Global Information Assurance Certification Paper

Building a Secure Testing Environment

Kent Bohanon
GIAC Security Essentials Certification
Version 1.4b, Option 2
August 2003

1 Abstract

This case study will outline the steps taken to develop the security for a Windows 2000 based testing environment. For the purpose of this case study, the term *environment* is defined as the servers, test ids, and other infrastructure connectivity required for application testing. The focus will be on the methods and actions I utilized to define and implement system security and controls for a Windows 2000 combined testing environment. This information includes following logical, repeatable steps that resulted in thorough documentation about security policy and implementation of the system security. As will be explained, the initial security objective in the effort was to complete a risk assessment. I will also discuss how I identified security needs and possible solutions at the infrastructure, system, and operational levels. Furthermore, I will communicate how I worked as part of a team to support and evolve the test environment system, including addressing any user concerns or optimizations I realized. This case study assumes a reasonable knowledge about the practices of a mid-sized IT development and testing firm.

2 Understanding the Situation

The concept of a system usually stems from one of two main motives, either to solve a problem or to take advantage of a situation. Recently, I worked to provide security services for the development of a testing environment. This environment was conceptualized on the basis of both of the previously mentioned motives. The goal of the project was to minimize the duplicated support efforts and inconsistencies created by maintaining separate testing environments for six different projects. The goal was to be accomplished by taking advantage of the change window for converting the applications from Windows NT to Windows 2000. This effort was built on the concept of obtaining a subject matter expert from each of the key areas of testing, including security. This team would setup and maintain a combined test environment.

All of the applications had extreme dependencies upon each other, and ultimately converged to form what appeared to be a single, seamless application. As can be surmised by their existence, each of the applications had previously been developed and tested on the Windows NT platform. The true cause for the development of the combined test environment was the surplus of problems encountered in the initial, independent development of these projects on Windows NT.

2.1 A Top-Down View

During the course of the NT development for the applications, each project coordinated through and answered to the Project Management Office (PMO). The PMO enforced the proper testing of each application and the separation of each testing phase. The three initial levels of testing defined were based on the standard system development lifecycle model (SDLC) and included (in order

completed): unit testing, integration testing, and system testing. During the unit and integration levels of testing, each project had a separate environment in which to test. System testing began when all projects were relatively stable independently. The code would then be combined in a completely new environment where interdependency of the complete system could be tested.

The PMO tracked all of the delays and defects each project encountered during these three testing phases, along with project analyst feedback. Before authorizing new projects to convert the application code to the Windows 2000 platform, the PMO investigated the documented information about delays and defects related to the initial Windows NT development and testing environments. Upon reviewing, the PMO realized that each of the six projects spent a considerable amount of time stabilizing and properly configuring their testing environment instead of actually testing. This was unacceptable because projects were being expending time on environment setup when it was not their responsibility. Great effort was also spent trying to stabilize the environment and define the combined test cases once all the projects converged for system testing. The PMO had scratched the surface of a large problem, and one that needed a solution. As a result, a new effort was initiated.

This effort would establish a new testing environment that could support the six projects through system testing. In essence this decision would mean the creation of six Windows 2000 server environments, a sharp decline from the fourteen different environments utilized to test the projects on the NT platform. The PMO viewed the Windows NT to Windows 2000 platform migration as an opportunity to change environment coordination methodologies. However, the decision to implement changes during the Windows platform switch created a problem of its own. The timeline for taking advantage of the platform switch meant that the combined test environment would have to be researched and implemented in a matter of months. This meant addressing several security related problems in the delays, defects, and analyst feedback. The three most prominent issues needing to be addressed before the new environment was developed involved the setup of infrastructure connectivity, analyst access, and valid test cases.

2.1.1 Environment Infrastructure Setup

One of the first security related issues identified by the PMO was that projects struggled to properly configure the testing servers to interact with other infrastructure elements. Two infrastructure pieces identified as causing critical delays were the host data systems and the directory databases used in test cases. Although the teams responsible for supporting these two components had been identified, testing coordinators for the projects had problems understanding the systems' security and test case dependencies. Because of their lack of understanding, the test coordinators did not convey the proper information to the support teams. This resulted in the projects' environments not

being configured correctly, creating a lack of availability for the test environments. Though a key issue, this was not the only problem.

