GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Auditing & Monitoring Networks, Perimeters & Systems (Audit 507)"
at http://www.giac.org/registration/gsna

A “Black Box” Audit of a Microsoft .NET web-based application

An External Auditor’s Perspective

GIAC Systems and Network Auditor Practical Assignment
Version 2.1
Option 1

William Blake
July 2003

© SANS Institute 2003, Author retains full rights.

Abstract

This paper presents the results of the audit of a web-based application that was conducted from an external viewpoint. That is, the primary objective of the audit was to determine the level to which the application is vulnerable to attack from the Internet. An initial risk evaluation was conducted to determine the assets with the highest level of risk. A checklist was then developed to test the identified areas of risk for possible vulnerabilities. Following the conduct of the nominated tests, the results of the audit are presented as a “Report to Management.”

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Table of Contents

1. Research in Audit, Measurement Practice and Control 3
   1.1. System Description 3
   1.2. Risk Evaluation 4
      1.2.1. Introduction 4
      1.2.2. Risk Rating Tables 4
      1.2.3. Identification of Assets 5
      1.2.4. Identification of Agents 6
      1.2.5. Risk Assessment 7
      1.2.6. Summary of Risk 8
   1.3. Current State of Practice 8
2. Audit Checklist 10
   2.1. Introduction 10
   2.2. Approach 10
   2.3. Checklist 11
      2.3.1. Phase 1 – Project Initiation (PI) 11
      2.3.2. Phase 2 – Footprinting (FP) 11
      2.3.3. Application Assessment (AA) 15
      2.3.4. Application Behaviour (AB) 20
      2.3.5. Other 23
3. Conduct the Audit 24
   3.1. Introduction 24
   3.2. Audit Results 24
   3.3. Residual Risk 29
   3.4. Is The System Auditable? 30
4. Audit Report 31
   4.1. Executive Summary 31
   4.2. Audit Findings 31
      4.2.1. Failed Tests 31
      4.2.2. Passed Tests 34
   4.3. Risks Identified 35
   4.4. Recommendations 35
      4.4.1. Implementation Costs 36
      4.4.2. Compensating Controls 36
5. APPENDIX A 37
   5.1. Nessus Report 37
   5.2. N-Stealth Report 40
   5.3. Error Page from HIDDEN_FIELD manipulation 40
6. References 42

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

1. Research in Audit, Measurement Practice and Control

1.1. System Description

The application being audited is a web-based application that makes exclusive use of Microsoft products. Microsoft's Internet Information Server (IIS) is used as a web server with SQL Server providing database services. The Microsoft .NET framework provides application services.

The application is hosted within the XYZ inc. acceptance environment. A firewall and other security devices are used to protect this environment from the Internet. The application does not currently contain any "live" data – this audit is being undertaken prior to the initial release to Production.

Data passing between client and server is encrypted via 128-bit SSL certificates.

The application itself was developed in-house by XYZ inc. a company that provides customer financial information to a large number of third party financial advisors. It is a portal-type application that allows authorised users to perform actions such as:

- View financial details about their customers;
- View various reports;
- Print forms;
- Submit questions/receive answers from XYZ inc. via an inbuilt 'mail' component

The application 'home page' also contains a News section, which allows XYZ inc. to post articles of interest to their customer population.

As stated above, the application stores personal financial data about a large number of individuals. It is vitally important that this data is not accessible by unauthorised parties. This includes both:

- Legitimate users of the system. These people should only be able to view records to which they have been explicitly granted access.
- Other parties attempting to subvert application security. These people should not be able to view any application data.

The scope of this audit is to determine the level to which the application is vulnerable to attack from the Internet. Auditors were given little information about the operating environment or the controls and practices/procedures governing this environment. These items were considered to be out of scope,

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

as a person deliberately attacking the application would not be overly concerned with any practices, policies and procedures that may be in place.

At the request of XYZ inc., several types of attack were considered to be out of scope. These included Denial of Service attacks, attacks on other devices within the network and physical attacks on the infrastructure.

The scope of the audit strongly influenced the risk assessment and development of the audit checklist. For example, many common control objectives such as reviews of policies and procedures and application patch levels and change control were not directly included.

1.2. Risk Evaluation

1.2.1. Introduction

In order to evaluate the risk to the system, there is a need to develop a risk profile. This will provide a means of 'measuring' the level of risk. As suggested by the Australian Defence Signals Directorate, Threat likelihood X consequence = Risk (Australian Defence Signals Directorate, Handbook 3). Using this approach, the following steps will be followed to develop such a profile:

- Identify the Assets which need to be protected;
- Determine the consequences if these Assets are damaged or destroyed;
- Identify the Threats against the Assets;
- Identify the agents, or possible source of the Threats; and
- Determine the likelihood of the Agents successfully executing the Threats.

1.2.2. Risk Rating Tables

These risk rating tables provide a standard means of estimating consequence of damage, likelihood of damage occurring and resulting level of risk. They are based on tables within the Australian Communications-Electronic Security Instructions 33 (Australian Defence Signals Directorate, Handbook 3). They have been designed for use by Government, however they are still perfectly acceptable for use within the private sector.

Insignificant	Will have almost no impact if threat is realised
Minor	Will have some minor effect on the asset value. Will not require any extra effort to repair or reconfigure the system.
Significant	Will result in some tangible harm, albeit only small and perhaps only noted by a few individuals or agencies. Will require some expenditure of resources to repair (eg "political embarrassment").
Damaging	May cause damage to the reputation of system management, and/or notable loss of confidence in the system's resources or services. Will require expenditure of significant resources to repair.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Serious	May cause extended system outage, and/or loss of connected customers or business confidence. May result in compromise of large amounts of Government information or services.
Grave	May cause system to be permanently closed, and/or be subsumed by another (secure) environment. May result in complete compromise of Government agencies.

Table 1: Consequence Estimation Rating

Negligible	Unlikely to occur
Very Low	Likely to occur two/three times every five years
Low	Likely to occur once every year or less
Medium	Likely to occur once every six months or less
High	Likely to occur once per month or less
Very High	Likely to occur multiple time per month or less
Extreme	Likely to occur multiple times per day

Table 2: Threat Likelihood Rating

		Consequence					
		Insignificant	**Minor**	**Significant**	**Damaging**	**Serious**	**Grave**
Threat Likelihood	**Negligible**	Negligible	Negligible	Negligible	Negligible	Negligible	Negligible
	Very low	Negligible	Low	Low	Low	Medium	Medium
	Low	Negligible	Low	Medium	Medium	High	High
	Medium	Negligible	Low	Medium	High	High	Critical
	High	Negligible	Medium	High	High	Critical	Extreme
	Very high	Negligible	Medium	High	Critical	Extreme	Extreme
	Extreme	Negligible	Medium	High	Critical	Extreme	Extreme

Table 3: Resultant Risk

1.2.3. Identification of Assets

The primary assets used by the application can be broken into two groups, information and equipment. Information refers to the data that is stored within the application and provided to the end user. Equipment refers to the hardware and software used to deliver the application. Due to the fact that Denial of Service attacks and physical attacks on the system are out of scope, this audit will be concentrating primarily on informational assets.

Informational assets can be further divided into the following three sub-groups:

- **Confidentiality of Data** – this includes the data stored within the application database and data being transmitted between client and server. Unauthorised disclosure of personal financial data could have **serious** consequences for both XYZ inc. and users of the application.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- **Integrity of Data** – ensuring the accuracy of information and the integrity of the application processes used for creating, updating and displaying that information is of vital importance. Unauthorised modification of data either within the application or in transit between client and server would have **damaging** consequences.
- **Availability of Data** – this relates to the reliability of the application from a users perspective. This also has obvious implications on the infrastructure delivering the application. Loss of service (i.e. unavailability of data) would have **damaging** consequences.

