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
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

Mitigating Web Application Risks With A Security Code Review And Appscan.
Michael Blase
GSEC v1.4 Option 1
July 31, 2002

Abstract

Web application attacks over standard ports have been increasing at an alarming rate and often go undetected until after the attackers damage is done. Web applications, which operate on the application layer, and need to be accessible from the Internet, create different challenges than that of protecting the network or transport layers. These challenges include insuring that the web application code is written securely and that application type vulnerabilities are patched and corrected. A security code review of the web application is an effective way of insuring that new application code has been cleaned of those problems. The difficult part for information security has been to dedicate appropriate time to do a thorough and consistent job. There has been a recent introduction of web application vulnerability scanners that automate the task of this type of security scanning, while providing additional benefits of greater accuracy and detail.

This paper will focus on the security risks associated with web application code and ways of mitigating those risks with a security code review using the Appscan application vulnerability scanner. It will begin with an overview of the complexities of web application and code development; then the paper will discuss common web application code vulnerabilities along with some of the associated attacks. The Security code review process will be discussed with a focus on the benefits of using Appscan. Appscan will then be reviewed from the installation process to actually running a scan to give the reader a general idea of the ease and benefits of this tool. In conclusion this paper will demonstrate how a security code review using Appscan can be implemented in your company to help identify and mitigate the risks of web application code.

The challenges of securing the web application code

The web application is often a weak link in the security chain, allowing attackers to use the application to hack into your network. Web application attacks often makes hacking far too easy for the attacker. By exploiting poorly written code or known vulnerabilities, an attacker can gain entry without the time and effort needed for traditional network hacking. These types of attacks have been increasing at a startling rate.

> In a March 2002 Computer Security Institute report, 70 percent of organizations cited their Internet connections as a frequent source of attack compared to 59 percent in 2000. Gartner Inc. estimates that 75 percent of Web site hacks that occur today happen at the application level.[1]

[1] Bar_Gad , page 1

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

To help secure web application code, we need to understand some of the challenging issues.

Most network tools are ineffective in securing web applications.
This is because web applications typically operate on the HTTP, or HTTPS protocol. The ports used for these protocols, which are typically ports 80, and 443, need to be wide open to allow Internet access to the web site. This can also mean that they are wide open for hackers. Many companies have done a good job applying network security to the network with firewalls, intrusion detection systems, (IDS), and secure transport but need to understand that these tools are not effective for keeping attackers out of web applications that use these protocols. Firewalls need to be open for these protocols to allow traffic to the website. Although properly configured IDS can see attacks with known signatures, these web attacks are often unique with no known signatures. In addition, the possible use of HTTPS adds encryption to the process and often renders IDS tools ineffective.

Developers and the need for training in secure code writing.

Hundreds of vulnerabilities can exist when an application is not built from the ground up with a security mindset. Far too often, web applications are designed for functionality and rapid development instead of using a solid security model. Security is often an afterthought, considered after the applications are developed. Many of the insecure coding problems are due to a lack of knowledge and misunderstandings of how applications work with the Internet protocol. Education in this area is often not a priority. Additional problems often arise when companies contract out development without regard for security. Educating the developer staff with a security model of best practices goes a long way towards mitigating the risk. It is, however, usually an ongoing effort with some successes and some failures.

> Fundamental security best practices are often not observed. Authentication failures such as 'This user does not exist' or 'The password is incorrect', while helpful to site clients, also help attackers compromise the authentication process through automated processes. Unfortunately, the knowledge required to ensure that a web application has been developed securely is currently reserved to only a small proportion of developers and security professionals [2]

Whether it is sloppy programming or lack of knowledge, the end result is the same: potential disaster and liability for the company. Developers need to be kept up to date with security issues and educated with best practices for creating secure web applications. Security needs to be considered and implemented from the beginning of the development cycle. Company Policies need to be in place and enforced to insure that this is a priority.

[2] Ollman, page1

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

The Risks Of Using 3rd Party Software.

The compounding problems of deploying 3rd party software within your applications adds more concerns as this can often add additional vulnerabilities when the code is not fully understood. One small problem in the programming code or unchecked vulnerability can allow an attacker to gain entry into your network. All code in the application, including 3rd party software needs to be fully understood and reviewed for security best practices.

