

Web Application Security

Risk Analysis

Version: 2.0

Date: 12/08/2006

Author: Jeff Guhin

Course: TS5508

Instructor: Philip Davis

Table of Contents

Overview	3
Objective	3
Scope.....	3
Assessment.....	3
Risks	4
How Hackers Get In.....	4
Examples of vulnerabilities.....	4
Table 1: (OWASP 2006).....	4
The Top Ten Vulnerabilities – a closer look	4
Table 2: (OWASP 2006).....	5
How do these Vulnerabilities Affect Our Customers?	5
Potential Safeguards	5
Manual Safeguards.....	6
Table 3: (Whitaker, 2006).....	10
Automated Safeguards.....	10
Automated Web Vulnerability Scanning Vendors	10
Risk Reduction Recommendations.....	10
People.....	10
Technology	10
Process	11
Conclusion	11
Annotated Bibliography.....	12
Project References	14

Overview

ACME company has recently purchased and began implementation of a PeopleSoft Enterprise Resource Planning (ERP) software package. ACME is filing for Initial Public Offering (IPO) and wants to be prepared for external audits and reporting.

Objective

This document was created to generate an awareness of a perceived security gap at the application level and generate further discussion of preventative methods. This document identifies the areas that need to be addressed but is not intended to propose a final solution. The next step would require an approved project plan to implement a web application security strategy.

Scope

This document aims to distinguish the difference between network level security and web application level security. ACME's Network Systems team produces routine network vulnerability scans at the network layer for intrusion detection, but ACME does not currently have tools or processes in place to scan for vulnerabilities at the web application level.

Hackers are concentrating their efforts on attacking applications in websites: 75% of cyber attacks are launched on shopping carts, forms, login pages, dynamic content etc. Firewalls, SSL and locked-down servers are futile against web application hacking (Whitaker 2006).

Most defense at network security level will provide no protection against web application attacks since they are launched on port 80 – which has to remain open. In addition, web applications are often tailor-made, therefore tested less than off-the-shelf software, and are more likely to have undiscovered vulnerabilities, and when packaged software such as the ERP PeopleSoft suite undergoes customizations and integrations new risks can be introduced like improper error handling, sql injections or cross-site scripting attacks. As ACME continues to grow and prepares for filing IPO we become an attractive target for attackers.

There have been an alarming number of Corporations and Universities that have had data breaches Refer to the URL: <http://www.privacyrights.org/ar/ChronDataBreaches.htm> and scroll to the bottom of that web page to view the public record of exploited systems.

Manually auditing a website for vulnerabilities is virtually impossible - it needs to be done automatically and on a regular basis. Many Universities such as Harvard, Princeton, University of Texas, Rhode Island and corporations such as the FDIC, Department of Defense, Bank of America, and Motorola currently use automated tools in their security processes (SPI Dynamics, 2006).

Assessment

The Network team focuses on the network not each specific web application. The ERP security team has been primarily focused on roles and permission lists as they relate to authorization, authentication, and access control. When I was assigned the role as QA Lead for ERP security I took multiple security courses and workshops and conducted extensive research on how attackers (hackers) exploit systems.

ACME has been growing at a rapid rate and the QA team has only been around since 2003. Testing web application security in many large organizations is often handled by a dedicated security team with specialized skill sets and tools that work in conjunction with the various IT department teams.

There is an opportunity for ACME to implement preventive measures to insure that web servers are patched, 3rd party components installed don't have known exploits, client side code is secure and adopt tools to automatically scan for web application vulnerably.

Risks

The consequences of a security breach are great: loss of revenues, damage to credibility, legal liability and loss of customer trust.

The Peoplesoft applications themselves are thoroughly tested when shipped at the vanilla state. When we begin to make our own customizations and system integrations new doors could be opened that could leave the system vulnerable to exploits at the application level regardless of what safeguards are in place at the network layer.

The Peoplesoft system is not the only ACME system at risk. Web applications that are outward facing or developed with custom code are also potentially at risk.

