Change operating system privileges on DAR/EAR files. Institute an operational procedure to ensure future deployments maintain this state.	1
C2	No	High	Restrict network access to JNDI naming service	2
C4	No	High	Restrict network access to RDBMS	2
C8	No	High	Extend functionality of JAAS LoginModule or LDAP login service to produce required behavior	3
C11	No	Moderate	Configure Java's security policy for the Partition	1
C17	No	Moderate	Employ a change detection tool, devise a procedure for automatic notification, and institute an operational procedure for regularly reviewing it	2
C19	No	Moderate	Extend LoginModule or LDAP server to record such events and devise a mechanisms for regular reporting	3
C22	No	High	Document restoration procedure and train an individual in those activities	3 + 5

Table 11. Non-compliant issues: risks, remedies and costs

Fortunately, in each case the vulnerability's exposure can be eliminated. The effort required to do so depends upon the solution adopted. This audit does not recommend solutions, buts does suggest remedies only so that some estimate of effort involved in reducing the residual risk to the system may be determined.

Since other solutions may also exist, which might require less effort to implement, actual effort could be less than what is estimated here.

Control-1 considered file access to DAR / EAR files from which database login credentials could be obtained. Test-1 showed the system was not in compliance with this control. If exploited this vulnerability would place the system at high risk. A remedy for this situation would be to change the operating system permissions for these particular files. Furthermore, to prevent a relapse of this exposure, an operational policy should also be developed which applies this restriction to future DAR/EAR files that are deployed. The effort required to devise, implement and re-test the control is estimated at 1 day for administrator familiar with the system.

Control-2 considered access to the JNDI naming service of the server, from which datasources could be retrieved and database login credentials extracted. Test-2 showed the system was not in compliance with this control. If exploited this vulnerability would place the system at high risk. A remedy for this situation would be to protect network access to the naming service port of the Partition by using tools such as SSH or a host-based firewall. The effort required to devise, implement and re-test the control is estimated at 2 days for an administrator familiar with the system and these technologies.

Control-4 considered access to the database by clients other than BES using credentials obtained (illegitimately) from BES. Test-4 showed the system was not in compliance with this control. If exploited this vulnerability would place the system at high risk. A remedy for this situation would be to protect network access to the database by using tools such as SSH or a host-based firewall. The effort required to devise, implement and re-test the control is estimated at 2 days for an administrator familiar with the system and these technologies.

Control-8 considered the threat of password discovery by enumeration and the system's ability to protect against this threat. Test-8 showed the system was not in compliance with this control. If exploited this vulnerability would place the system at high risk. A remedy for this situation would be to add the missing behavior (throttling access attempts) in the JAAS LoginModule loaded by BES or in the LDAP server. The effort required to devise, implement and re-test the control is estimated at 3 days for an administrator familiar with the system and these technologies.

Control-11 considered the threat of components hosted within the Partition modifying the system's configuration files and thereby altering the security configuration of the system. Test-11 showed the system was not in compliance with this control. If exploited this vulnerability would place the system at moderate risk since it does not provide direct access to the data, but does facilitate subsequent access. A remedy for this situation would be to configure the Java security policy for the Partition. The effort required to devise, implement and re-test the control is estimated at 1 day for an administrator familiar with the system and this part of Java technology.

Control-17 considered the threat of changes to the system's configuration going undetected. Test-17 showed the system was not in compliance with this control. If exploited this vulnerability would place the system at moderate risk since it does not provide direct access to the data, but does facilitate subsequent access. A remedy for this situation would be to employ a change detection tool such as Tripwire[52], configure some procedure for automatic notification, and institute an operational procedure for regularly reviewing the solution to ensure it is functioning correctly. The effort required to devise, implement and re-test the control is estimated at 3 days for an administrator familiar with the system and the technology.

Control-19 considered the threat of login attempts going unrecorded and unobserved. Test-19 showed that the system did log successful login attempts, but not unsuccessful ones. This vulnerability, if exploited, would place the system at moderate risk. A remedy for this would be to customize the authentication process to record such events (for example, by extending the JAAS LoginModule used, or by customizing the LDAP server), and generate summary reports at specified intervals. The effort required to devise,

[52] Tripwire. http://www.tripwire.org

implement and re-test the control is estimated at 3 days for an administrator familiar with the system and the technology.