The complexity of the web application.

As web applications become more complex so do the number and types of vulnerabilities. New configurations of software add to the difficulty of securing web applications. There are a multitude of different types of web applications, different software and programming languages, making the learning process complex. The typical web application is made up of layers of code and data housed on a multiple servers. These servers include a web server, an application server, and a database server. A user will interface to the web site over the Internet by using a browser like Internet Explorer or Netscape Navigator. The browser connects and communicates to the web site using HTTP (Hypertext Transfer Protocol), or HTTPS (Hypertext Transfer Protocol Secure) for an encrypted link. The web server will host pages, programs, and scripts to the client and makes the connection to the application server. The web server will then connect to an application server and a database server as a trust relation is established. The application server may make connections to other systems, and databases behind the firewall. The database server processes data base queries. The complexity of making all these parts work together involves programming code at each of these layers. For any one person to know the entire application in detail can be an overwhelming challenge. The diagram below demonstrates the layers of the web application with each layer communicating with another. Poor coding, server misconfigurations or known vulnerabilities at any of these layers can allow an attacker to get into your system.

Figure 1: *Multi-layer Web Applications* [3]

[3] Sanctum, page 4

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

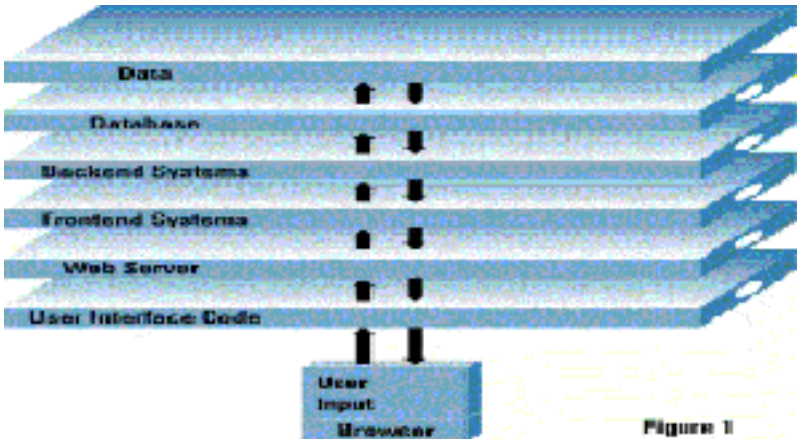


Figure 1

What are the vulnerabilities?

It seems like almost every other day we hear about another company, and occasionally a government agency, that has been attacked using known vulnerabilities or exploiting poorly written code web applications. These attacks are becoming much more serious in nature – beyond the malicious web defacement, to include stolen credit cards information, identity theft, changing prices in shopping carts, installing Trojan Horses, and bringing down sites. The problems can result from poor editing in the programming code, a misconfigured system or even testing code left in from the development team. The list goes on and on. Hackers are very aware of the vulnerabilities, which are published on web sites, along with instructions on how to exploit them. There are even script tools, which will make the process easier for the hacker by automating the process of discovering and exploiting the vulnerable areas.

For example: SQL Injection is a popular attack that can allow an attacker to gain access to the data on a SQL database through the web application. This is accomplished by an attacker manipulating the SQL instructions to gain access to the backend database. Typically a hacker will look for pages that present a Form page for passing data back to a site or searching for a "Form tag" in the HTML. The hacker would then try a number of known exploits with specific characters in the URL. The characters and the known exploits are posted on web sites and are usually specific to the configuration and type of SQL. If security best practices were not applied when the application was developed, the hacker could be in. To make things easier, malicious tools have been developed to automate the process of trying different characters in the URL based on the type of SQL that is being used by the application. Common successful attacks using this method have been seen with exposing sensitive information, identity theft, stolen credit card information and much more. In the case of SQL Injection, these attacks can often be avoided by applying needed patches, providing proper editing in the input parameters, limiting user access, and editing out the possible malicious characters.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

The following is a list from Sanctum Inc. describing common web application vulnerabilities. Appscan will scan for these vulnerabilities and provide recommendations to correct the problems

1. **Cookie Poisoning**—*Identity thef*t. By manipulating the information stored in a browser cookie, hackers assume the user's identity and have access to that user's information. Many web applications use cookies in order to save information (user-id,timestamp, etc.) on the client's machine. Since cookies are not always cryptographically secure, a hacker can modify them, thus fooling the application to change their values by "poisoning the cookie." Malicious users can gain access to accounts that are not their own and perform activities on behalf of that user.