1.2.4. Identification of Agents

The scope of this audit is to determine the level to which the application is vulnerable to attack from the Internet. Therefore, the only Agents to be considered are external (Internet) users. These Agents can be divided into two groups:

- Known sources without intent. This group consists of the legitimate users of the system. These users typically have a low level of intent – any harm they cause the application may well be accidental. Legitimate users will have varying levels of skill (which is important to understand when considering the likelihood of a successful attack).
- External sources with intent. These people will have a high level of intent and determination – for some reason they want to gain unauthorised access to the application. It has to be assumed that this Agent group has a high level of skill.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

1.2.5. Risk Assessment

The following table provides a risk profile for the XYZ inc. web application.

(Legend for Agent Column: KS – Known Sources (Without Intent). ESWI – External Sources (With Intent)).

Asset	Nature of Threat	Agent	Likelihood	Consequence	Risk
Confidentiality of Information	Valid user can obtain access to system data which they should not be able to view.	KS	Low	Serious	High
	Attacker can gain unauthorised access to system data.	ESWI	Low	Serious	High
	Application data is intercepted in transit.	ESWI	Very Low	Serious	Medium
Availability of Resources and Services	Denial of service attack	ESWI	Very Low	Damaging	Low
	Equipment failure	All sources	Very Low	Damaging	Low
Integrity of Information	Corruption of data	KS	Medium	Damaging	High
	Unauthorised access and tampering with data	KS	Medium	Damaging	High
	Hacking of web page	ESWI	Medium	Damaging	High
	Corruption of data	ESWI	Medium	Damaging	High
Equipment, including Software	Theft of equipment	ESWI	Medium	Damaging	High

Table 4: Risk Profile

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

1.2.6. Summary of Risk

Based on Table 4, the overall risk to the XYZ Inc. web based application is evaluated as HIGH. It should be noted that the likelihood estimations may be somewhat on the high side. As the audit has not yet been performed, little consideration has been given to any mitigation strategies that may be in place (Although it is assumed that some controls are in place, otherwise the likelihood ratings would have been even higher). Following the conduct of the audit, an evaluation of the residual risk will be performed. This evaluation will consider any mitigation strategies and the level to which (if any) they reduce the likelihood of a successful attack. This may, in turn, reduce the overall level of risk.

1.3. Current State of Practice

There is a huge amount of information, both on the Internet and in print, relating to the security of web applications. I found that the real skill required when researching the current state of practice was the ability to quickly determine which pieces of information were useful and which should be discarded.

I began by researching current approaches to auditing web-based applications. Once the approach had been determined, it would then be possible to tailor that approach to suit the particular audit being conducted. An obvious place to start was the material presented during the SANS Audit Track, in particular Rhoades' "Auditing Web Servers and Applications". I found this document to be invaluable – it proved to be my primary resource. It provides a great overview of the type of activities that should be performed during the audit of a web-based application. It also provides a useful list of references for further research.

Peer discussion was another resource of great benefit. I have been working in the IT Security field for a number of years. During that time I have met and worked with many highly skilled individuals, a number of whom specialise in the assessment of web-based applications. By talking to these people I was able to learn a lot about successful, proven auditing approaches and strategies for identifying application weaknesses.

The SecurityFocus web site http://www.securityfocus.com contains some very useful information. This site has recently published two articles on "Penetration Testing for Web Applications" (with additional articles on the subject to be published in the future). These articles complemented Rhoades' document, providing a greater level of detail and re-enforcing the approach and techniques. The SecurityFocus site also hosts the bugtraq mailing list that is a good resource for researching vulnerabilities.

The above resources provided me with sufficient information to begin developing an audit checklist. The next step was to develop testing techniques for assessing specific application vulnerabilities.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

I began researching this from the bottom up. That is, the first step was to determine the hosting platform and discovering whether the software being used contained known vulnerabilities. As this information was not provided to the auditor, this discovery phase became an item on the audit checklist. Once the software was identified, possible vulnerabilities were researched using sites such as:

- http://cve.mitre.org - CVE (or Common Vulnerabilities and Exposures) "provides a list of standardised names for vulnerabilities and other information security exposures". The list is searchable, enabling the researcher to quickly locate published vulnerabilities relating to a particular piece of software.
- Bugtraq – this mailing list is hosted by SecurityFocus.com (mentioned earlier).
- http://xforce.iss.net - a commercial vulnerability research site, hosted by Internet Security Systems
- The CERT Coordination Center at http://www.cert.org - a good site for researching vulnerabilities.
- Microsoft – useful for this audit as the site made exclusive use of Microsoft products. A search for ".NET security" reveals many articles relating to security and the .NET framework. This search also led me to a recent article by Foundstone titled "Security in the Microsoft .NET framework." I found this document provided a useful overview of .NET security architecture.

My final stage of research was to gain a more detailed understanding of data input validation and its relationship to web-based applications. I concentrated on Cross-Site Scripting and SQL Injection techniques, as my previous research suggested that these two attacks had a lot of potential to cause serious damage, whilst at the same time providing a good basis for testing the overall robustness of an application. SPI Dynamics has two very good papers on Cross-Site Scripting and SQL injection, both of which provide a step-by-step methodology for testing whether an application is vulnerable to these techniques. NGSSoftware also publish a great paper titled "Advanced SQL Injection in SQL Server applications." This paper provides an excellent description of SQL Injection techniques.

Using the research into audit approach and vulnerability assessment techniques described above, I was able to develop an audit checklist containing detailed testing procedures. This checklist will provide the auditor with a means of determining whether a web-based application is vulnerable to common Internet attack scenarios.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

2. Audit Checklist

2.1. Introduction

This audit checklist has been designed to test against the threats identified in the risk assessment. The checklist contains a series of tests which can be used as a means of assessing the likelihood of each threat being successfully carried out by an attacker.

Due to the nature of the scope of this audit, the tests will primarily concentrate on the application itself, as opposed to the surrounding environment.

2.2. Approach

The checklist has also been designed in accordance with the audit approach. This audit will follow a phased approach, consisting of the following steps:

Phase 1 – Project Initiation - This phase consists of a briefing with key personnel, describing the assessment approach and discussing details such as IP address, connectivity and configuration procedures.

Phase 2 – Footprinting - Footprinting is the process of acquiring information about the target system and includes:

- Host scanning including port and service discovery; and
- Discovering information on software versions and patch levels.

Footprinting is primarily a tools-based activity.

Phase 3 – Application Assessment - Using the information obtained from the footprinting activity, this phase attempts to exploit the application and associated infrastructure. For this particular audit, the focus was on using IIS/ASP .NET and operating system vulnerabilities directly against the application, including their use to perform functions that the application was not designed to do. Typical tests conducted during this phase include:

- Specific attacks that attempt to 'overflow' input checks;
- Cross site-scripting vulnerability testing;
- SQL Injection vulnerability testing;
- Combination system / OS attacks to expose application data vulnerabilities;
- Deeper network exploration – in the case where further access is afforded by previously uncovered vulnerabilities;
- URL manipulation;
- Attempts to bypass bounds checks; and
- HTML source code reviews to find ways to abuse information

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Phase 4 - Analysis and Reporting - This is the final phase of the vulnerability assessment. This includes:

- Analysis of data;
- Immediate needs out-brief; and
- Report preparation, review and presentation

2.3. Checklist

2.3.1. Phase 1 – Project Initiation (PI)

Identifier	PI–1.
Description	Briefing with key XYZ personnel
Reference	Personal Experience
Control Objective	Ensure all relevant parties are aware of the audit and the approach being taken. Also include the scope inclusions and exclusions.
Risk	It is important to ensure that the audit is being conducted in accordance with XYZ's expectations. It is also important to demonstrate professionalism and knowledge. This will help assure XYZ that the audit has been conducted in a competent manner and the results can be relied upon.
Compliance	Compliance achieved through verbal agreement from all parties.
Testing	Not applicable for this step.
Objective/Subjective	Subjective