Control-22 considered the threat of not having personnel available to perform essential system operations when needed. Test-22 showed that the system was not in compliance with this control. If realized, this vulnerability could have significant impact on the system and thus places it at high risk. A remedy for this situation is to first document the system restoration procedure (including its settings etc) and then train an individual in those tasks. The effort required is estimated at 3 days for the present system administrator and a further 5 days for the trainee.

3 The System's Auditability

The extent to which the system was auditable is examined in this section.

Primary factors affecting the system's auditability were the objectiveness of the compliance tests and the reliance upon the presence of auditable information.

Objective Tests
The objectiveness of the compliance tests used in auditing the system is important since it is on the results of compliance tests that an assessment of the system rests. Objective compliance tests allow the state of the system to be independently verified. Therefore, where possible, objective compliance tests were sought. However, not all situations will accommodate objective tests. Below are cases where subjective compliance tests were used and the reasons for them.

Most of the audit's technical controls (C1-C15) have objective measures and independently verifiable and repeatable procedures to determine compliance. The exception to this are Controls 8 and 14 (Protecting the Application, and Protecting the System against password discovery by enumeration, respectively). In these cases although the measure for compliance was an objective one (the number of authorization access attempts per second), the compliance level (3 per second) was a subjective value. Thus the compliance criteria can be accurately determined, but where to set the compliance level is a matter of opinion.

In addition to technical controls, the audit also included operational and management controls. These were concerned primarily with the people who interact with the system. The compliance tests of these controls were therefore concerned with the domain of personnel, procedures and guidelines rather than system configurations and parameters. Such a domain is typically more difficult to verify objectively. This was the case for Controls C-16, C-18, C-19, C-20 and C-21.

In the case of Controls C-16, C-18, C-19 and C-21, the test for compliance depends to some extent on the ability of the person performing the task – i.e. executing the operational procedures. Procedures written for people often contain ambiguities and often make various assumptions about the reader's familiarity with the system and underlying technology. Therefore testing the system for compliance will also indirectly measure the individual's ability to follow the procedures (or not), their skill at performing those instructions (if those instructions are vague or ambiguous), and their familiarity with the system. It could be argued that the person's capabilities should be considered part of the test, since ultimately the test is concerned with the outcome of the operation. However, if this viewpoint is adopted, then compliance is with respect to a particular individual filling a particular role, not the organization in general. Such issues need to be remembered when reporting the outcome of operational control tests.

In the case of Control-18 (ensuring that product defects are addressed promptly), the test contains subjectively set compliance levels (period of time a critical patch must be applied to the system) similar in nature to Controls 8 and 14. As in those cases, although the values are subjectively chosen, they are objectively measurable.

In the case of Control-20 (restricting physical access to backup media), one requirement is to confirm that all backups either exist in the secure storage area or have been properly disposed of. Although such a condition clearly must be validated for compliance, determining the existence and location of all backups made of the system to-date must rely on information supplied by the administrator/operator. In this particular environment, such information can not be independently verified.

Auditable Information

The other important factor affecting system auditability is the presence of auditable information. Many of the operational and management controls require that procedural activities be documented (i.e. be as independent of the operator as possible) and supporting systems be in place (e.g. Intrusion Detection Systems). When such procedures are not documented and the systems do not exist, then it is difficult to determine compliance, since there is inadequate information available. This was the case for Controls C-18, C-19, and C-22. The absence of this information meant that compliance information could not be obtained, so by default the system was considered non-compliant in these areas.

Part IV

The Audit Report

Table of Contents

Executive Summary

On February 9 and 10, 2004, an audit of the system was conducted to determine the amount of risk it faced from security threats.

The audit's scope was limited to consider risks posed to the system at the AppServer layer only. This is but one layer of many in the system; other layers include the network and operating system. Of all the layers comprising the system, the AppServer is the one with the least amount of security related resources and wisdom currently available; primarily due to its being the least mature technology in the system's infrastructure stack. Consequently, determining the risk facing the system at this layer is more difficult than at other layers and therefore the reason for considering the AppServer layer only during the audit.

The audit measured the risk facing the system by evaluating it against security controls (see Part III). These controls were developed during the preparatory phases of the audit (see Part II) and were based upon threats identified by a threat analysis also performed as part of the audit (see Part I). The threat analysis examined the organization's use of the system as a business tool from a business perspective and identified the system's primary assets to be the data it manages. The most serious threats to this asset are a breach of its confidentiality or integrity. Such threats have the potential to impact the business of the organization by causing operational losses, loss of reputation, loss of competitive standing in the marketplace, regulatory penalties and fiduciary liabilities.