2. **Hidden Field Manipulation**—*eShoplifting*. Hackers can easily change hidden fields in a page source code to manipulate the price of an item. These fields are often used to save information about the client's session, eliminating the need to maintain a complex database on the server side. Because e-Commerce applications use hidden fields to store the prices of their merchandise, Sanctum auditors were able to view the sites' source codes, find the hidden field and alter the prices. In a real word scenario, no one would have discovered the change and the company would have shipped the merchandise at the altered prices and may even have sent a rebate.

3. Parameter Tampering—*Fraud*. Changing information in a site's URL parameter. Because many applications fail to confirm the correctness of CGI parameters embedded inside a hyperlink, parameters can be easily altered to, for example, allow a credit card with a $500,000 limit, skip a site login screen, give access to alternate orders and customer information.

4. **Buffer Overflow**—*Closure of business*. By exploiting a flaw in a form to overload a server with excess information, hackers can often cause the server to crash and shut down the web site.

5. **Cross-Site Scripting**—*Hijacking/Breach of Trust*. When hackers inject malicious code into a site, the false scripts are executed in a context that appears to have originated from the targeted site, giving attackers full access to the document retrieved, and maybe even sending data contained in the page back to the attacker.

6. **Backdoor and Debug Options**—*Trespassing*. Often, programmers will leave in debug options in order to test the site before it goes live. Sometimes, in haste, they will forget to close the holes, giving hackers free access to sensitive information.

7. **Forceful Browsing**—*Breaking and Entering*. By subverting the application flow hackers access information and parts of the application that should normally be inaccessible, such as log files, administration facilities and application source code.
8. Stealth Commanding—*Concealing Weapons*. Often hackers conceal dangerous commands via a Trojan horse with the intent to run malicious or unauthorized code

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

that is damaging to the site.

9. 3rd Party Misconfiguration—*Debilitating a Site.* Since vulnerabilities are posted and patches made available on public web sites (such as www.securityfocus.com), hackers are alerted to new vulnerabilities as they arise. For example, through a configuration error a hacker could create a new database that renders the existing one unusable by the site.

10. Known vulnerabilities – *Taking control of the site*. Some technologies used in sites have inherent weaknesses that a persistent hacker can exploit. For example, Microsoft Active Server Page (ASP) technology can be exploited to gain the administrators' passwords and take control of the entire site.[4]

A security code review with Appscan, my personal experience.

The security code review is an effective way to audit and catch many of the vulnerabilities, closing down the problems that could give a hacker a way into your network. Until recently this had been a manual effort in our company. It required Information Security personal to have a good background in application development and was very time intensive. The average code review took days to complete. The extensive effort produced good but often-inconsistent results due to human errors and the time factors. The review involved understanding the application in detail, and then investigating the vulnerabilities associated with the code in the application. The effectiveness of this effort was often in direct relation to the time one was able to spend investigating security databases for these vulnerabilities and then testing the application. This included testing every input parameter in the application for lack of effective edits in the programming code. It was an extensive, somewhat inefficient effort but always produced positive security benefits. New web applications were being developed at a rate where our security staff was spending too much of their week on this task. We decided to look at the new breed of automated application scanners. After a good amount of research and testing we decided on Appscan due to its thoroughness, lack of false positives, and productive reports. The results we have received from this product have far outperformed our initial expectations. We are now doing a consistent and thorough job, delivering high quality assessments to our staff. This has cut down the time frame to complete a review from days to hours. The results of this product include a detailed report on how to correct the problems. Additional rescans of the application after the problems are corrected are now part of our review process.