2.3.2. Phase 2 – Footprinting (FP)

Identifier	FP-1
Description	Determine open ports on target device(s)
Reference	Personal Experience http://www.securityfocus.com/infocus/1704
Control Objective	Ensure unnecessary services have been disabled.
Risk	Unnecessary services may provide avenues for system compromise. Many services have well-known weaknesses that, if exploited, could provide an attacker with a means of gaining unauthorised access to application data.
Compliance	Only web servicing ports should be open (i.e. 80 and 443).
Testing	Use NMAP to detect open ports. NMAP can be obtained from http://www.insecure.org/nmap/ Use the following command: **Nmap –sS –PT –O –T 3 <IP address>**
Objective/Subjective	Objective.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	FP-2
Description	Identify operating system and application versions.
Reference	http://www.securityfocus.com/infocus/1704
Control Objective	Facilitate research activity - to determine whether any known vulnerabilities exist for identified software.
Risk	There are long lists of application vulnerabilities publicly available on the Internet. Tools designed to exploit some of these vulnerabilities are also available for download. It would be trivial to exploit any such vulnerabilities which may exist on the system.
Compliance	Develop a list of possible vulnerabilities.

Testing	User automated tool-based signature profiling (see FP-4 for detail).
	
	Check HTTP responses using a proxy tool such as Achilles.
	
	Telnet to the application on port 80 (or 443) and type:
	`OPTIONS / HTTP/1.0`
	Press Enter twice. You should receive a response looking something like:
	`HTTP/1.1 200 OK`
	`Server: Microsoft-IIS/5.0`
	`Date: Wed, 04 Jun 2003 11:02:45 GMT`
	`MS-Author-Via: DAV`
	`Content-Length: 0`
	`Accept-Ranges: none`
	`DASL: <DAV:sql>`
	`DAV: 1, 2`
	`Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH`
	`Allow: OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK`
	`Cache-Control: private`
	
	In this example, the web server is Microsoft IIS version 5.0

Objective/Subjective	Objective

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	FP-3
Description	Determine susceptibility to possible weaknesses identified during FP-2. (Note that this step is repeated to a certain extent in the final audit step – AA.O-1. This is deliberate – the final audit step is intended to remind the auditor to consider whether any further testing is warranted).
Reference	Personal Experience
Control Objective	Determine whether operating environment is patched against known vulnerabilities.
Risk	As with FP-2, it is vital that the application is impervious to known, published vulnerabilities.
Compliance	Difficult to determine whether environment is compliant, especially considering the externally-focussed scope of the audit. All that can really be done is to study system behaviour and try to determine whether application is susceptible.
Testing	Compile list of vulnerabilities for software identified during FP-2. Some resources include: • http://cve.mitre.org • Bugtraq - http://www.securityfocus.com/archive/1 • http://xforce.iss.net • http://www.cert.org
Objective/Subjective	Subjective

Identifier	FP-4
Description	Scan device for known vulnerabilities using Nessus.
Reference	Rhoades, p. 58 Personal Experience.
Control Objective	Evaluate application and operating environment for known vulnerabilities.
Risk	Nessus provides a useful starting point for evaluating the application. Its port scanning capabilities can be used to confirm previous tests. Its vulnerability assessment capabilities assist in identifying any areas that might warrant further investigation.
Compliance	Compliance involves an analysis of the Nessus report. There should be no vulnerabilities identified. Note that automated assessment tools are notorious for reporting false-positives, so any weaknesses identified need to be confirmed.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Testing	Nessus can be obtained from: http://www.nessus.org/ Ensure latest signature file is obtained. Run Nessus using the default settings. Do not run DoS attacks – these are out of scope for this audit. Run all other attacks (including Unix attacks – the OS may have been identified incorrectly).
Objective/Subjective	Objective

Identifier	FP-5
Description	Scan for known vulnerabilities using HEAT.
Reference	Rhoades, p. 58 Personal Experience
Control Objective	Same as FP-4
Risk	HEAT is a vulnerability assessment tool developed by Computer Sciences Corporation. For more information, see: http://www.heatscanner.com HEAT may pick up vulnerabilities missed by Nessus and vice versa. Alternately, if both tools identify the same vulnerability the chance of it being a false positive is reduced. For this reason, it is useful to run similar tests with multiple tools to compare results.
Compliance	As with FP-4. No vulnerabilities should be identified.
Testing	Run HEAT against identified IP address(es). Select all tests other than Denial of Service Tests.
Objective/Subjective	Objective

Identifier	FP-6
Description	Scan for CGI weaknesses using N-Stealth
Reference	Rhoades, p.59 Personal Experience
Control Objective	As for FP-4
Risk	Weaknesses in application software could be exploited to gain unauthorised access to information.
Compliance	As with FP-4. No vulnerabilities should be identified.
Testing	A demo version of N-Stealth can be obtained from: http://www.nstalker.com/nstealth/ Run N-Stealth against identified IP address. Select all tests other than Denial of Service Tests.
Objective/Subjective	Objective

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	FP-7
Description	Scan for CGI weaknesses using Nikto
Reference	Rhoades, p.59 Personal Experience
Control Objective	As for FP-4
Risk	As for FP-4. Nikto can be used to verify results from tests performed at FP-6
Compliance	As with FP-4. No vulnerabilities should be identified.
Testing	Nikto can be obtained from: http://www.cirt.net/code/nikto.shtml Note that Nikto requires Perl. Run Nikto against identified IP address, using the following command: **Perl nikto –h <IP address> -allcgi –output <filename>** This command will output the results to the file <filename>. Examine the contents of this file.
Objective/Subjective	Objective

2.3.3. Application Assessment (AA)

The application assessment has been divided into a number of sub-categories, each one designed to examine a different facet of application behaviour.

2.3.3.1. Encryption (EN)

Identifier	AA.EN-1
Description	Ensure all pages are encrypted.
Reference	Rhoades, p.109
Control Objective	Determine whether any information is passed between client and server in clear text.
Risk	Sensitive information could be revealed to eavesdroppers if it is passed in the clear.
Compliance	All pages should be encrypted.
Testing	Traverse the entire site. Ensure “lock” is visible in browser window for all pages. Ensure all URLs begin with “https://”
Objective/Subjective	Objective

Identifier	AA.EN-2
Description	Use packet sniffer to confirm clear text data is not being transmitted.
Reference	Rhoades, p.109
Control Objective	A further check to ensure all data is being encrypted.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Risk	Any data passed in the clear can: a) reveal application data to unauthorised parties; and/or b) be used to mount further system attacks in an attempt to gain unauthorised system access.
Compliance	All pages should be encrypted.
Testing	Run packet analyser whilst accessing all site pages. Examine results, paying particular attention to any links, etc. which leave the site and/or return.
Objective/Subjective	Objective.

2.3.3.2. Logon Process (LP)

Identifier	AA.LP-1
Description	Examine HTTP to determine whether userid/password data is passed in cleartext.
Reference	Rhoades, p.158 Personal Experience.
Control Objective	Determine whether authentication process is appropriately secured.
Risk	The combination of Userid and password is the primary mechanism of identifying the user and determining the information they are able to see. If this is easily compromised then the application is also easily compromised.
Compliance	Logon data is either encrypted (or "masked" in some way) or passed in the clear. If "masked" then an objective opinion on the amount of effort required to reveal the data can be made.
Testing	Use a combination of Achilles and a packet sniffer to analyse client – server conversation during the logon process. The packet sniffer is used to determine whether traffic between client and server is encrypted. Achilles is used to examine whether user-id and logon data is encrypted within the HTTP message.
Objective/Subjective	Objective.