For example, disclosure of the data to unauthorized individuals (a breach of confidentiality) could result in regulatory fines for failure to protect the privacy of the individuals described in the data; result in revocation of special business licenses issued by the government necessary for operating in this particular market; and the loss of confidence from the public and business partners caused by a damaged public image / reputation. Likewise, if data integrity was compromised this could result in immediate operational losses including lost revenue as the application is offline while the data's integrity is restored (scrubbed of unauthorized changes), plus the cost of performing data verification, which would be incurred by the organization.

Hence, costs to the organization resulting from breached security can be significant. The magnitude of the costs in some cases would be compartible with the revenue generated by the system for the business. Therefore, some of these threats could have catastrophic impact on the organization's business.

The Degree of Compliance
The level of compliance found during the audit is shown below. The system's compliance with these controls indicates the degree to which the system is exposed to security exploits and therefore at risk.

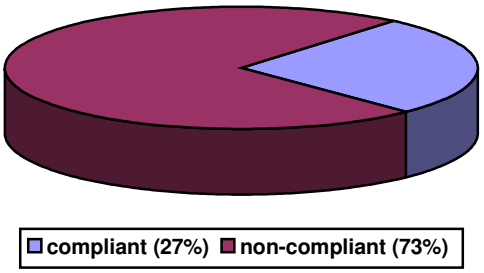

Figure 40. System's Compliance with Technical Controls (15 total)

Technical controls are concerned with the AppServer product and its configuration.

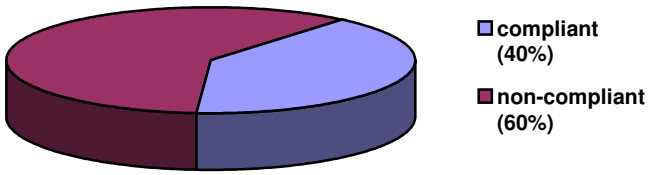

Figure 41. System's Compliance with Operational Controls (5 total)

Operational controls are concerned with the personnel, procedures and practices employed to support the AppServer in this environment.

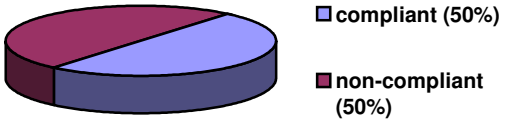

Figure 42. System's Compliance with Management Controls (5 total)

Management controls are concerned with the needs of the system being represented and supported at the management level of the organization.

The Cost to Remedy

The estimated cost to bring the system into compliance is divided into three disjoint categories: critical risks, non-critical risks, and on-going costs:

Category	Estimated Effort	Approximate Cost
Critical risks	11 days	$3,800
Non-critical risks	22 days	$7,800
On-going costs to maintain security	10 hours / week	$453 / week

Recommendations

It is recommended that critical risks be addressed immediately. These are instances where the system is currently exposed to threats, which if exploited could have significant impact on the assets of the system. The critical risks are identified by five technical controls and one operational control.

Once the critical risks have been resolved, non-critical risks should be addressed. These are instances where the system is also exposed to threats but the level of risk to the system is lower. The non-critical risks are identified by six technical controls, two operational controls and one management control.

To ensure the system remains secure, ongoing effort must be devoted to the security of the system. For greatest efficiency, that effort should be guided by organizational procedures, which will have been developed as part of the non-critical remedies. This ensures that the security of the system is maintained as it evolves over time in response to new requirements and new threats.

2 Audit Findings

The audit measured the system against 22 controls (defined in Part-II of this report). The system was compliant with 3 of those controls. The non-compliance against the remaining 19 controls demonstrates there are vulnerabilities in the system which are presently exposed and can be exploited to obtain unauthorized access to the application's data, thereby threatening an important business asset for this organization.

The findings from the audit are summarized below (See Table 12). The impact of the system's non-compliance is discussed in the next section.