Appscan Overview

Appscan is a product from Sanctum Inc. It's an automated security risk assessment scanner for web-based applications also referred to as application layer security. Appscan automates the complex, time-consuming tasks of manual code reviews. It checks for known and unknown vulnerabilities in a consistent and reliable way. Known

[4] Sanctum, page 1

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

vulnerabilities are usually vender related products, which require patching, while unknown vulnerabilities include misconfigured systems and application specific coding that does not conform to Appscan rules. Appscan crawls the site your scanning to discovers the all the links and records the responses as it navigates as a user would. It then creates a database of possible vulnerabilities based on the learning's of the crawl. It then tests those vulnerabilities using actual hacker techniques to insure that they are truly vulnerabilities and not false positives. Once the Test phase is complete, it produces a detailed report that includes a severity ranking and recommended fix(s). These reports are well designed with good explanations for the fix of the found vulnerabilities. The reports are easy to read and prioritized by how damaging they could be. They are organized well so they can be sent to the developers and system administrators to correct the problems with little additional explanation.
Appscan is a very powerful tool and can be configured to do both safe and unsafe attacks. In the unsafe mode, the denial of service and buffer overflow attacks can actually bring down the web site you are scanning. It is therefore essential that your company and the person(s) doing the scanning fully understand this. If you're working in a true development environment, this may be okay; otherwise it's advisable to run Appscan in a safe mode. In the safe mode Appscan will report that there is potential for this type of attack but will not do an actual attack. This option is accomplished by a configuration checkbox, which will be discussed later. In our company, Appscan is only run by information security personal that have an understanding of web applications and this product. When properly configured, false positives are kept to a minimum, but it is always advisable to have an experienced set of eyes review the results to insure the reported problems actually apply.

In the sections below I discuss the steps involved to set up and run an automatic scan using Appscan. I will then give a brief description of the results. Appscan has many options, filters and advanced features that this paper will not cover. The purpose here is to give the reader a general overview of some of the useful capabilities and to demonstrate how a product like this might be beneficial to other companies that create web applications to identify and mitigate the risks discussed. For more information, and more detailed instructions on this product, I have included the URL for additional Appscan information at the end of this document.

Appscan Installation:

The Appscan installation includes a computer based training, (CBT), course to give the user of this product a good understanding of the product and the importance of a correct configuration. Additional training can be sought through Sanctum. I have found that the CBT was thorough and provided enough information to get started. I would recommend that the scan first be applied to several test sites to fully understand its potentials. Above all, there is no substitute for a trained set of security eyes with a good understanding of web application code.

Appscan 3.0 was installed on a Windows 2000 client with SP2 and Internet Explorer 5.5. The CD installation has an install wizard that automates the quick installation. The

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

minimum recommended PC requirements are - 800Mhz and 256 Mbytes Ram. The hardrive requirements are at least 50MB. A 10/100 MBPX Nic card for a network connection is needed. Note that Appscan will not work with a Netscape browser.

Appscan is sold by license. When you purchase the product, you are asked to supply your Computer Host ID. This can be found by running the following: ipconfig /a from the command prompt. Each copy of Appscan can only be run on the machine it's installed on so you may want to consider purchasing multiple copies.

Sanctum only allows you to scan sites that you have been authorized for. The IP range that you must supply to Sanctum controls this. Anything outside of this range will not be scanable unless you formally request it.

Updates of vulnerabilities are available periodically and can be found, downloaded, and installed by going to Sanctums Customers Extranet site using the supplied username and password.

Running a scan with Appscan

Appscan is comprised of a setup section where you provide information that will configure and use the information for the scan cycle. The scan cycle is further broken up into 3 stages: Explore, Test and Report. The Explore stage crawls the site to be scanned to find potential vulnerabilities. The Test stage tests those potential vulnerabilities to see if they apply and classifies the severity. The Report stage produces reports that identify and suggest ways to correct the problems. These will be described in the following sections along with several screen prints for clarity so the reader can get a general understanding of this product.