Identifier	AA.LP-2
Description	Analysis of failed logon messages
Reference	Rhoades, p.167 Personal Experience.
Control Objective	Determine whether logon messages reveal excessive information.
Risk	Failed logon messages can give away unnecessary information to an attacker. For example, if an "incorrect password" message and an "unknown username" message are returned to a user, it is possible to determine whether a valid username has been guessed. This can assist brute-force attacks.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Compliance	Error messages should be identical.
Testing	Enter incorrect username. Record results. Enter correct username with incorrect password. Record results. Compare results.
Objective/Subjective	Objective

Identifier	AA.LP-3
Description	Brute force password attack.
Reference	Rhoades, p.174
Control Objective	Determine susceptibility to password brute forcing.
Risk	Given enough time and a valid user name, it may be possible for an attacker to try all possible password combinations until the correct one is found.
Compliance	Compliance for this test is dependent on company policy. Ask what the company policy is with regard to account lockout and reset times and confirm that this standard is being met. For the purposes of this audit, XYZ inc. have stated that permanent lockout should occur after 6 failed logon attempts.
Testing	Ask system administrators about lockout policy. Ensure that they are happy for you to confirm their statements (i.e. they can unlock the testing account if necessary). Attempt to logon with the same user-id and an invalid password 6 times. Try using the correct password on the 7th attempt. This logon attempt should fail.
Objective/Subjective	Objective.

2.3.3.3. Application Code (AC)

Identifier	AA.AC-1
Description	HTML Code review for unnecessary comments
Reference	http://www.securityfocus.com/infocus/1704 Rhoades, p.103
Control Objective	Ensure comments embedded in HTML code do not reveal unnecessary information.
Risk	Comments could give indications of the way the application works. This could even include scripts and other bits of code that have been commented out and not removed.
Compliance	Code should not contain revealing comments.
Testing	Capture all HTML passing between client and server using a tool such as Achilles. Manually review for comments. Comment lines will begin with // or <!- . Meta tags should also be examined. They will typically look like **<meta NAME = xxx**
Objective/Subjective	Objective.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AC-2
Description	Review HTTP traffic for unnecessary information
Reference	http://www.securityfocus.com/infocus/1704 Rhoades, p.96
Control Objective	Determine whether HTTP conversations between client and server reveal excessive information about the application and other software in use.
Risk	HTTP typically reveals information such as software version numbers, methods allowed, etc. This information can be used to formulate attacks.
Compliance	HTTP traffic should not reveal software and operating system version numbers.
Testing	Using the capture from AA.AC-1, review all HTTP conversations. Analyse HTTP headers for values such as Server: xxx.
Objective/Subjective	Objective.

Identifier	AA.AC-3
Description	Hidden Field Manipulation
Reference	http://www.securityfocus.com/infocus/1704 Rhoades, p.198
Control Objective	Determine susceptibility of any hidden fields to manipulation.
Risk	Hidden fields typically contain data passed between client and server. Manipulation of this data before it is passed back to the server could cause the application state to change, thus revealing unauthorised data.
Compliance	Compliance can be difficult to determine and is somewhat based on the auditors experience. Some items are obvious (e.g. cost=$x.xx), however it may be time consuming to exhaustively test more cryptic hidden fields. The best an auditor can do is to utilise all available resources and use their experience to devise suitable tests. The auditor can then make a judgement based on the results of those tests.
Testing	Examine all data captured in previous steps for hidden fields. Analyse data passed in hidden fields. If encoded, attempt to decode the data (e.g. Base 64 encoding could be in use). Change the data and submit back to server. Record results. Repeat the last two steps trying different changes. Record results each time. Make particular note of the content of any error messages generated.
Objective/Subjective	Objective.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AC-4
Description	Input field manipulation
Reference	http://www.securityfocus.com/infocus/1704 Rhoades, p.199 http://www.securiteam.com/securityreviews/5DP0N1P76E.html
Control Objective	Test input validation routines
Risk	Values supplied to the application which are not correctly validated could be used for attacks such as buffer overflow exploits and stealth command insertion.
Compliance	See comments for AA.AC-3
Testing	Examine all input fields. Try the following: • Change field length and submit long strings. • Submit null strings. • Submit control characters. • Submit code snippets such as <script>alert("hello)</script> (This can be used to check for cross-site scripting vulnerabilities). • Anything else the auditor can think of. Record results. Make particular note of any unexpected application behaviour and/or error messages generated.
Objective/Subjective	Objective

Identifier	AA.AC-5
Description	Cookie Checks
Reference	Rhoades, p.138
Control Objective	Determine whether cookies reveal excessive information and/or can be used as an attack vehicle
Risk	Cookies storing session information may allow session hijacking. Session hijacking enables an attacker to impersonate a valid system user.
Compliance	See comments for AA.AC-3
Testing	View cookie in the browser (preferably IE). The domain should be reasonably restrictive. The cookie should be set to "secure". The cookie should expire at the end of the session. Also check whether cookies are the only method of session tracking. To do this, copy a GET or POST request from a current application session using Achilles. Start Achilles on a different machine and send the copied data to the web server. If session hijacking is possible, you will get valid data returned to the browser on the second machine.
Objective/Subjective	Objective.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AC-6
Description	Examine caching properties.
Reference	Rhoades, p.125
Control Objective	Determine what anti-caching techniques are being used.
Risk	Sensitive information could be cached on the client machine. This could be captured by an attacker by various means, such as: Sniffing whilst session is in progress. Obtaining files from cache after session is completed.
Compliance	Caching is somewhat beyond the control of the application, however it is possible to determine whether generally accepted practices are being employed.
Testing	View HTML for every page using Achilles. Check for the page expiry date. This should be set to a time in the past. Check whether pages show the following field: **<meta http-equiv = "pragma" content = "no-cache">** This tells proxy servers to avoid caching the page (Note, however that this field can be overridden by proxy server settings).
Objective/Subjective	Objective

2.3.4. Application Behaviour (AB)

Identifier	AA.AB-1
Description	Cross Site Scripting checks
Reference	Rhoades, p.191 http://www.securityfocus.com/infocus/1704 http://www.cert.org/advisories/CA-2000-02.html http://www.spidynamics.com/whitepapers/SPIcross-sitescripting.pdf
Control Objective	Data/Application Integrity
Risk	Cross-site scripting can be used to run arbitrary code on other machines. This could be used for activities such as stealing cookies or other files.
Compliance	Cross site scripting checks should fail.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Testing	Testing for this vulnerability is a manual process. Use the following code, which if successful will cause an alert box to appear on the screen: **<script>alert("hello")</script>** Check all input fields, especially any fields which generate a URL, link to another page or are used for search input. Also try inserting this code into URLs that appear to be running some kind of CGI script. Rhoades suggests the following approach: • Find an error that will embed data from URL into HTML (i.e. display user input) • Insert sample JavaScript into the URL at the appropriate place.
Objective/Subjective	Objective.

Identifier	AA.AB-2
Description	SQL Injection checks
Reference	http://www.securiteam.com/securityreviews/5DP0N1P76E.html http://www.spidynamics.com/whitepapers/WhitepaperSQLInjection.pdf http://www.nextgenss.com/papers/advanced_sql_injection.pdf
Control Objective	Data/Application Integrity
Risk	SQL injection can be used to manipulate the application database. It may be possible to bypass application security controls to read, add, modify and even delete application data
Compliance	SQL injection should not be possible.
Testing	Testing involves changing the data being sent to the application in an attempt to manipulate the underlying database. Rather than go into great detail here, I suggest that the reader review the references provided. The SPI Dynamics whitepaper, in particular, provides a very good lesson in SQL injection techniques.
Objective/Subjective	Objective.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AB-3
Description	Error Message Analysis
Reference	http://www.securityfocus.com/infocus/1704
Control Objective	Determine whether Error Messages reveal excessive information.
Risk	Error messages may reveal excessive information regarding the application and other software. This information could then be used to mount further attacks.
Compliance	Error messages should not reveal unnecessary information.
Testing	Record all error messages generated during other tests. Look for information such as software versions and source code snippets (stack traces).
Objective/Subjective	Objective.