How Hackers Get In

Browser-based attacks use flaws in the web-based application code. Software most vulnerable to these types of attacks includes:

- User interface code -- provides the look and feel of the site
- Web server -- supports the physical communication between the user's browser and the web applications
- Front-end applications -- interfaces directly with the user interface code, and back-end systems

Examples of vulnerabilities

Hack attack	What hackers use it for
Cookie Poisoning	Identity theft/session hijack
Hidden Field Manipulation	eShoplifting
Parameter Tampering	Fraud
Buffer Overflow	Denial of Service/ Closure of Business
Cross-Site Scripting	Hijacking/Identity Theft
Backdoor and Debug Options	Trespassing
Forceful Browsing	Breaking and Entering
HTTP Response Splitting	Phishing, Identity Theft and eGraffiti
Stealth Commanding	Concealing Weapons
3 rd Party Misconfigurations	Debilitating a Site
Known Vulnerabilities	Tacking control of the site
XML & Web Services Vulnerabilities	New layers of attack vectors & malicious use
SQL Injection	Manipulation of DB information

Table 1: (OWASP 2006)

The Top Ten Vulnerabilities – a closer look

Application Threat	Negative Impact	Example of the Business Impact
Unvalidated Input	Attackers can use these flaws to attack backend components through a web application	Alter application logic, access confidential information, etc.
Broken Access Control	Attackers can exploit these flaws to access other users' accounts, view sensitive files, or use unauthorized functions	Risk to confidential information. Malicious users can reach administration parts of the web application.

Broken Authentication and Session Management	Session Hijacking.	Attackers can compromise passwords, keys, sessions etc. and cash out someone else's account.
Cross Site scripting	Identity Theft	Hackers can impersonate legitimate users, and steal their accounts
Buffer overflow	Denial of Service (DoS)	Site unavailable to customers, could allow execution of malicious code on the server's operating system.
Injection Flaws	Attackers can manipulate queries to the database	Hackers can access backend database information, alter it or steal it.
Improper Error Handling	Attackers can gain detailed system information	Malicious system reconnaissance may assist in developing further attacks
Insecure Storage	Weak encryption techniques may lead to broken encryption	Confidential information (SSN, Credit Cards) can be decrypted by malicious users
Denial of Service	Legitimate users can no longer access or use the application	System is inaccessible, loss of business
Insecure Configuration Management	Web, application or database server are configured insecurely	Attackers may exploit known vulnerabilities in the web, application or database servers

Table 2: (OWASP 2006)

How do these Vulnerabilities Affect Our Customers?

Our Customers can be affected in a variety of ways: from identity theft to session hijacking to the compromise of confidential and private user data.

- Cross-Site Scripting (XSS) is one of the leading methods used in identity theft; it attacks the user via a flaw in the website that enables the attacker to gain access to login and account data from the user. Many of the phishing email-based schemes use cross-site scripting and other application layer attacks to trick users into giving up their credentials.
- SQL injection is one of the main attacks used when backend databases are compromised. General consensus has pegged SQL injection as the method used behind the compromise of credit card numbers in 2004 according to ComputerWeekly (2004).
- Cookie Poisoning cases are still reported where cookies aren't properly secured, allowing an attacker to 'poison' the cookie, hijack active sessions or manipulate hidden fields to defraud ecommerce sites.

As web applications become more pervasive and more complex, so do the techniques and attacks hackers are using against them. Recent new vulnerabilities and attack methods discovered or reported show an alarming trend toward attacks with multi-faceted damages and even anti-forensics capabilities. This means hackers are using more powerful attacks to cause significantly more damage, while at the same time getting better at covering their tracks.

Potential Safeguards

Many organizations today, especially ones in regulated industries, test security using costly manual processes that cannot address all potential website risks. There are preventive measure that the organization can make that can be incorporated into existing processes by adopting the use of application scanning tools for known vulnerabilities. This can provide additional reassurance to the stakeholders.

Manual Safeguards

Some preventive measures to protect against common attacks are outlined below in Table 3.