2.1.2 Inconsistent Access Setup

Project analysts had constantly complained about the inconsistent server accesses granted to the developers of different projects by the project security analysts. The problem existed because there was no company policy on the standard project analyst access setup. Each security analysts setup security in a different fashion, resulting in varied access levels. Some project analysts were administrators on a server, while others simply had access to remotely connect to a server. Still, others were only allowed to change the content of an application directory. This situation created a loss of confidentiality for the systems. In certain instances, project analysts were granted far more control than the situation warranted.

2.1.3 Lack in Understanding of the Test Cases

Beyond the access issues were problems with test id validity. Each of the six projects developed and tested a web-based application. Building the test cases for each of these applications first required identifying all the possible user types. The project was then responsible for coordinating the building of each the test actors, or test ids, on three different platforms: domain, directory, and host. It was crucial for each of these test ids to sync perfectly across all three platforms, just as they would in production. The projects then based their test cases on how user interaction with the application should look and operate as a result of the attributes of the users. It quickly became apparent that the project analysts lacked the expertise to properly setup the test ids to reflect test cases. This was a battle for projects throughout the NT based testing. Ultimately, this lead to degradation in the integrity of the application testing and numerous small problems as projects moved into further levels of testing and production.

3 Securing the Situation

In the months before the six interdependent applications were to begin the Windows 2000 conversion process, the project management office took action. The PMO initiated the envisioned effort by bringing together five subject matter experts from five different areas of testing, including me as the security consultant. The directions were clear, develop an environment that would solve the core problems experienced during the NT environment testing. I began my approach by trying to understand the risks inherent to the situation, while my other team members focused on their areas of expertise: server management, application development, project coordination, and test environment implementation.

3.1 Establishing Security Checkpoints

To understand the entire view of the situation, I decided to a complete a risk assessment for the proposed implementation. Though the PMO had defined

core, prominent risks, there were still unidentified security risks and issues. I chose this initial step because a risk management process should begin with a swift assessment of the risks. From this initial assessment, the bulk of the topics would be identified and documented.

3.1.1 The Risk Analysis

After researching the different tools available, I chose a standardized risk assessment my company had brought in-house. The methodology had several benefits, such as user friendliness. This type of pre-defined methodology had been developed to assist those without expertise in the risk assessment arena. After further research, I realized that the risk assessment would not only help me to identify the threats and vulnerabilities of the developing combined test environment, but it would also help me to correlate possible risks to actual impacts on business.

The first step of the risk assessment process was to complete a business impact assessment. The goal this portion was to rate how the general risks of the system could impact the business. The risks were classified into five main categories: confidentiality, integrity, availability, financial/operational, and compliance. The standardized risk assessment also called for the business managers of the system, including the PMO, to be involved in the analysis. I called a four-hour meeting, consisting of my teammates and representatives from the PMO. This portion of the assessment allowed me to merge the business partners with the technological problem solvers. I could ensure that the business stakeholders both gave input to the assessment and understood the gravity of security risks in terms of business impacts. The business managers' input to, and eventual acceptance of, the completed business impact assessment would be a great asset when tough "cost versus security" decisions would have to be made later.

The next step of the risk analysis methodology required an assessment of the proposed system's threats, vulnerabilities, and controls. I again setup a four-hour meeting involving my teammates and the key stakeholders of the PMO. This is where the actual risks were documented and rated on a scale of probability. The questions for this portion were based on the same categories as the business impact assessment. The difference was that with this section I was able to get a more complete view of the specific security risks and issues that could occur in the new system. It also required rating the probability of each risk occurring and documenting possible actions that could be taken to mitigate the risks and correct the issue. The results of this portion of the risk assessment methodology were far more informative than the few problem descriptions first compiled by the PMO.

With the risk assessment completed, I assembled a list of the risks identified. Employing the conversations and documentation from the two meetings, I noted both the business impact rating and probability of each risk, on separate one-to-

five scales. I then multiplied each of the referencing numbers (impact rating x probability of occurrence) to create an overall rating on a scale of one to twenty-five. Gathering this information allowed me to rank each risk in order of importance, resulting in a focused list of tasks. As with most implementations, setting direction would be key to ensuring priority issues were solved. From this list, an action plan was outlined to provide steps on how move forward. The most pertinent risks were separated into one of three security and control categories: infrastructure, system, or operational. With the risks categorized, each category was examined to determine if policies and mitigation plans could be coordinated between parallel risks.

At this point, it is important to note the effort expended to document each step taken to research and implement security for this effort. I meticulously documented all of the pertinent information gathered and the steps take to implement security solutions. The documentation would be vital to the security policy being created. It would both create awareness and easily repeatable steps to restore security if complications occured in the future.