Identifier	AA.AB-4
Description	URL Manipulation.
Reference	http://www.securityfocus.com/infocus/1704 http://www.kb.cert.org/vuls/id/111677
Control Objective	System Integrity
Risk	URL manipulation is a very common attack which tries to obtain system information and execute commands on the application server. A system vulnerable to such an attack could be easily compromised.
Compliance	URL manipulation should not be possible.
Testing	Directory traversal is a well known URL manipulation vulnerability that allows an attacker to access files and folders anywhere on the server. One way of testing for this vulnerability is to use a tool such as Socket80, which can be obtained from: http://www.astalavista.com/tools/auditing/network/http-server/ To use it, you simply type in the server name or IP address and hit the 'connect' button. If the server is vulnerable, Socket80 will allow you to send arbitrary commands to the machine.
Objective/Subjective	Objective.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AB-5
Description	HTTP transfer method.
Reference	Rhoades, p. 45
Control Objective	Determine whether application is revealing excessive information.
Risk	The application should use POST as opposed to GET to submit sensitive data to the server. The GET method may leave sensitive information in web server logs, the users history file and at other sites (via the HTTP referrer field).
Compliance	POST method should be used to transmit sensitive data to the web server.
Testing	Review all client server interaction using Achilles. Confirm that the method being used is POST.
Objective/Subjective	Objective.

2.3.5. Other

Identifier	AA.O-1
Description	Miscellaneous other tests.
Reference	Original contribution.
Control Objective	Further verification of previous testing.
Risk	It is important to do as much application testing as possible in the given time frames. Results from earlier tests may suggest to the Auditor that further investigation is required.
Compliance	This is not really a compliant item, more of a reminder to Auditors to consider what (if any) further testing may be warranted.
Testing	Review results from previous tests. If unsatisfied with any results, draw on research and experience to devise further tests. Only perform this step if time permits
Objective/Subjective	Subjective.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

3. Conduct the Audit

3.1. Introduction

The following ten checklist items are presented below:

Reference:	Test:	Outcome:
FP-1	Port Scan	Pass
FP-4	Nessus Scan	Fail
FP-6	N-Stealth Scan	Pass
AA.LP-2	Examination of Logon Process error messages	Fail
AA.AC-1	HTML Code Review	Pass
AA.AC-3	Hidden Field Manipulation	Fail
AA.AC-2	Examination of HTTP conversations	Fail
AA.AC-4	Input Field Manipulation	Pass
AA.AC-5	Cookie Analysis	Fail
AA.AB-3	Error Message Analysis	Fail

3.2. Audit Results

Identifier	FP-1
Description	Port Scan. Ensure that only the required ports are visible from the Internet. This helps to confirm that unnecessary services have been disabled.
Stimulus/Response	Yes. Conduct port scan and record results
Results	See copy of Nessus Report at Appendix A.
Assessment	The only ports visible from the Internet were ports 80 and 443. As the site is using SSL, port 443 was expected. Port 80 is used to re-direct users to port 443 (if they were to enter “HTTP” instead of “HTTPS”) – this is also considered acceptable.
Outcome	**PASS**

Identifier	FP-4
Description	Nessus Scan. Examine system for known weaknesses.
Stimulus/Response	Yes. Scan was conducted, with results recorded at Appendix A.
Results	Nessus was configured to scan for all vulnerabilities other that Denial of Service (as this was deemed out of scope by the customer). See results at Appendix A

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Assessment	Nessus identified one vulnerability and several warnings. The vulnerability related to the version of OpenSSL being employed. This will be passed to the system administrator for verification. Notable warnings include: • Revelation of web server version information • Possible exposure of internal IP address information. Both these findings can be used during further testing. It is worth noting that Nessus also identified the system as a Wireless Access Point – obviously a false positive!!
Outcome	**FAIL**

Identifier	FP-6
Description	N-Stealth Scan. Examine system for known weaknesses.
Stimulus/Response	Yes. Scan was conducted, with results recorded at Appendix A.
Results	N-Stealth was configured to scan for all vulnerabilities within its database.
Assessment	N-Stealth did not identify any vulnerabilities.
Outcome	**PASS**

Identifier	AA.LP-2
Description	Examination of Failed Logon Messages.
Stimulus/Response	Yes.
Results	Testing was conducted as follows: • Enter incorrect user id. Applicable HTML snippet from Achilles logs: <P><FONT size="5" face="Verdana, Arial">Authentication Failed (**Unknown UserName**)</FONT> • Enter correct user id with incorrect password. Applicable HTML snippet from Achilles logs: <P><FONT size="5" face="Verdana, Arial">Authentication Failed (**Incorrect Password**)</FONT>
Assessment	As can be seen from the above code snippets, the two tests produced different error messages. This means that it is possible for an attacker to determine when a correct user id has been identified.
Outcome	**FAIL**

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AC-1
Description	HTML code review.
Stimulus/Response	No.
Results	Visit all site pages and record results in Achilles. Review HTML code for comments. Also note hidden fields for later testing. ***Being both very large and difficult to sanitise, the Achilles transcript has not been included in this report.***
Assessment	There are very few comments in the code. There is no legacy code or comments revealing application behaviour.
Outcome	**PASS**

Identifier	AA.AC-3
Description	Hidden field manipulation.
Stimulus/Response	Yes
Results	During the code review, it was noted that a hidden field titled HIDDEN_FIELD was passed between client and server on almost every page. The data in this field was not clear-text, however testing revealed that it was base64 encoded and contained lists of numbers separated by commas. This looked like session data being passed. Further analysis revealed that the same number was always in the first field, suggesting a possible session identifier. This field was selected as a candidate for manipulation. Achilles was used to insert various values (appropriately encoded). Passing a NULL value produced the error message at Appendix A. This error message did not appear to be handled by the application correctly (i.e. it was generated by the application server and contained stack trace information). An example string from HIDDEN_FIELD (taken from Achilles log) **<input type="hidden" name="__HIDDEN_FIELD" value="dDwxNzEyODI1NzU7Oz4=" />** This value can be decoded (using a base64 decoding tool such as that available at http://www.securitystats.com/tools/base64.asp) to: **t<171282575;;>** When the string was changed to **t<;;>**, the error message at Appendix A was produced.

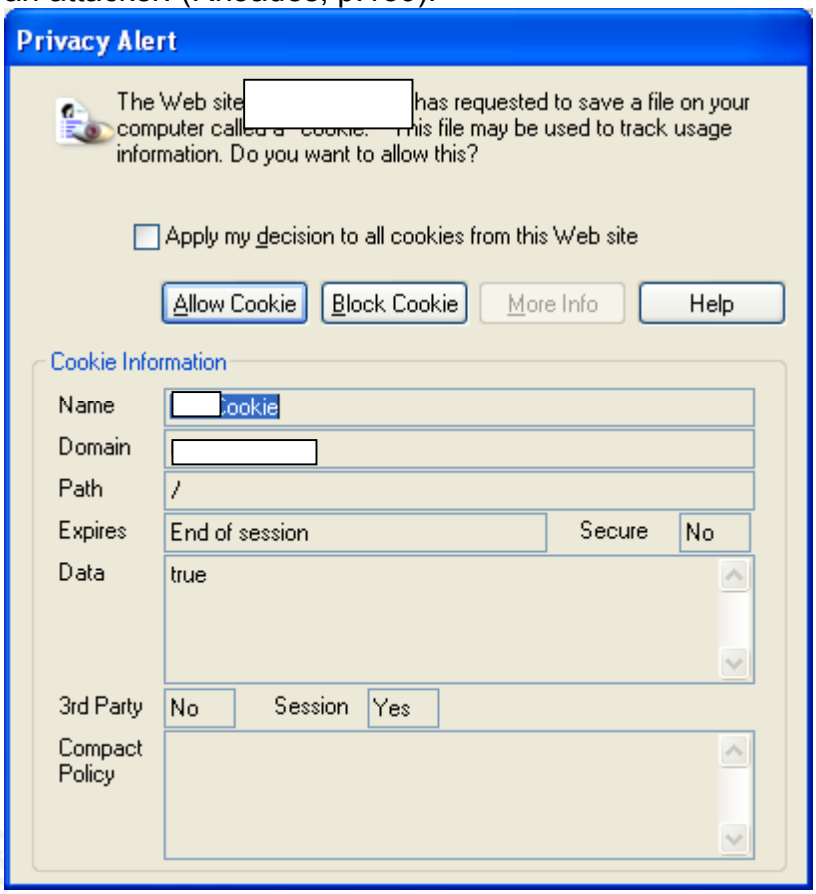
© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Assessment	The audit timeframe did not permit extensive testing of the HIDDEN_FIELD field. The fact that un-handled errors could be produced suggests that further exploitation may be possible. The information revealed by the error message itself is discussed during a later test.
Outcome	**FAIL**