Attack	Type	How to Protect:
Performing Reconnaissance	Footprinting	<p>The objective is to verify sensitive information is not revealed in source code or error messages such as passwords, usernames, database names that could be useful to an attacker.</p> <p>Old code snippets found in comments may provide hints for an attacker. You can manually scan the pages or use an automated tool such as Grep for larger sites.</p>
Guessing Files and Directories	Footprinting	<p>Make sure to configure the web server to disable directory browsing so it can't serve pages outside the application.</p> <ul style="list-style-type: none"> • Look for patterns such as sequential numbers incrementing or decrementing for file name and account numbers. • Look for sub-sites in the root site. Folders titles admin, control, cp are good directories to try. • Admin pages could be running on a different port on the web server. There are a limited number of port available, and you may be able to recognize those that are already reserved. Application specific ports are typically greater than 1024
Risks with Third Party Components	Footprinting	<p>Hackers subscribe to mailing lists that publish vulnerabilities. Many of these provide details of what the issue is and how to exploit it. Audit the web server for third party components. If any are found research how frequently it gets updated, if it has been subjected to previous attacks or if there are security concerns. Any component that is installed should be validated for proper input and output validation.</p>
Google Hacking	Footprinting	<p>Information that the Google Hacking Database identifies:</p> <ul style="list-style-type: none"> • Advisories and server vulnerabilities • Error messages that contain too much information • Files containing passwords • Sensitive directories • Pages containing logon portals • Pages containing network or vulnerability data such as firewall logs. <p>The easiest way to check whether your web site & applications have Google hacking vulnerabilities, is to use a Web Vulnerability Scanner. A Web Vulnerability Scanner scans your entire website and automatically checks for pages that are identified by Google hacking queries. (Note: Your web vulnerability scanner must be able to launch Google hacking queries).</p>
Bypass Restrictions on Input Choices	Client Side	<p>Prevention is the best approach in this case. All restrictions enforced on the client side should be verified on the server side to avoid this attack.</p> <p>In this example, lets refer to a web form that takes orders online. An attacker has two options to perform this attack.</p> <ol style="list-style-type: none"> 1. Save the web page locally. View the source and modify default restrictions. The attacker can use the modified page locally instead of the page served by the web server to send the modified values to the remote server. 2. Use a program to automate sending requests with pre-selected or dynamically created values to bypass the GUI imposed selection criteria. This technique generates more possibilities in a short amount of time compared to the first option.

Web Application Security Risk Analysis

Attack	Type	How to Protect:
Bypass Client-Side Validation	Client Side	<p>Any real validation should also be handled on the server side, but there may be performance hit in result.</p> <p>As a tester, you want to identify what is being validated, remove the validation, and determine if the data is being validated on the server-side. If its not, write a defect. Any mismatch in data could cause maintenance problems for developers.</p> <p>You can turn off the ability to run scripts in the browser by using the browser settings, but it may break other unrelated features like navigation. An alternative would be to save the HTML source locally and edit the validation script and change the relative URLs to absolute.</p>
Hidden Fields	State based	<p>If the application does use hidden fields, determine if the information stored there would be useful to an attacker. Does it contain session information or pass product details such as prices or quantities from one page to the next?</p> <p>Avoid use of hidden fields at all. If they are necessary consider using obscure field names that are less obvious and encrypting or hashing the values.</p> <p>Open the HTML source and search for the word: "hidden". Save the page locally, change relative links to absolute links, remove the word "hidden" in the HTML source, Save and reload the page. Removing the "hidden" value displays a textbox on the page that can now be modified.</p>
CGI Parameters	State based	<p>Validate all input on the server side.</p> <p>This attack can be conducted by:</p> <ul style="list-style-type: none"> • First browse the site and note the address bar value. • Hover the mouse over hyperlinks to note the URL displayed in the information bar at the bottom of the browser. • Note any CGI parameter (the values after the ? in a URL) and determine what that data represents and if it could benefit an attacker. • You can modify the HTML source to change the POST method values. • You can simply modify the URL for GET parameters in the address bar.
Cookie Poisoning	State based	<p>Consider carefully what data is getting stored on client machines. Use certificates or server authentication processes for granting access instead of cookies. http://www.lodoga.co.uk/attackinfo/thethreat/examples/cook.htm</p> <p>Consider encrypting any sensitive information located in cookies.</p> <p>This attack can be conducted by: Look for expiration dates userid and password information in your cookies folder located on the client machine. For example Internet Explorer stores cookies in: C:\Documents and Settings\username\Cookies. You can use Google to find the specific cookie formats for the browser you are using. Determine if there is any sensitive information being stored, see if you can edit the values and revisit the website and watch the behavior.</p>