3.2 Infrastructure Security and Controls

With the risk identified and classified, I then began investigating each classification. The first category of concern was the infrastructure security. Researching this category first would be beneficial to understanding how the servers would interact with the environment. This type of information must be understood before once can effectively examine system or operational security. As a result, the environment in which the system was being deployed was investigated to ensure the appropriate security was implemented to allow for proper end-to-end testing. Priority risks identified in this category included connectivity between the environment servers and remote systems. I met with my teammates responsible for project coordination and environment implementation. We discussed what environment elements would need to interact with the combined test environment servers. Three key elements were identified: host or mainframe data, directory data, and workstations.

To better understand the usage of workstations, I contacted my teammate responsible for addressing application development concerns. This teammate informed me that workstations were usually used to carry out test cases and only the ids used for the test cases would need access the workstations applications on the servers. Already knowing that test ids were based on host and directory platform information, I realized the inherent connections between the three infrastructure elements. After working with my team to review the NT testing documentation of the six Windows NT based projects, we defined three sets of host data necessary for the test cases of all six Windows 2000 projects. This was a huge leap in consolidating efforts. From this point, the data sets were created and the mainframe support team was contacted for instructions on how to connect to the proper systems. This would be key for the availability of the end-to-end environment. With this information in hand, I created an executable

that would install the proper settings on a server. Although the changes could have easily been installed by hand, the process was automated to ensure it was done correctly every time. This effort ensured the integrity of the testing data and the test cases.

The next risk in this category dealt with connectivity between the testing servers and the applications' directory needs. I researched the application development and testing documentation for the six original NT applications. I found the two key pieces of directory connectivity that the applications would need for authentication measures. The setup of this infrastructure connectivity was required to ensure the confidentiality measures of the applications would be properly tested. After tracking down how to setup the directory connectivity on the servers, I compiled the setup methods into an installation issuance.

Now came what would be the most painstaking process of the environment setup. To provide better integrity in the test cases, my team decided to setup and manage the test ids needed. This choice was made to coordinate the id knowledge and responsibility, which in turn would reduce duplicated efforts and errors. I facilitated a session with two key PMO members and my team. The meeting produced a compiled list of testing actors that would be sufficient to test all of the applications. I began the process of modeling each of the test ids to look exactly as they would in production. If other risks were properly mitigated, the test ids would be the defining factor for the integrity of the application testing. To track progress, I meticulously documented each test id in a complex matrix of attributes and application accesses. With the test id creation effort underway, I focused on a larger effort.

3.3 System Security and Controls

By far, the system security and controls category contained the most risks. The goal of this category was to identify the points of control that would allow the system to function as intended, to maintain system integrity, and prevent unauthorized access. The priority risks at hand in this category were developer access and the Windows 2000 Active Directory (AD) structure. The PMO had encountered very little testing with the new domain structure; many standards for testing in the Windows 2000 environment were still being developed.

In a planning session, my team discussed what the appropriate level of access for developers should be in each level of test. We also discussed other access that would need to be built into the environment; this discussion was based upon topics identified in the risk analysis. Coming out of the meeting we had defined the access as shown in *Figure 1*. The support team decided that in order to

Group	Test Level	Access on Server
Project Developers	Unit	Developer access (as defined)
Project Developers	Integration, System	No special access granted
Support Team	Unit, Integration, System	Administrators
Temporary Troubleshooters	Unit, Integration, System	Read access

Figure 1: Initial access matrix for testing.

accomplish application development, a project analyst would require certain accesses in the unit testing level. This list of accesses had to ensure there was balance between allowing analysts quickly develop the application and ensuring the servers were not seriously altered.

When the PMO was first approached with the list of access defined, they were more than hesitant to accept them. This is where a having PMO involvement in the risk analysis was helpful. I worked to remind the PMO members of the possible risks and business impacts that could result from granting unnecessary accesses. After some discussion about risks to time and integrity, the PMO accepted the list of accesses with only minor changes required. In integration and system testing, developers would not be granted any special access. In these testing phases the support team would install application issuances. Testing would be carried out using only the test ids, as the ids would be able to access the environments through the application code. My teammates and I would support and troubleshoot the environments we had built. A support team domain global group would be created and granted administrators access on the servers. Not being over confident, my team also realized that at times problems would occur on the servers that we could not resolve. Domain groups for troubleshooting would be created and allowed read-only access to each server. Those helping the support team troubleshoot could be dynamically added to the appropriate group for a short period of time. I also documented the list of accesses the project developers would have in the unit environment, relating a description to each access.