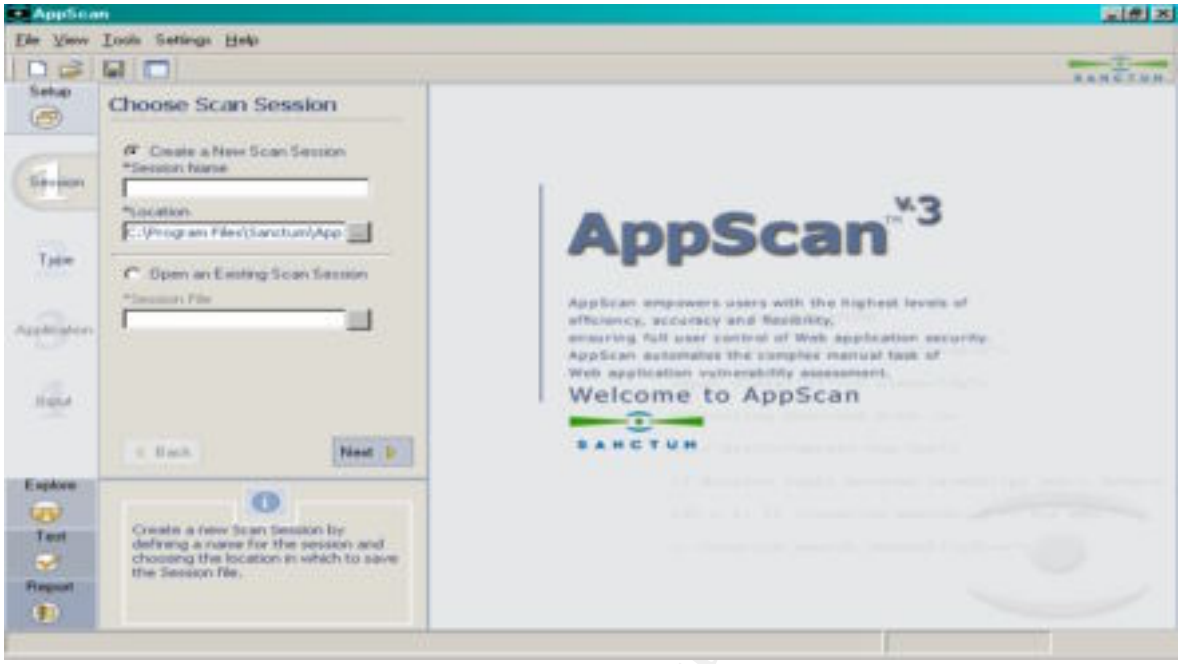
We will first fill in the setup information. The set up screens provides the different options that will be fed into the next 3 stages of the actual scan process.

The Setup stage, please refer to figure 2 below.
Here we provide information on whether this is to be a new scan or you if want to work with an existing scan. The selection of an existing scan allows you to rerun a previously run scan with the same or different options.

For our scan we will enter:
-Create a new scan.
-Specify a session name.
-Specify the location where the session will be saved. Saving the session allows you the ability to reopen the scan at a later time for review, rerun, or reprint.

Figure 2, the Setup screen.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



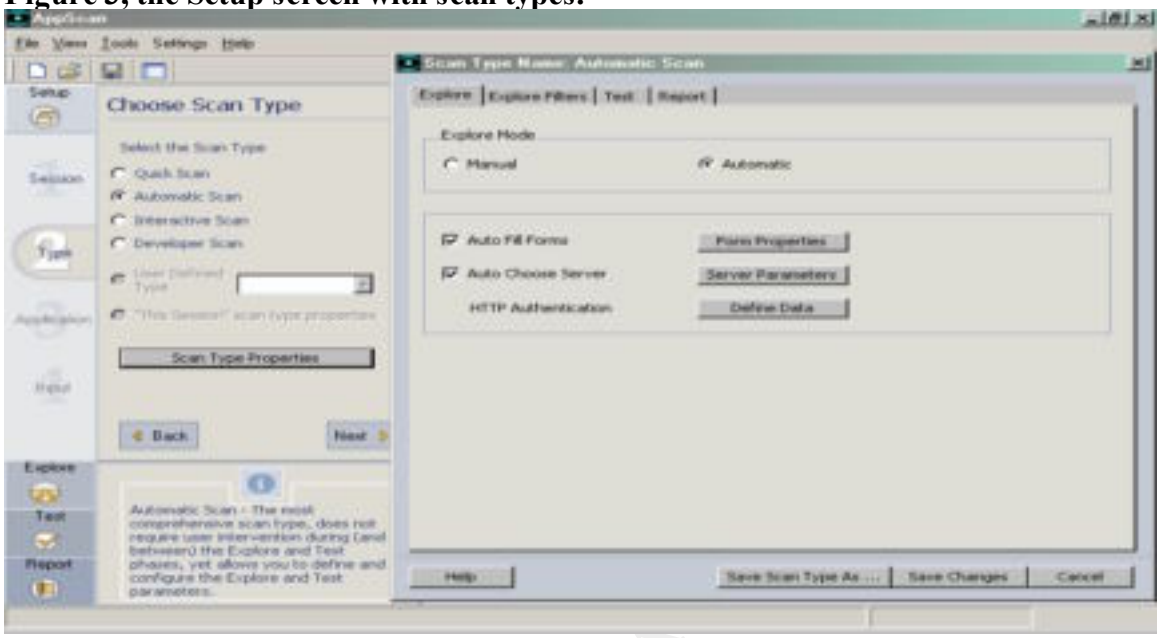
Next we determine the type of scan. Please refer to figure 3 below.
Appscan allows you to scan the following different ways.