Identifier	AA.AC-2
Description	HTTP examination
Stimulus/Response	No.
Results	HTTP session traffic was examined using Achilles. The following information was revealed. *User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)*
Assessment	Excessive system information is revealed during HTTP sessions. Unless this header data has been deliberately changed, it is simple for an attacker to identify the operating system and application software. Tools are available (e.g. URLscan) to hide this information, making an attacker's task more difficult.
Outcome	**FAIL**

Identifier	AA.AC-4
Description	Input field testing
Stimulus/Response	Yes.
Results	All input fields were checked for incorrect data, including: • Long strings/numbers; • Control characters; • NULL values; and • Command insertion (such as cross-site scripting) The application handled the supplied data appropriately in all cases. For example, when the following was entered into an input field: *<input name="1:txtTFN" type="text" value="<script>alert("hello")</script>"* The following error was produced: *H1>Enter report criteria</H1> <P> !ERROR Please enter a valid Number. *
Assessment	The application does not appear to be susceptible to input field manipulation
Outcome	**PASS**

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AC-5
Description	Cookie Analysis
Stimulus/Response	Yes
Results	Cookies used by the application were examined. A copy of the cookie data is provided below. It was noted that the "secure" flag in the cookie is set to "no". This allows the browser to transmit the cookie in clear-text, increasing the chance of it being captured by an attacker. (Rhoades, p.139).  During cookie analysis, the following test was carried out (1) Copy a GET request from a current Achilles session, which included Cookie data. (2) Initiate an Achilles connection from another machine and paste in the GET request. (3) Analyse results. This test succeeded. In other words, if a valid session cookie can be obtained, session hijacking was possible.
Assessment	The primary defence mechanism preventing an attacker from gaining a session cookie is the use of SSL. This is not a complete solution, however, as it only provides end-to-end protection. It may still be possible to obtain a cookie directly from the client machine. In addition, the cookie should be set to "secure" to ensure that it is transmitted over SSL.
Outcome	**FAIL**

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AB-3
Description	Error message analysis
Stimulus/Response	Yes
Results	All errors produced during application testing were analysed for content. An example is provided at Appendix A (the error generated during the manipulation of hidden fields).
Assessment	Some of the error messages revealed excessive information. The example provided reveals stack information, including source code snippets, as well as software version information. Parameter details are also revealed. An attacker generating a sufficient number of this type of error message may be able to build a good enough picture of the source code to enable development of more detailed attack strategies.
Outcome	**FAIL**

3.3. Residual Risk

As stated during the initial risk evaluation, in order to evaluate the residual risk it is necessary to:

- Review the threats;
- Identify any mitigating factors which have been identified during the audit;
- Revise the likelihood of a successful attack; and
- Assess the residual risk.

The table below presents this assessment. The risk rating tables presented earlier will be used to assess likelihood and risk.

Nature of Threat	Mitigating Factors	Revised Likelihood	Revised Risk Rating
Valid user can obtain access to system data which they should not be able to view.	Test results indicated that this was not possible (AA.AC-1) (AA.AC-4)	Negligible	Negligible
Attacker can gain unauthorised access to system data.	Testing revealed that it is possible to hijack a user session and it may also be possible to manipulate hidden fields to access data. **These factors have resulted in the revised likelihood being increased.**	Medium	High

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Application data is intercepted in transit.	Use of 128-bit SSL certificates makes decryption difficult	Negligible	Negligible
Denial of service attack	Nil – this item was out of scope	No change	Very Low
Equipment failure	Nil – this item was out of scope	No change	Very Low
Corruption of data	Testing was not able to identify any issues. (AA.AC-4)	Very Low	Low
Unauthorised access and tampering with data	No mitigating factors. (FP-4) (FP-6) (AA.AC-1) (AA.AC-2)	No change	High
Hacking of web page	Testing wan unable to identify any relevant vulnerabilities	Very Low	Low
Theft of equipment	Nil – this item was out of scope	No Change	Medium

Based on the above table, the residual risk to the application is assessed as **HIGH.** This compares with the initial assessment, which was also **HIGH**. The primary reason for this is that several audit tests relating to the ability to gain unauthorised access to data failed. This caused the initial likelihood assessment to be RAISED. Reduction of this likelihood measurement will result in the overall level of risk being reduced.

3.4. Is The System Auditable?

I believe the system is auditable. It is possible to devise a checklist containing a number of security tests and measure the system against those tests. The majority of these tests were objective, meaning that they are repeatable and the results reproducible. This allows you to develop a baseline which says that the system has been tested against X, Y and Z. Of course, this is different to saying that the system is completely secure. The application being audited is visible to the Internet, a hostile environment, some of whose population are constantly developing new threats and exploits.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

4. Audit Report

4.1. Executive Summary

XYZ inc. requested an audit of their .NET web-based application during June 2003. The purpose of this audit was to determine the degree to which the application is vulnerable to attack from the Internet.

The audit revealed that the application is well structured and stable. However, a number of vulnerabilities were found that had the potential to affect both end-users and information residing on the server. The most significant vulnerability involved the ability to hijack a current user session, thereby impersonating that user (note, however, that there are mitigation strategies in place to reduce the likelihood of this vulnerability being successfully exploited).

This report presents the audit findings, discusses the risks identified and recommends some mitigation strategies designed to reduce the overall level of residual risk.

4.2. Audit Findings

The auditor undertook the following ten tests. Of these ten tests, six received a FAIL rating, with the remainder being rated as PASS. The outcome of these audit tests is presented in the following below.

4.2.1. Failed Tests

Identifier	EN-1
Objective	Test for known weaknesses using automated vulnerability scanning tools
Outcome	**FAIL**
Results	A product called Nessus was used to scan the application and the hosting environment for known vulnerabilities. Several vulnerabilities were identified, one of which may allow an attacker to execute their own code on the web server. The results of this assessment are included at Appendix A.