Attack	Type	How to Protect:
URL Jumping	State based	<p>Developers may compare the last visited page to the page requested to validate they are following the proper sequence. It is slightly more secure to use cookies to store this information than hidden fields or CGI parameters because temporary cookies are passed in the HTTP header which users can't control though the browser and is harder to modify.</p> <p>It is helpful for the tester to have a page map and diagram the intended sequence of pages.</p> <ul style="list-style-type: none"> • First navigate through the site as how the developer intends a user to experience the site, and note the addresses of each page and their sequence. • Then use the list to attempt to access the URL's at random, and look for any error messages the attacker might find useful. • You can modify hidden fields, cookies values, or the HTTP referrer to try and force access. If you can access a page this way then write of a defect.
Session Hijacking	State based	<p>Andrews and Whittaker (2006), recommend a number of precautions:</p> <ul style="list-style-type: none"> • Make sure the sequence of identification numbers are unpredictable • Cookies are generally more difficult to modify than hidden fields or CGI parameters. You can protect them by using mechanisms like setting the secure flag (so they cannot be "sniffed" unencrypted on the network). • Time-out session identifiers so someone cannot reuse them after a predetermined period of time. • Allow users to log out and clear their session. • Utilize the HTTP referrer field to identify multiple clients browsing with the same identifier. • Make sure session cookies are sent over secure channels so they can't be captured in transit. <p>This attack can be conducted by: Gather multiple session identifiers and attempt to decipher a pattern. IF you can find a pattern, replace the session identifier value with another valid and request the page again. Attempt this scenario several time and if you are able to view personalized information of another user, write the defect up.</p>
Cross-Site Scripting:	User supplied input data	<p>The simplest way to avoid XSS is to add code to a Web application that causes the dynamic input to ignore certain command tags that are interpreted as code. It is common for developers to have white-list that include a list of acceptable tags.</p> <p>This attack is commonly attempted through form input fields that write to a database or file or a placed in a URL replacing the CGI parameters (Andrews, Whittaker, 2006). Once XSS has been launched, the attacker can change user settings, hijack accounts, poison cookies with malicious code, expose SSL connections, access restricted sites and even launch false advertisements.</p>

Web Application Security Risk Analysis

Attack	Type	How to Protect:
SQL Injections	User supplied input data	<p>SQL injection attacks can be avoided with server side input validation. Any validation on the client side can be tampered with. Andrews, Whittaker, (2006), recommend:</p> <ul style="list-style-type: none"> • Filter special characters from URL's, forms, and user controls. • Use stored procedures • Create database credentials for each user and only allow access to what they need. <p>An attacker focuses on submitting queries into web forms or parameter calls to an API. This typically starts with inserting a select command into the form to determine if any mitigations are in place to protect against this attack. Once an attacker determines this is a vulnerability there is potential to steal data, insert values into the database, or perhaps drop tables from the database. See the examples of actual sql statements in the Testers Toolbox section listed below.</p>
Directory Traversal	User supplied input data	<p>Andrews and Whittaker (2006), provide the following recommendations with the disclaimer that security can also slow down the delivery of the pages and potentially affect performance:</p> <ul style="list-style-type: none"> • Restrict web application to server pages from the web root directory and sub directories. • Use Access Control Lists to grant page access • Server side filtering <p>To conduct this attack: Use the web application in the intended manner and note the URL addresses that are accessing files to be rendered in the browser. You can attempt to access other files if you can determine other file names. Enter commands to such as "../" in the URL to attempt to navigate up directory levels. If the attacker is able to navigate all the way up to the servers cmd.exe file they could run the windows command shell and view the directory listing on the C drive. This is a huge danger. Refer below to the Testers Toolbox link for syntax examples.</p>
Buffer Overflows	Language based	<p>The easiest way to protect against this attack is to truncate the extra data locally and let the user know. But also add server side validation because an attacker can bypass any client side validation.</p> <p>The idea is to fill every input field with as much data as possible. Look for parameters or form fields and attempt to insert 100,000 characters to be passed. If client side validation is being used but not server side the attacker can modify the client side validation and still execute this attack.</p>
Canonicalization:	Language based	Keep the web server patched and up to date because is the first step. The second step is validating input.
Null-String Attacks:	Language based	Web applications are written in high-level languages like ASP, JSP, PHP but use libraries of pre-written code often written in C/C++ which is a low-level language. When there are different ways of handling null characters between programming languages there can be risks in the data validation mechanisms.
SQL Injections using Stored Procedures	Server side	Don't use admin accounts to trigger stored procedures. Use accounts specifically set up for this purpose that do not have permissions to drop or create tables.
Command Injection	Server side	The primary prevention method here again is input validation on the server. Never rely on client side validation because it can be altered.
Fingerprinting the Server	Server side	Keep track on any vulnerabilities that are discovered on the systems you own. Attackers know when new exploits are published. You must also be aware. BugTraq is a good source for monitoring new exploits: http://www.securityfocus.com/
Denial of Service	Server side	This is difficult to protect against. Clustering and load balancing can help to reduce the risk.