Control #	*Purpose of Control*	*Compliant*
Technical Controls		
C-1	Restrict file access to deployed DAR/EAR files	No
C-2	Restrict access to JNDI service of Partition	No
C-3	Maintain confidentiality of database credentials while in transit	No
C-4	Restrict database connections to those which originate from BES only	No
C-5	Restrict database permissions granted to BES JDBC Connections	No
C-6	Maintain confidentiality of user credentials while in transit between browser and BES	No
C-7	Maintain confidentiality of user credentials while in transit between BES and LDAP	No
C-8	Protect application domain against password discovery by enumeration	No
C-9	Remove default user accounts which are installed with the product from the application domain	Yes
C-10	Protect BES configuration files from modification by external users (i.e. members of 'other' group)	Yes
C-11	Protect BES configuration files from modification by components hosted within Partition through Partition security services	No
C-12	Protect BES configuration files from modification by components hosted within Partition through Operating System security services	No
C-13	Remove default user accounts and security configurations from BES admin domain	Yes
C-14	Protect BES administration domain from password discovery by enumeration	No
C-15	Maintain confidentiality of user credentials while in transit between BES administration clients and BES.	Yes
Operational Controls		
C-16	Ensure system and data can be properly restored from a backup	Yes
C-17	Ensure changes to system's configuration are detectable and result in automatic notification	No
C-18	Ensure product defects are addressed promptly	No
C-19	Ensure login attempts (successful and unsuccessful) are monitored	No
C-20	Ensure physical access to backup media is restricted	Yes
Management Controls		
C-21	Ensure that responsibility for the system's security is assigned at the management level.	Yes
C-22	Ensure skills needed to perform essential system operations reside with more than one individual within the organization	No

Table 12. System's compliance against the Ruler

3 Risks facing the system

The risks identified are ones that can threaten the application's data confidentiality and integrity. The impact of this to the organization's business was addressed in the Executive Summary and also at length in the threat analysis of Part I.

The risks identified by the audit are methods that would lead such an impact on the application – the loss of the application's data confidentiality and integrity. What the audit showed was which of those methods are exploitable in the system. The rest of this section will review the non-compliant controls and summarize the particular threat that each posse to the security of the system.

3.1 Technical Controls

Controls C1-C5
Controls C1-C5 address threat T-1 that is concerned with database login credentials being obtained by a threat agent and used to establish a separate connection to the database.

The threat analysis of Part I identified several methods whereby the credentials could be extracted from the system directly (see threats T1.1 – T1.6) and based on these identified vulnerabilities, security controls C1-C5 were devised (see Part II).

The system was evaluated against these controls to determine whether it was exposing these vulnerabilities. The evidence collected (and documented in Part III) demonstrates that the system was not compliant with any of these 4 controls.

Non-compliance against controls C1-C3 demonstrates there are three vulnerabilities in the system that could be exploited to obtain credentials directly from the system. Non-compliance with C4 demonstrates there are be no restrictions that would prevent a new connection being established with the database using these credentials. And non-compliance with C5 demonstrates that a threat agent would have privileges to modify the schema within the database.

Controls C6-C9
Controls C6-C9 address threat T2 that is concerned with user login credentials to the application being obtained by a threat agent and used to gain unauthorized access to the application.

The threat analysis of Part I identified several methods whereby these user credentials could be obtained (see threats T2.1-T2.3) and based on these identified vulnerabilities, security controls C6-C9 were devised (see Part II).

The system was evaluated against these controls to determine whether it was exposing these vulnerabilities. The evidence collected (documented in Part III) demonstrates that the system was not compliant with 3 of the 4 controls.

Non-compliance against controls C6-C8 demonstrates there are three separate vulnerabilities that could be exploited to obtain user login credentials to the application. Compliance with C9 demonstrates that the threat agent would not be successful in using the default user accounts installed with the product to gain access to the application.

Controls C10-C15
Controls C10-C15 address threat T3 that is concerned with unauthorized access to the BES security configuration which a threat agent could then modify to defeat other technical security controls of the system and application.

The threat analysis of Part I identified several paths of access to the security configuration of BES (see threats T2.1-T2.3) and based on these identified vulnerabilities, security controls C10-C15 were devised (see Part II).

The system was evaluated against these controls to determine whether it was exposing these vulnerabilities. The evidence collected (documented in Part III) demonstrates that the system was not compliant with 4 of the 6 controls.

Non-compliance against controls C11-C12 demonstrates there are two separate access paths to the configuration that could be exploited. Compliance with C13 demonstrates that the threat agent would not be successful in using the default user accounts installed with the product to gain access to the application. Non-compliance with C14 demonstrates that the system is not well protected against password enumeration attacks on the administration domain, which could be used to obtain administrator privileges and thus access to the configuration. Compliance with C15 demonstrates that the system does protect the privacy of administration domain credentials in transit and therefore a threat agent would not be successful in attempting to obtain the credentials and thereby privilege and access in this way.

3.2 Operational Controls

Controls C16-C20
The preceding controls (C1-C15) are technical ones and concern threats which attempt to access the application's data – either directly or indirectly. Augmenting these controls are operational ones. Operational controls are concerned with ensuring the system's operations continue to be secure.