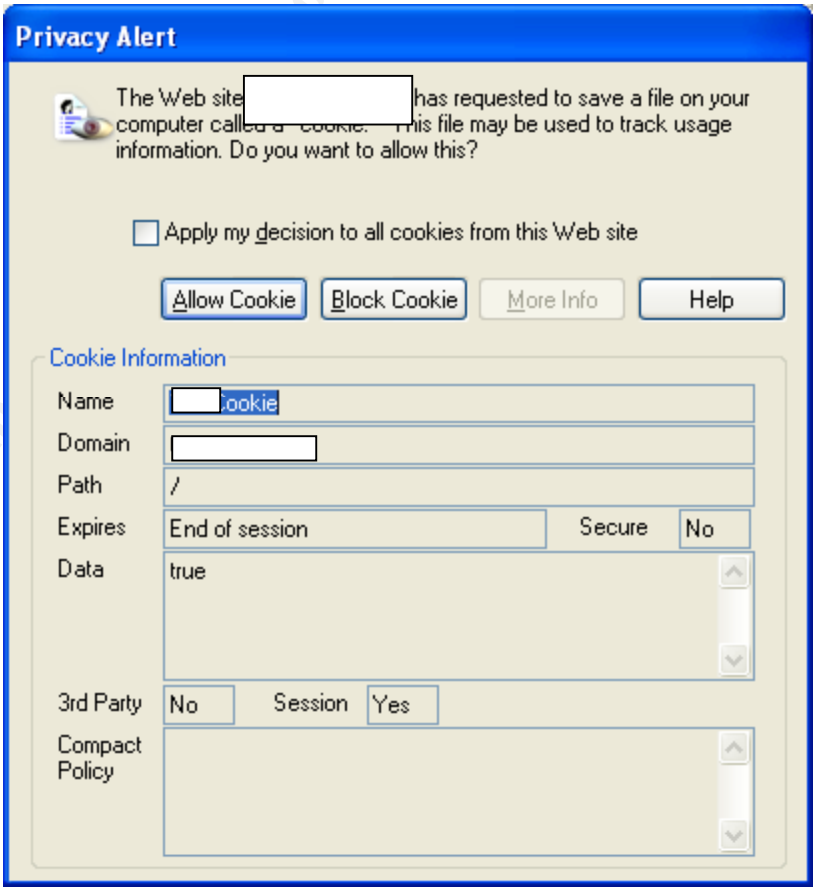
© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.LP-2
Description	Determine whether failed logon messages reveal excessive levels of information
Outcome	**FAIL**
Results	This test involved three steps: • Entering an incorrect user id, which resulted in an error message stating “Unknown User ID”. • Entering a correct user id with an incorrect password, which resulted in an error message stating “Incorrect Password”. • Comparing the two results. As can be seen, the two results were different. The nature of the two messages means that an attacker is able to determine whether they have identified a valid user id. This information can be used for password guessing attacks.

Identifier	AA.AC-3
Objective	Determine whether any hidden fields embedded within application code can be manipulated to exploit the system.
Outcome	**FAIL**
Results	During a review of the application code, it was noted that a hidden field titled HIDDEN_FIELD was passed between client and server on almost every page. The data in this field could be easily decoded, revealing information being passed to the application database. As an example, the data below was captured from a web page: **<input type="hidden" name="__HIDDEN_FIELD" value="dDwxNzEyODI1NzU7Oz4=" />** This value decodes to: t<171282575;;> Bogus information inserted into this field (to replace the ‘171282575’) caused the application to crash with an error that was not appropriately handled by the application. A copy of this error message is attached at Appendix A

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AC-2
Objective	Determine whether traffic passing between client and server revealed information that could be used to mount further attacks.
Outcome	**FAIL**
Results	HTTP session traffic was examined. Software version information was being passed to the client.

Identifier	AA.AC-5
Description	Determine whether application Cookies can be used to exploit the application
Outcome	**FAIL**
Results	Cookies used by the application were examined. Whilst all data within the cookie was encoded, by capturing a valid cookie it was possible to impersonate another user. Also note that the “secure” flag in the cookie it set to “no”. This allows the browser to transmit the cookie in clear-text, increasing the chance of it being captured by an attacker. (Rhoades, p.139). 

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Identifier	AA.AB-3
Objective	Determine whether system Error messages reveal excessive levels of information.
Outcome	**FAIL**
Results	All errors produced during application testing were analysed for content. An example is provided at Appendix A (the error generated during the manipulation of hidden fields). Some error messages reveal information that could be used to mount further attacks.

4.2.2. Passed Tests

Identifier	FP-1
Objective	Port Scan. Ensure that all unnecessary services have been disabled.
Outcome	**PASS**
Results	See copy of Nessus Report at Appendix A. Only ports 80 and 443 were visible from the Internet.

Identifier	EN-3
Objective	N-Stealth Scan. Scan application for known weaknesses.
Outcome	**PASS**
Results	A copy of the N-Stealth report is attached at Appendix A. N-Stealth did not identify any vulnerabilities.

Identifier	AA.AC-1
Objective	Review all application code visible to the end user.
Outcome	**PASS**
Results	The HTML contents of all pages were examined. No security-related issues were identified.

Identifier	AA.AC-4
Objective	Determine whether data input fields are subjected to appropriate validation routines
Outcome	**PASS**
Results	All input fields were checked for incorrect data, including: • Long strings/numbers; • Control characters; • NULL values; and • Command insertion (such as cross-site scripting) The application handled the supplied data appropriately in all cases. For example, when the following was entered into an input field:

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

	<input name="1:txtTFN" type="text" value="<script>alert("hello")</script>" The following error was produced: *H1>Enter report criteria</H1> <P> !ERROR Please enter a valid Number. *

4.3. Risks Identified

The audit tests that received a FAIL rating revealed the following risks to the application and XYZ Inc.:

- A version of OpenSSL in use that may allow an attacker to gain control of the system. If this vulnerability was successfully exploited an attacker would be able to access application data. It may also be possible to use the exploited server as a stepping-stone to mount further attacks against other XYZ systems.
- Many components of the application reveal excessive levels of information. The easier it is for an attacker to identify a system, the easier it is for them to formulate an attack plan. Making an attacker's job more difficult may discourage them enough so that they will look for an easier target.
- It is possible to hijack a current session. This allows an attacker to bypass all authentication mechanisms and gain unauthorised access to application data. It should be noted that the risk of this attack being successfully executed is reduced to a certain extent by the use of SSL. This is not a complete solution, however, as it only provides end-to-end protection. It may still be possible to obtain a cookie directly from the client machine. In addition, the cookie is not set as "secure". This means that it is possible to transmit the cookie in the clear (i.e. the use of SSL is not mandatory).
- It may be possible to manipulate hidden fields within the application to reveal application data. (Note that time restrictions prevented auditors from exhaustively testing this risk.)

4.4. Recommendations

The following recommended measures would enhance the security of the application and environment:

- Ensure that all software is patched to the latest levels. Also confirm that appropriate processes and procedures are in place to track and apply software patches.
- Implement a single error message for unsuccessful logon attempts, such as "The username/password combination you entered was incorrect. Please try again."
- Consider applying further encryption controls to sensitive data being passed between client and server.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- Review data passed in HIDDEN_FIELD field. Consider whether bogus data entered in this field could cause the application to malfunction.
- Remove IIS version information from all dynamic linked libraries (DLLs).
- Remove .NET version information.
- Set session cookie to “secure”.

4.4.1. Implementation Costs

The following section provides indicative levels of effort that would be required to implement the report recommendations.

1. Ensure that all software is patched to latest levels. **ESTIMATED EFFORT**: Approximately one man-day.
2. Implement single logon error message. **ESTIMATED EFFORT**: Two man-days.
3. Further encryption controls to sensitive data. Without undertaking a source code review, it is difficult to estimate the level of effort required to make this change. It could range from a few days to a number of weeks.
4. Review data passed in HIDDEN_FIELD. As with the previous item, it is difficult to estimate the required level of effort. This should be referred to the application developers for further analysis.
5. Remove software version information. This can be achieved using a tool called URLScan. **ESTIMATED EFFORT:** Three man-days.
6. Change to cookie parameters. **ESTIMATED EFFORT:** Approximately one man-day.

4.4.2. Compensating Controls

It was not possible to provide a reasonable level of remediation effort for two the report recommendations. There is one mitigating control in place that reduces the level of exposure created by the weaknesses identified. This is the use of 128-bit SSL encryption to protect all session traffic. This level of encryption makes it difficult for an attacker to capture session cookies (which are used to hijack current sessions). Note, however that end-to-end encryption does not prevent a cookie from being obtained from a users machine. This means that session hijacking may still be possible.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

5. APPENDIX A

5.1. Nessus Report

Below is a copy of the report produced by Nessus.