A big change from the NT testing environments would be that on the Windows 2000 platform, project security analysts would no longer have elevated rights. Security analysts would be granted the same accesses as every other project analyst in the combined test environment. This would help minimize the risk of drastic changes being made without the knowledge of the support team. Because of the vast changes in access rights granted to the security analyst, I developed a new policy that outlined the new role of a project security analyst. The document explained how a security analyst should work with the support team and others to get their tasks and responsibilities accomplished. With the security policies for access rights defined and ready to be communicated, I moved my efforts forward.

Due to the need to separately administer the combined test environment from standard testing environments, a request was submitted to have a dedicated organizational unit (OU) built into the company AD structure. As the security representative of the support team, I was granted access to administer the OU. Research of Active Directory (AD) technology yielded useful information. I found that an OU is simply a container the AD structure uses to logically organize objects, such as servers, global groups, and user ids, within the domain. Next, I

researched group naming convention standards and defined the appropriate names for each of the groups needed to implement the accesses described in *Figure 1*. I created the groups in the dedicated OU. By this time, the projects responsible for converting the applications had been formed. As such, I was able to setup all the appropriate project developer groups.

My next challenge was to plan an approach to implement the security accesses. I consulted the World Wide Web and other security analysts that had encountered similar situations. Searches on *Microsoft TechNet* yielded quite a lot of information about the types of controls that could be implemented using group policy objects (GPO). I setup a meeting with a fellow security analyst to discuss the usage of a GPO. Even though I can understand how something theoretically works, I decided to consult an informed co-worker. Companies often tweak new technologies to meet their needs. I wanted to understand the standards my company had developed for group policy. The discussion with the analyst yielded very beneficial information. I learned that I could use startup scripts to add the support and temporary troubleshooting domain groups to the appropriate local groups on a server. I also learned that I could link to a GPO in the domain that modeled production security. I thought the last bit of information was phenomenal. Linking to this GPO would model the environment just a bit closer to production security and controls, ensuring further confidentiality and integrity of testing.

I began implementing security within my OU by linking to the GPO that enforced the production like accesses. I also created my own GPO specific to the combined test environment needs. Within this dedicated GPO, I enforced many security details and implemented a startup script. Lastly was the daunting task of implementing developer access. I worked with a security scripter to implement the exact controls as my team had outlined them. An executable was created to install the bulk of the required accesses. The other access details, which would be more dynamic, were completed through group policy. I kept the dynamic controls separate because it would be easier for me to change group policy settings than to have another person alter and recompile an executable. With this category of risks addressed, I only had one category of risks left to mitigate.

3.4 Operation Security and Controls

The last risk management step was to define a security policy around the operations of the combined test environment. The goal of this procedure was to address elements that would mitigate the possibility and/or length of a breach in security during environment operations. Two prominent risks were defined in this category, availability of the test environments and information logging management.

One of the issues identified was the inability to effectively restore a server to a desired level in a timely manner if a terminal error occurred. This was a risk to the availability of the system. In a support team meeting, the decision was made

to schedule and implement a weekly system backup for each of the servers in the combined test environment. We feared that not having backups would open the environment up to an availability risk. Existing hardware and connectivity were utilized to complete the backups, proving this to be a simple way to mitigate this risk.

To minimize the chances that the test environment servers would be susceptible to unauthorized access, the environment servers were set up to log a combination of security, system, and application information. Because the support team was especially concerned about unauthorized access attempts, I developed a policy that defined when and how the server logs should be reviewed. These log reviews would alert the support team to any reoccurring unauthorized access attempts on a server. This information would help us to decide if developers had a valid argument for being granted a new access or if the complaints referred to errors that rarely occurred. Other unauthorized access attempts from individuals not supposed to be in the combined test environment were also of a concern. Whether by ignorance or malicious intent, we did not want others to hinder the controlled environment we were trying to provide. Though server logs would not stop an intruder, they would help the support team to uncover repeated access attempts or access gained by an unrecognized identity. Lastly, the application logs would alert the support team to any problems or errors a server was having that might not have been visible in daily testing; this would help to strengthen the integrity of the testing environment.