Five operational controls were devised for the system. The audit found that the system was compliant with only one of those – C16.

Compliance with C16 demonstrates that basic recovery of the system can be performed by the organization. This is an important operational capability to have since the system is presently exposed to many exploits that can compromise system's configuration integrity.

Non-compliance with C17-C19 demonstrates that there is no formal procedure in place for maintaining accepted secure operations of the system. Without such procedures, the security of the system is destined to decline over time due to neglect and inadequate maintenance.

3.3 Management Controls

Controls C21-C22
Management controls are used to determine whether personnel resources, budget and management visibility is adequate to meet the needs of the system's security. Without these, the system operations may not receive proper attention, which will result in the security of the system declining over time.

Two management controls were devised for the system. The audit found that the system was compliant with one of those – C21.

Compliance with C21 indicates that the needs of the security of the system are communicated to management. As mentioned, this is important for the ongoing security of the system.

Non-compliance with C22 indicates that the organization is placing the security of the system at risk because there is not enough redundancy in the personal assigned to maintaining it.

4 Recommendations

In response to the above vulnerabilities identified in the system by the audit, the following recommendations are made.

Obtain necessary resources
Applying the suggested remedies to the system will require effort from the administration staff to implement. Therefore, before embarking upon remedial activities, a commitment from management should first be obtained to allocate the necessary time and budget for implementing the remedies. An estimate of the amount of time and thus the anticipated cost is discussed in a later section (see Costs).

Address critical risks first
Risks which have the potential for significant impact on the application, have relatively simple exploits, and are exposed to a large threat agent population are ones which are more likely to be exploited and would have a significant impact on the system. Technical controls C1, C4, C5, C8, C11 and operational control C18 fall into this category of and thus should be considered for remedy ahead of other non-compliance controls.

Address remaining non-compliance
The remaining risks should be addressed once critical risks have been resolved. The remaining risks are addressed by technical controls C2, C3, C6, C7, C12, C14, operational controls C17 and C19 and management control C22.

Institute a security-oriented culture
Addressing aspects of non-compliance in the system should be viewed as the initial step that reduces the risk facing the system. However, it is not the only step. Once the system is brought into compliance with these controls, maintaining the system in a secure state requires further effort and a culture which values security. Such a culture will manifest itself in the following ways:

- Changes to the system in production are considered from a security perspective. Specifically, changes are considered to see if they introduce new vulnerabilities or weaken existing controls.

- The threat analysis and subsequent security controls developed in this audit will be updated when new system requirements are introduced and architecture changes are made. In this way the audit remains useful and its controls remain relevant as the system evolves.

- Further risk assessments maybe considered which continue beyond the scope of this initial audit. This audit's risk assessment purposely examined only threats that would have a major impact on the business. Other threats that have less of an impact should also be considered, even if they do not lead to formal audits. Such threat analysis should be undertaken even if just to confirm the belief that they do not pose a serious threat to the organization.

5 Costs to Remedy

The costs of securing the system can be separated into the cost of bringing the system into compliance and the ongoing cost of maintaining the security of the system.

The cost of bringing system into compliance will primarily be the effort required from the administration staff to implement remedies. Although it is not the purpose of this audit to recommend specific solutions, some estimate of the effort required to implement the solutions must be made. Therefore, the following are proposed as an estimate of the effort required. Some of these were discussed in more detail in Part III of the audit.

Control	Remedy	Effort estimate (days)
C1	Change operating system privileges on DAR/EAR files.	0.5
C2	Restrict network access to JNDI naming service	2
C3	Utilize a transport level encryption scheme, such as SSL or an SSH type solution	2
C4	Restrict network access to RDBMS	2
C5	Edit database permissions to remove schema manipulation privileges	1
C6	Require SSL for communication with the web server	1
C7	Utilize a transport level encryption scheme, such as SSL or an SSH type solution	1
C8	Extend functionality of JAAS LoginModule or LDAP login service to produce required behavior	3
C9	*(already in compliance)*	-
C10	*(already in compliance)*	-
C11	Configure Java's security policy for the Partition	1
C12	Run Partition under a different user ID than IAS, which does not have write access to these configuration files	1
C13	*(already in compliance)*	-
C14	Extend functionality of JAAS LoginModule or LDAP login service to produce required behavior. Effort for this is based on implementation available from C8.	0.5
C15	Utilize a transport level encryption scheme, such as SSL or an SSH type solution	1
C16	*(already in compliance)*	-
C17	Employ a change detection tool, devise a procedure for automatic notification, and institute an operational procedure for regularly reviewing it	3
C18	Develop an operating procedure to obtain latest patches and a procedure for rolling out patches into production.	3
C19	Extend LoginModule or LDAP server to record such events and devise a mechanisms for regular reporting	3
C20	*(already in compliance)*	-
C21	*(already in compliance)*	-
C22	Document restoration procedure and train an individual in those activities	3 + 5