Nessus Scan Report

This report gives details on hosts that were tested and issues that were found. Please follow the recommended steps and procedures to eradicate these threats.

Scan Details

Hosts which where alive and responding during test
1

Number of security holes found
1

Number of security warnings found
3

Host List

Host(s)
Possible Issue

192.168.1.100
Security hole(s) found

[return to top]

Analysis of Host

Address of Host
Port/Service
Issue regarding Port

192.168.1.100
http (80/tcp)
Security warning(s) found

192.168.1.100
https (443/tcp)
Security hole found

192.168.1.100
general/tcp
Security warning(s) found

192.168.1.100

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

general/udp
Security notes found

Security Issues and Fixes: 192.168.1.100

Type
Port
Issue and Fix

Warning
http (80/tcp)

It seems that your web server rejects requests
from Nessus. It is probably protected by a reverse proxy.

Risk factor : None

Solution : change your configuration
if your tests to be accurate
Nessus ID : 11238

Informational
http (80/tcp)
A web server is running on this port
Nessus ID : 10330

Informational
http (80/tcp)
The remote web server type is :

Microsoft-IIS/5.0

Solution : You can use urlscan to change reported server for IIS.
Nessus ID : 10107

Vulnerability
https (443/tcp)

The remote host seems to be using a version of OpenSSL which is
older than 0.9.6e or 0.9.7-beta3

This version is vulnerable to a buffer overflow which,
may allow an attacker to obtain a shell on this host.

*** Note that since safe checks are enabled, this check
*** might be fooled by non-openssl implementations and
*** produce a false positive.
*** In doubt, re-execute the scan without the safe checks

Solution : Upgrade to version 0.9.6e (0.9.7beta3) or newer
Risk factor : High
CVE : CAN-2002-0656, CAN-2002-0655, CAN-2002-0657, CAN-2002-0659, CVE-2001-111
BID : 5363
Nessus ID : 11060

Warning
https (443/tcp)

Asking the main page, a Content-Location header was added to the response.
By default, in Internet Information Server (IIS) 4.0,
the Content-Location references the IP address of the server
rather than the Fully Qualified Domain Name (FQDN) or Hostname.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

This header may expose internal IP addresses that are usually hidden or masked
behind a Network Address Translation (NAT) Firewall or proxy server.

Solution: See http://support.microsoft.com/support/kb/articles/Q218/1/80.ASP

Risk factor : Low
CVE : CAN-2000-0649
BID : 1499
Nessus ID : 10759

Informational
https (443/tcp)
A TLSv1 server answered on this port

Nessus ID : 10330

Informational
https (443/tcp)
A web server is running on this port through SSL
Nessus ID : 10330

Informational
https (443/tcp)
Here is the SSLv3 server certificate:
Certificate:
Data:
Version: 1 (0x0)
Serial Number: 0 (0x0)
Signature Algorithm: md5WithRSAEncryption
Issuer: REMOVED
Validity
Not Before: Jun 3 04:39:10 2003 GMT
Not After : Jul 3 04:39:10 2003 GMT
Subject: REMOVED
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
REMOVED
Exponent: 65537 (0x10001)
Signature Algorithm: md5WithRSAEncryption
REMOVED

Nessus ID : 10863

Informational
https (443/tcp)
This TLSv1 server does not accept SSLv2 connections.
This TLSv1 server also accepts SSLv3 connections.

Nessus ID : 10863

Informational
https (443/tcp)
The remote web server type is :

Microsoft-IIS/5.0

Solution : You can use urlscan to change reported server for IIS.
Nessus ID : 10107

Informational
https (443/tcp)
The address in Content-Location is: 1.20.55.55
CVE : CAN-2000-0649
BID : 1499
Nessus ID : 10759

Warning
general/tcp

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The remote host is a Wireless Access Point.
You should ensure that the proper physical and logical controls exist around the AP.

Risk factor : Medium/Low
Nessus ID : 11026

Informational
general/tcp
HTTP NIDS evasion functions are enabled.
You may get some false negative results
Nessus ID : 10890

Informational
general/tcp
Remote OS guess : D-Link DI-713P Wireless Gateway (2.57 build 3a)

CVE : CAN-1999-0454
Nessus ID : 11268

Informational
general/udp
For your information, here is the traceroute to 192.168.1.100 :
192.168.52.2
192.168.1.100

Nessus ID : 10287

This file was generated by Nessus, the open-sourced security scanner.

5.2. N-Stealth Report

The following report was produced by N-Stealth during the conduct of this audit.

N-Stealth Report

N-Stealth report for xyz.com (192.168.1.100)
Date: 11/06/2003 9:43:45 PM

Scan Rule: Normal

192.168.1.100
Host name: **xyz.com**
Port: 80
Server: Unknown Server

No bugs were detected.

N-Stealth 3.7 (Build 67)

5.3. Error Page from HIDDEN_FIELD manipulation

The following error was produced during manipulation of the HIDDEN_FIELD hidden field.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Request could not be completed
HTTP Status : 500 Internal Server Error

Server Error in '/SessionProxy' Application.

Value cannot be null. Parameter name: String

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.ArgumentNullException: Value cannot be null. Parameter name: String

Source Error:

```
An unhandled exception was generated during the execution of the
current web request. Information regarding the origin and location of
the exception can be identified using the exception stack trace
below.
```

Stack Trace:

```
[ArgumentNullException: Value cannot be null.
Parameter name: String]
   System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo
info) +0
   System.Web.UI.Page.LoadPageViewState() +89
   System.Web.UI.Page.ProcessRequestMain() +421
```

Version Information: Microsoft .NET Framework Version:1.0.3705.288; ASP.NET Version:1.0.3705.352

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

6. References

Anley, Chris, "Advanced SQL Injection in SQL Server Applications", 2002, http://www.nextgenss.com/papers/advanced_sql_injection.pdf

Australia Defence Signals Directorate, "Handbook 3 – Risk Management", Australian Communications – Electronic Security Instruction 33 (ACSI 33), 20 December 2000, http://www.dsd.gov.au/infosec/acsi33/HB3.html

CERT Co-ordination Centre, "CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests", last revised 3 February 2000, http://www.cert.org/advisories/CA-2000-02.html

CERT Co-ordination Centre, "Vulnerability Note VU#111677 – Microsoft IIS 4.0/5.0 vulnerable to directory traversal via extended Unicode in url (MS00-078), revision 22, last updated 18 September 2001 http://www.kb.cert.org/vuls/id/111677

Foundstone Strategic Security, "Security in the Microsoft .NET Framework – An analysis by Foundstone, Inc. and CORE Security Technologies:, 2003. http://www.foundstone.com

Melbourne, Jody and Jorm, David, "Penetration Testing for Web Applications", 16 June 2003. http://www.securityfocus.com/infocus/1704

Melbourne, Jody and Jorm, David, "Penetration Testing for Web Applications (Part Two)", 3 July 2003. http://www.securityfocus.com/infocus/1709

Rhoades, David, "Auditing Web Servers and Applications" version 1.4, from SANS Audit Track – day 3. 2002.

SPI Dynamics, "Cross-Site Scripting – Are your applications vulnerable?", http://www.spidynamics.com/whitepapers/SPIcross-sitescripting.pdf (accessed June 2003)

SPI Dynamics, "SQL Injection – Are your web applications vulnerable?", http://www.spidynamics.com/whitepapers/WhitepaperSQLInjection.pdf (accessed June 2003)

Standards Australia, HB 231:2000 – Information Security Risk Management Guidelines, Sydney: Standards Australia International, 2000.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Last Updated: April 27th, 2017

Upcoming Training

SANS Secure Europe 2017	**Amsterdam, Netherlands**	**Jun 12, 2017 - Jun 20, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**