3.5 Completing the Security Installation.

Now that the servers were completely loaded with the proper application software, the projects were very near to beginning testing. I just had to complete the security issuances installations. The GPO startup script had worked successfully. So as an administrator on each server, I had three key issuances to install. The issuance to grant developer access was installed on the three unit testing servers in the environment. Initially the issuance failed because I had forgotten to create one of the necessary groups the issuance tried to grant access to on the server. The installation log quickly alerted me to this fact. I fixed this problem and moved on to the directory and host setup installations. Each of these issuances appeared to run perfectly on all six servers. The only true way to test the connectivity was to utilize the projects' code. I would be unable to do so until two of the six projects were up and running unit testing server. So, with the PMO applying pressure on the whole support team, projects were approved to being testing. After all, it was only when the projects began testing that the security solutions implemented could be deemed a success or a failure.

4 Reviewing the Secured Situation

I felt comfortable with the security solutions I had implemented. I had followed a very complete risk management process. When the projects began to test, they posed multiple questions about the new access rights being enforced by the

support team. While developers wanted a consistent solution, they were also counting on higher levels of access to come with that solution. Projects began raising access right concerns to the PMO, but the PMO backed the support team's policy that had been painstakingly researched. Project analysts had no choice but to try the new accesses. Project analysts also had general questions about how the infrastructure and systems were setup. The efforts put forth to document every step of the environment setup and policy was rewarded. All of the un-sensitive documentation was quickly placed on a shared drive. My unending documentation efforts and organization allowed me to effectively communicate a wealth of knowledge to the project analysts. I could refer individuals to the shared documents and have a single place to update them. In actuality, it was not long before pieces of the security policy began to change.

4.1 Security Policy Complications and Solutions

Once the projects began unit testing, so began the testing of the combined test environment and the security policies. While my team had tried to test as much as possible before the projects entered the environment, time eventually expired. Three key issues evolved within the first few weeks of testing. The support team worked to address each of these issues quickly, as they greatly impacted testing. The security policies and risk management plans were updated with each of the situations and solutions. This ensured the development of the environment and its current state could be tracked and/or restored at any point in time. The first major problem encounter surrounded the access granted for temporary troubleshooting.

4.1.1 Limited Troubleshooting Access

Within the fist month, two of the servers within the combined test environment experienced problems caused by the interaction of Windows 2000 and project code. It was at this point that the temporary, read-only troubleshooting access was truly tested for the first time. I added two Windows 2000 subject matter experts to the temporary troubleshooting domain groups. I quickly found out that the read access would not always be sufficient for these types of problems. With this first issue, the troubleshooters needed access to start and stop certain services and alter applications on the servers. I realized at this point that troubleshooters were always going to need varied accesses for different problems. After discussing my thoughts and the risks with the rest of the support team, it was agreed that having timely troubleshooting was a necessity. The support team believed that defining troubleshooter access on a case-by-case basis would not be efficient and would cost each project precious time, resources, and money. A troubleshooter would usually be coordinating their efforts with one of the environment support team members. So, I implemented other troubleshooting groups that would have administrator access on the servers. I created a group for each server in the OU. Then I altered the dedicated GPO's startup script to install the different groups to each of the correct servers. I was then able to control this type of access on a per server basis. Now, the support team carefully coordinates and supervises administrator

based troubleshooting efforts. This arrangement has had an excellent success rate. But, in some cases not even troubleshooting will solve the problems.

4.1.2 System Restoration Failure

At one point, a unit testing server in the environment became completely useable. Developers from different projects had installed conflicting code. This was the first problem resulting from multiple projects testing in the same unit environment. Though the error was found, it could not be corrected. The decision was made to simply restore the server to the last backup point. When the proper contacts were made, the support team was told the restore would take two days. After waiting for the restoration, projects were finally given the green light to continue testing. Unfortunately, the backup restoration did not successfully restore all of the applications' functionality. Our support team had not taken the time to review the exact details for the process. Instead of taking an image of the server, this initial style of backup only copied certain portions of the machine. There was no choice; the servers entered the process of being reloaded from scratch. My team quickly went in search of an alternative backup solution.

One of the support team members contacted a newly developed backup solution coordinator. This solution provided the ability to take a full, complete image of the server. The coordinator even stated that the machine could be restored within a matter of hours, not days. This alternative sounded great, but there were risks involved. To implement this technology on any server, we would have to reload the server from scratch with new hardware. My team weighed the risk of unavailability to the environment for one week to reload versus the possibility that any one of the six servers could be need to be reloading at some time during the testing of these projects. We believed that the environments would experience further problems like this in the months to come, so the reload seemed obvious. To ensure we escalated the situation through the proper channels, the PMO was contacted for approval. They agreed with our solution and the servers were retrofitted and reloaded. Though this seemed like a harsh step, projects were still at the initial stages of development and testing where some delays could be absorbed. Delays and risks to availability from reloading a server in the later stages of testing could have been disastrous to meeting timelines. As such the previous backup method was unacceptable.