Table 13. Non-compliant issues: risks, remedies and costs

Total estimated effort to address critical risks = 10.5 days.
Total estimated effort to address non-critical risks = 21.5 days

Based on the rate of $363 per day[53] for the administrator's time,

Total estimated cost to address critical risks = $3,812
Total estimated cost to address non-critical risks = $7,805

The cost of ongoing maintenance can only be approximated at this time since the organization has not yet developed specific operating procedures and therefore has not defined specific tasks and activities.

Nevertheless, it is estimated that such operational maintenance administrator would be in the order of at least 2-3 hours per week to monitor the security of the system and probably 1 day per week (on average) to

[53] The figure of $363 per day for the cost of an administrator's time is based on a 240-day work year and an annual salary of $87,238. This salary figure is the average reported salary for an administrator in the San Francisco/San Jose/Silicon Valley, CA Metro area for 2002 according to the SAGE/SANS/BigAdmin Annual Salary Survey (Kolstad, p40).

be spent on obtaining and applying patches and making other such necessary updates to the system in production. Hence,

Total estimated effort to support on-going security activities = 10 hours / week

Based on the rate of $363 per day for the administrator's time,

Total estimated cost to support on-going security activities = $453 / week

6 Compensating Controls

The approach described above used controls that were designed to directly and comprehensively address identified threats. Most of these were preventative controls although other weaker types such as detective and deterrent controls were used when stronger forms were not possible.

Although preventative controls offer the greatest security for the system, their costs to implement are often higher than other types of controls. When the implementation cost of a control is an issue, alternative weaker controls maybe considered. Weaker controls often have a lower implementation cost but they tend also to provide less security for the system. The amount of security offered (and whether it is adequate) should be more important than cost since there is no advantage in reducing implementation costs if it causes the security to fall below an acceptable level. In that case, the entire cost of implementing a weaker control is wasted since it does not adequately secure the system.

Some alternative compensating controls are suggested below, primarily as ones that might be cheaper for the organization to implement.

Coarse-grain security

As noted in Part I of this audit, the audit's scope was limited to consider only the Application Server element of the system. However that is only one layer of many in the system and one system of many in the organization's IT environment.

This inter-relatedness can be used to advantage; security vulnerabilities at one layer can be protected by controls of another layer. For example, vulnerabilities identified at the AppServer layer might be protected by controls implemented at the operating system or network layer.

Such compensating controls will typically be coarse-grain in their coverage since they are implemented at a different level of abstraction than the vulnerability.

When the skills, knowledge and technical resources to implement controls at one layer are significantly better and cheaper than at another, employing them to effect protection for another different layer can provide a cost effective approach to implementing security. In the case of this system, that is certainly true. Part I of this audit (section - Current State of Practice) noted the scarcity of resources available for securing this AppServer and J2EE AppServers in general. This can be compared with the abundance and maturity of resources available for the operating system and network levels. Also, for this particular organization, the administration staff is much more familiar and experienced with the operating system and network levels of the system than at the AppServer.
Against these advantages of using more host-based and network-based security mechanisms is the disadvantage that they are more coarse-grain solutions and therefore maybe less effective.

These issues are well illustrated by two controls that were proposed, C-11 and C-12 which both seek to restrict a Partition process's ability to modify configuration files of the AppServer. C-11 achieves this goal by using the JAAS security service of the AppServer. This is a fine-grained approach implemented at the AppServer level. It requires knowledge of JAAS security policies, a familiarity with the application loaded, and knowledge of the AppServer's footprint directory structure. Thus, although this control is

highly effective it requires considerable specialist knowledge to implement. By contrast, C-12 uses operating system file access permissions to achieve the same goal. Implementing this approach only requires an understanding of operating system security mechanisms (users, groups, and file permissions). However, if the Partition process ever needed to modify the security configuration files then C-12 would be too coarse-grain: it would deny legitimate as well as illegitimate access. In that case, C-11 would need to be relied upon.