Attack	Type	How to Protect:
Fake Cryptography	Authentication	Developers should use algorithms that are acknowledge as secure such as RSA, Triple DES, and AES, and avoid using any home grown or bleeding edge algorithms because they are easier for attackers to exploit.
Breaking Authentication	Authentication	Make sure all authentication occurs on secure connections. Use complex passwords with mixed case, special characters and numbers. Enforce locking of accounts after a number of failed attempts.
Cross-Site Tracing	Authentication	Turn off TRACE HTTP method on all web servers so attackers cannot obtain authentication tokens.
Forcing Weak Cryptography	Authentication	Make sure the server use SSL, remove weak ciphers

Table 3: (Whitaker, 2006)

Security at the enterprise level should include a unified approach with the Network team, Operations, DBA's, QA, Development and the internal and external auditors. A security awareness should be present at all levels of the software development lifecycle, from development to testing and what is currently in production.

Automated Safeguards

The easiest way to check whether a web site and applications have vulnerabilities, is to use a Web Vulnerability Scanner. A Web Vulnerability Scanner searches your entire website and automatically checks for vulnerabilities and develops reports prioritizing the risks, possible causes, and recommended fixes.

Any externally facing Web presence or intranet that stores sensitive data should have a comprehensive Web security program. ACME already protects the network perimeter with intrusion detection systems and firewalls, but firewalls have to keep Ports 80 and 443 (SSL) open in order to facilitate online business. These ports are a common entry point to hackers who have figured out ways to penetrate Web applications.

Using automated tools requires highly trained staff to configure the tool properly and limit the number of false positives.

Automated Web Vulnerability Scanning Vendors

The top two vendors for web vulnerability scanning are listed below:

- Watchfire - <http://www.watchfire.com/>
- SPI Dynamics - <http://www.spidynamics.com/>

Risk Reduction Recommendations

There are some key initiatives that must be established and maintained in the areas of people, process and technology.

People –those responsible for the development and deployment of web applications understand the fundamentals of both secure design principles and security threats.

Technology – The use of automated web application security tools is beneficial in scale and cost. An automated tool allows consistent, reliable and scalable examination of web application security vulnerabilities across large diverse environments. In addition, an automated tool can be used to

provide consistent and relevant recommendations that are consistent with corporate policy and requirements.

Process – Integrating web application security testing into the software development lifecycle is a key initiative for establishing good risk management. While this can and should be performed by a dedicated and knowledgeable security assessment team as part of the final review, it should also be integrated into the early stages of application development to focus on security issues as they appear and to save on time and cost.

Without the proper measurement and visibility of these initiatives, it is impossible to determine if adequate protections have been implemented to mitigate the risk. These metrics must include key components such as threats, vulnerabilities, remediation tasks and criticality, while establishing baselines and history over a period of time.

Conclusion

As the size and complexity of web applications increase, we must increase the defense mechanisms and preventive measures associated with them. Organizations must implement policies and controls that span entire distributed environments, meet the complexities of the latest technologies and that map to the software development lifecycle within the organization. A layered approach is suggested for maximum coverage. Additionally, management must be ready to meet their board, regulators and their customers with enterprise metrics and information regarding current security planning and posture.

Annotated Bibliography

ComputerWeekly, (2004) Top Five Threats,. Retrieved October 11, 2006 from <http://www.computerweekly.com/Article129980.htm>

This article describes how to use a layered security approach when implementing preventative security measures and the network and application levels.

Hutcheson, M. (2006). System and Application Testing for Security Vulnerabilities International Institute for Software Testing

I received this technical manual while attending a workshop for Quality Assurance security testing. This reference displays an anatomy of hacker attacks and discuss countermeasures that can be taken at both network and application layers.

Jaikumar, V. (2006) IT execs on firing line over security breaches The most recent incident involved AOL's CIO Retrieved October 11, 2006, from <http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9002748>

This article discusses how jobs are at stake when security is breached. This article can be used to argue the need to proactively invest in security prevention instead of reacting to exploits.

Jaikumar, V. (2006) Ohio University CIO warns more breaches may be found School conducts sweeping security audit Retrieved October 11, 2006, from http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9000439&source=rss_news82

This article discusses how one university's security was compromised to poor IT maintenance. Not properly applying patches and upgrades to all servers may leave holes for attackers.