During the reload of the servers, I requested that the support team install the security solutions without my help. I believed this would be a great test of the documentation I had created. For the most part, the installation went smooth, but I did take feedback from the team. I was able to update the security implementation procedures to include some more detailed information that was initially overlooked.

4.1.3 Environment Status

Once projects moved into integration and system testing, code interdependencies became much more important. The support team began to realize that projects were experiencing problems they could not initially explain. Many times troubleshooting revealed that it was another project's code failing. As this problem became more prevalent, the support team decided to try and find a resolution to this issue. This issue was a combination risk, including availability and integrity, which needed to be addressed. Ultimately the decision was made to create a "connectivity test" that would be completed each morning after the daily change window. This test was implemented to ensure that test ids could navigate through each projects code. If the test failed, all projects were notified of the code failure.

To implement this solution, the six projects were requested to map out their exact dependencies on the other projects' applications. Once this was completed, each project was requested to provide the support team with methods that could be used to test the dependencies of other projects. After several days of coordination, the returned information was compiled and the test cases for the daily connectivity test were defined. The overall test would ensure that both project interdependencies and the test ids were working correctly. Once implemented, the solution was tested for validity and revised as needed. The connectivity test allowed our team to communicate any testing problems to every user of the environment. It became the cornerstone of daily testing strategies, minimizing efforts that were being wasted on incorrect testing and unneeded troubleshooting. This simple risk mitigation went a long way for return on investment, saving the company hundreds of work hours a month.

4.2 The Impact of the Total Security Solution

To date, the PMO and project analysts have been pleased with the success of the security solutions and policies. The documentation surrounding how and why the security was implemented has been crucial to this success. I cannot comment on the complete results because the projects have not transitioned into production. What I can say is that the overall impact on projects, and moreover for the company, has been great. For the most part, projects have been able to test at their own projected pace dictated by project plan timelines. This was sharp contrast from the NT platform projects, where the unstable environments caused many delays. Utilizing five key subject matter experts for a period of two months has reaped great rewards for my company. The PMO has stated that versus the NT projects, the Windows 2000 projects have had a twenty-two percent reduction in timeline delays.

The concept of set developer access has also been a thriving success. Project analysts can no longer make manual changes to their code in the integration and system levels of test without communicating to the other projects. In testing levels where interdependencies are instrumental to testing success, projects must work through the support team to have changes made. The support team

communicates all changes made to the environment on a daily basis. All of this correlates to more informed projects with a more stable environment. If issues do arise, they can usually be tracked to the changes in the last change window.

The PMO has been thrilled with the results of the overall risk management process and security policies provided. The project management office has begun to implement other mini testing environments for interrelated projects transitioning to the Windows 2000 platform. Though each environment is not implemented exactly the same, the security methodologies I developed are being referenced. The concept of a completing a risk assessment for large effort such as this is quickly growing in the company. Though I was not the first one to complete such an analysis, the PMO visibility has given the concept great attention. Overall, this means better risk management processes, more complete security solutions, and proper documentation for my company.

6 REFERENCES

Broubi, Gardinier, Morimoto, Noel. "Chapter 6: Designing Organizational Unit and Group Structure." Designing Organizational Unit and Group Structure in Windows Server 2003. 4 July 2003. URL: http://www.informit.com/isapi/product_id~%7B66D14A55-F117-4919-ABE6-F86F7CDC35F1%7D/content/index.asp (17 Aug. 2003).

Cole, Fossen, Northcutt, Pomeranz. SANS Security Essentials with CISSP CBK, Version 2.1. United States of America

Dupuis, Clement. "CISSP Study Booklet on Operations Security." 5 Apr 1999. URL: http://shinsoojung.pe.kr/cert/Domain_7.htm (15 Aug. 2003).

Henderson, Thomas. "With Windows 2000, NT grows up." 1 Mar. 2000. URL: http://www.nwfusion.com/reviews/2000/0103rev.html (15 Aug. 2003).

Hoffer, George, and Valacich. Modern System Analysis and Design, Second Edition. Reading: Addison Wesley Longman, Inc, 1999. 769 – 775. The SANS Institute, 2003. 828 – 842.

Sadhwani, Arjan. "Chapter 13 Security and Controls" URL: http://www.cob.sjsu.edu/sadhwa_a/BUS%20188/Chapter13%20Controls%20and%20security.doc (15 Aug. 2003).