Candidates for these types of compensating controls include:

- Use network layer controls to restrict the source of web requests to the AppServer to mitigate the problem of password discovery by enumeration. Since the web application would not be exposed to the entire Internet but only to those hosts which are known to be legitimate users, this control will reduce the likelihood of this vulnerability being exploited. This could be achieved using VPN technology or by adding source-IP rules to the firewall. However, this is a coarse-grain approach because it precludes the user from using a browser on any host and doesn't address the case where the threat agent is launching the attack from a legitimate user's host.

- Use the services of the LDAP server to limit the authentication-attempt rate rather than implement this throttle at the AppServer. This approach could be cheaper to implement since the implementation layer is LDAP rather than the AppServer, but again provides a coarser-grain solution. The solution is coarser because it impacts other systems that use that common LDAP server for authentication services.

Time-based controls
Another type of compensating control for the system is to limit its availability: or more specifically, taking the system offline outside of business hours.

This particular system is used by the organization only during business hours. Therefore, making the system available outside of business hours provides no benefit to the organization but does provide opportunity to threat agents. By making the system unavailable outside its hours of use will reduce the opportunity afforded to threat agents.

This control can be implemented with comparatively little cost (one of the goals for compensating controls), by denying or restricting access to the system rather than necessarily shutting down the system.

This control by itself is obviously not a substitute for the other controls recommended by the audit. However, it can contribute to the security of the system, particular in protecting against exploits which require substantial uptime from the system such as the password enumeration attack (see threat T2.3).

Obfuscating the Application Data
Yet another compensating control is to obfuscate key parts of the application's data within the database.

Most of the non-compensating controls identified above aim to prevent access to the application's data. Clearly such control types are to be preferred. However, because there are so many independent paths through which access to the data can be obtained, it is reasonable to consider what controls may reduce the impact of that event should it occur. Since it is not access to the database, per se, that poses a risk to the system but access to the data then there is the opportunity to place a control beyond the point of database access. One such control would be to obfuscate key elements of the application's data so that even if access to the database is obtained and even if the data could be extracted then it would not be intelligible or useful to the threat agent.

Candidate data for obfuscation would include data that could identify an individual, such as name, address, and social security number. Such occurrences in the data could be replaced with a hashed value, or a key that is mapped to actual data kept by another (more secure) system. Other candidates for obfuscation

include schema entries such as table names and column names. These items provide the important context needed to interpret the data. By replacing these labels with more cryptic entities (which would require corresponding changes be made within the application), the data's confidentiality maybe retained even if it were to be accessed by a threat agent.

A Multi-layer security defense

Reliance should not be placed on the AppServer's security controls alone. As was shown by the audit, the perimeter of this system is often thin and requires only one or two vulnerabilities to be exposed for a threat agent to gain access to the application's data and/or access to the administration domain of the system.

It is therefore prudent that the outer layers be used to form a multi-tier defense around the AppServer layer. The aim of such an approach is to require a threat agent traverse several layers before reaching the assets of the system.

Traversing multiple layers would then involve a combination of different exploits and different techniques and hence require the threat agent to possess diverse knowledge and tools. This raises the requirements placed on the threat agent and thereby reduces the likelihood of it occurring, and consequently the risk facing the system. Also, having to transcend multiple layers will allow the administration staff more opportunity to detect the presence of a threat and allow more time to take countermeasures.

Many of the compensating controls mentioned above could be applied in conjunction with existing controls to form such a multi-tiered defense architecture. Furthermore, it is recommended that the security architecture developed for this system's host and underlying network be reviewed with a view to consider the specific needs of this system and how controls at that level can provide additional protection to the system.

7 Summary

This section (Part IV) is the culmination of the audit. It reported the findings of the audit. Specifically, the degree of compliance found in the system to a set of security controls specifically developed for this system. It also recommended remedies that would bring the system into compliance, thus reduce the level of risk it faces. The cost to implement the recommendations was estimated and documented. In addition, alternative remedies which may been less costly to implement have also been discussed.

This audit has thus provided the organization a more detailed and objective view of the risk facing the system at the AppServer level as well as the costs required to reduce the exposure of the system and maintain its ongoing security.

List of References

1. Day, Kevin. Inside the Security Mind: Making the Tough Decisions. Upper Saddle River: Prentice Hall, 2003. 89-92.

2. Taylor, A., Buege, B., Layman, R. Hacking Exposed: Java & J2EE New York: McGraw-Hill, 2002.

3. Allamaraju, S., Buest, C., Davies, J., Jewell, T., Johnson, R., Longshaw, A., Nagappan, R., Sarang, P.G., Toussaint, A., Tyagi, S., Watson, G., Wilcox, M., Williamson, A., O'Connor, D. Professional Java Server Programming, J2EE 1.3 Edition Birmingham: Wrox Press Ltd., 2001.