Meier, J. D., Mackman, A., Dunner, M., Vasireddy, S., Escamilla, R., & Murukan, A. (2003). Improving Web Application Security: Threats and Countermeasures. Retrieved April 28, 2006, from <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/ThreatCounter.asp>

This article describes design patterns and practices for secure ASP.NET application but many of the concepts apply to any software systems. Many weaknesses are discussed at various layers and countermeasures are recommended.

OWASP. (2004). The Ten Most Critical Web Application Security Vulnerabilities. Retrieved May 28, 2006, from <http://www.owasp.org/documentation/top10.html>

The Open Web Application Security Project (OWASP) is dedicated to finding and fighting the causes of insecure software. This site offers a large amount of useful statistics for hacker attacks, prevention suggestions, and a collaborative environment with other security professionals.

Pudipeddi, H., (2005). Improving Software Security. Retrieved April 23, 2006, from <http://stickyminds.com/sitewide.asp?function=search&kind=simple&tt=SRCHBOX&freetext=improving+software+security>

Whittaker, J. A. and H. H. Thompson (2006). How to break web software security : functional and security testing of web applications and web services. Boston, Pearson/Addison Wesley.

This book offers concise information for developers and testers on how hackers use vulnerabilities in client side application code to gain root access to the server and steal information from the database.

Privacy Rights Clearinghouse A Chronology of Data Breaches (2006). Retrieved October 28, 2006, from <http://www.privacyrights.org/ar/ChronDataBreaches.htm>

The data breaches published have been reported because the personal information compromised includes data elements useful to identity thieves, such as Social Security numbers, account numbers, and driver's license numbers.

Stasiukonis, S., (2006). How Identity Theft Works. Dark Reading. Retrieved October 31, from http://www.darkreading.com/document.asp?doc_id=102595&print=true

This is a great article that discusses how one company was hired to penetrate a university to determine a baseline in their security. Social engineering is a main component in how this company was able to manipulate information out of unsuspecting employees.

Verry, J., Hired to Hack an ERP System. Tech Republic. Retrieved October 25, 2006 from <http://articles.techrepublic.com.com/5100-1009-6128952.html>

Another good example of how penetration testers were hired to expose security vulnerabilities. A holistic approach is suggested when assessing security prevention which helps to argue that security goes beyond the network layer and web application security needs diligent attention as well.

Project References

Below is a list of the references that were used in preparation of this document.

ComputerWeekly, (2004) Top Five Threats,. Retrieved October 11, 2006 from <http://www.computerweekly.com/Article129980.htm>

Hutcheson, M. (2006). System and Application Testing for Security Vulnerabilities International Institute for Software Testing

Jaikumar, V. (2006) IT execs on firing line over security breaches The most recent incident involved AOL's CIO Retrieved October 11, 2006, from <http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9002748>

Jaikumar, V. (2006) Ohio University CIO warns more breaches may be found School conducts sweeping security audit Retrieved October 11, 2006, from http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9000439&source=rss_news82

Meier, J. D., Mackman, A., Dunner, M., Vasireddy, S., Escamilla, R., & Murukan, A. (2003). Improving Web Application Security: Threats and Countermeasures. Retrieved April 28, 2006, from <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/ThreatCounter.asp>

OWASP. (2004). The Ten Most Critical Web Application Security Vulnerabilities. Retrieved May 28, 2006, from <http://www.owasp.org/documentation/topten.html>

Pudipeddi, H., (2005). Improving Software Security. Retrieved April 23, 2006, from <http://stickyminds.com/sitewide.asp?function=search&kind=simplesite&tt=SRCHBOX&freetext=improving+software+security>

Whittaker, J. A. and H. H. Thompson (2006). How to break web software security : functional and security testing of web applications and web services. Boston, Pearson/Addison Wesley.

Dornfest, R., Bausch, P., Calishain, T., (2006). Google Hacks, Third Edition Tips & Tools for Finding and Using the Worlds Information. Orielly Press.

Privacy Rights Clearinghouse A Chronology of Data Breaches (2006). Retrieved October 28, 2006, from <http://www.privacyrights.org/ar/ChronDataBreaches.htm>

The data breaches published have been reported because the personal information compromised includes data elements useful to identity thieves, such as Social Security numbers, account numbers, and driver's license numbers.

Stasiukonis, S., (2006). How Identity Theft Works. Dark Reading. Retrieved October 31, from

http://www.darkreading.com/document.asp?doc_id=102595&print=true

Verry, J., Hired to Hack an ERP System. Tech Republic. Retrieved October 25, 2006 from

http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9000439&source=NLT_SEC&nid=38