4. Oaks, Scott. Java Security: Writing and Deploying Secure Applications 2nd Edition. Sebastopol: O'Reilly & Associates, Inc., 2001.

5. Jaworski, J., Perrone, P.J. Java Security Handbook Indianapolis: Sams Publishing, 2000.

6. King, C.M., Dalton, C.E., Osmanoglu, T.E. Security Architecture: Design, Deployment and Operations. Berkeley: RSA Press / McGraw-Hill, 2001.

7. Harris, Shon. CISSP Certification Exam Guide 2nd Ed. Emeryville: McGraw-Hill, 2003.

8. Sun Microsystems. "The J2EE FAQ". March 22, 2004. URL: http://java.sun.com/j2ee/faq.html (March 29, 2004).

9. Borland Software Corp. "The Borland Enterprise Server". Product Documentation for Version 5. March 29, 2004. URL: http://info.borland.com/techpubs/bes/index_v5.html (March 29, 2004).

10. Chouanard, Jean. "YASSP: Yet Another Solaris Security Package" How to install Solaris and have a good host security. November 19, 2000. URL: http://www.yassp.org/ (March 29, 2004).

11. Powell, B. M., Farmer, D., Archibald, M. "Titan Security Toolkit" A freely available host-based security tool that can be used to improve or audit the security of a UNIX system. March 25, 2004. URL: http://www.fish.com/titan (March 29, 2004).

12. Kimmelman, Jeff. "Risk Assessment and Management" A Presentation given to the New England Chapter of the ISSA (Information Systems Security Association) April 18, 2002. URL: http://www.issa-ne.org/documents/IT_Risk_Assessment_Methodology%20.pdf (March 29, 2004).

13. Borland Software Corp. "Borland Enterprise Server Security Service FAQ" March 29, 2004. URL: http://info.borland.com/devsupport/bes/faq/5.2/security/SecurityFAQw52.html (March 29, 2004).

14. Borland Software Corp. "Securing Web Access in BES5.2" Securing BES - Part II. March 29, 2004. URL: http://info.borland.com/devsupport/bes/faq/5.2/security/WEB_Security.html (March 29, 2004).

15. Naidu, Krishni. "Web Application Checklist" February 24, 2003. URL: www.sans.org/score/checklists/WebApplicationChecklist.pdf (March 29, 2004).

16. BEA Systems. "Securing a Production Environment" Product Documentation For Weblogic Server 8.1 March 29, 2004. URL: http://e-docs.bea.com/wls/docs81/lockdown/practices.html (March 29, 2004).

17. IT Governance Institute. "COBIT Control Objectives" 3rd ed. July 2000. URL: http://www.isaca.org (March 29, 2004).

18. Sun Microsystems. "Looking up an Object" The JNDI Tutorial. November 1, 2002. URL: http://java.sun.com/products/jndi/tutorial/basics/naming/lookup.html (March 29, 2004).

19. Russel, Christopher R. "Penetration Testing with dsniff" SANS Reading Room. February 18, 2001. URL: http://rr.sans.org/threats/dsniff.php (March 29, 2004).

20. Eyler, Pat. "Using ngrep" Linux Gazette September 2000. URL: http://www.linuxgazette.com/issue57/eyler2.html

21. McDonald, Stuart. "SQL Injection: Modes of attack, defense, and why it matters" SANS Reading Room. April 8, 2002. URL: http://www.sans.org/rr/papers/index.php?id=23 (March 29, 2004).

22. Brutus Web Site. URL: http://www.hoobie.net/brutus (March 29, 2004).

23. McAuliffe, Wendy. "Computer Passwords Reveal Worker's Secrets" ZDNet (UK), June 28, 2001. URL: http://www.zdnet.com/zdnn/stories/news/0,4586,2781327,00.html (March 29, 2004).

24. Kolstad, Rob. "SAGE/SANS/BigAdmin Annual Salary Survey" Results for Calendar Year 2002. URL: http://www.sun.com/bigadmin/content/salsurvey/salsur2002final.pdf (March 29, 2004).

25. Song, Dug. "dsniff" Web Site. URL: http://naughty.monkey.org/~dugsong/dsniff (March 29, 2004).