Quick scan: This is an abbreviated Automatic Scan, which detects only common vulnerabilities. It is not as complete as the automatic scan described below, detecting only known vulnerabilities. No user intervention is required. Filters can be set to limit the scope of the scan.
Automatic Scan: This is a full Automatic Scan that is much more comprehensive than the quick scan described above. This scans for known and unknown vulnerabilities. Unknown vulnerabilities include misconfigured applications and application code and logic that do not conform to Appscan rules. No user intervention is needed. It allows you to define and configure parameters for a more focused, detailed scan. Filters can be set to limit the scope of the scan.
Interactive Scan: This is a very useful type of scan for a large site. This scans manually, allowing you to control what gets scanned in the site by your navigation, (mouse clicks), in the site. You can manually scan pages you select using the browser. This is good for emulating a path a user might take through a web site, limiting the scope of the scan. You can go back to automatic mode anytime.
Developer scan: This scan is designed for scanning application specific code for vulnerabilities. This is useful for scanning code that is under development.

In addition to the above 4 scans types, you can also create User Defined Scans, which allows you to modify one of the scan types to be more specific to your needs.
Choose Scan Type:
For this scan we will select an Automatic scan.
Clicking on Scan Type Properties, allows you to select the following 4 tabs:
Explore, Explore filters, Test and Report. These are described below.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Figure 3, the Setup screen with scan types.



The Explore tab*:*
This is used to choose an automatic or manual Explore mode, define any needed input parameters, input server information, and supply any HTTP authentication.
For this scan, we will designate the Explore stage to work in an automatic mode. This will instruct Appscan to do its own discovery of the site and any links. If we had chosen the manual Explore, Appscan would display the website and allow you to navigate through the site by you mouse clicks. Appscan would then only generate test requests for the path on the website that you clicked through.
By selecting *auto-fill*, any input has been pre defined and filled in under form properties. This is input that Appscan will fill-in on forms in this application. If this is not checked, Appscan will require you to fill in the information when needed.
By selecting *auto choose server*, we will let Appscan determine the server type.
If there is a user id and password required for this web site, they can be defined in the HTTP Authentication.

The Explore Filters tab*:*
Here you can define additional filters for the Explorer stage. These filters allow you to restrict Appscan from exploring links beyond what you specify. It also allows you to limit the depth and number of links. For this scan we have defined links depth of 5. This will help with performance and insure that we only scan what is needed.

The Test tab:
The Test tab asks for information that will be needed during the Test stage of the actual scanning process.
Group filter: Allows you to filter out types of tests for specific types of vulnerabilities. For this scan we will test all categories.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Send unsafe is a very important filter selection. When this is checked Appscan will execute tests that are considered damaging. These include denial of service tests and buffer overflow tests, both of which can bring down a site. It is *very* important to understand this and only use the unsafe feature in a development mode. For this scan we will include the unsafe attacks.

The Report tab*:*
Here you can change report filters on the report. Options include reporting only on certain types of vulnerabilities, and severity of vulnerabilities. For this scan we will take the defaults of reporting on all types of vulnerabilities. Additional changes to this configuration can be made during and after the actual reporting stage.

Setup – Define the application. Please refer to figure 4 below.
Type in the URL:
We now provide the starting point URL for what we want to scan. For security reasons, I do not show the actual sites URL for this scan. The show button next to the URL will test to make sure the site is available. It will display the site if it is available.

Additional Servers:
If this application used additional servers you can specify them below. For this scan, we will leave this blank.

Figure 4, define the application.

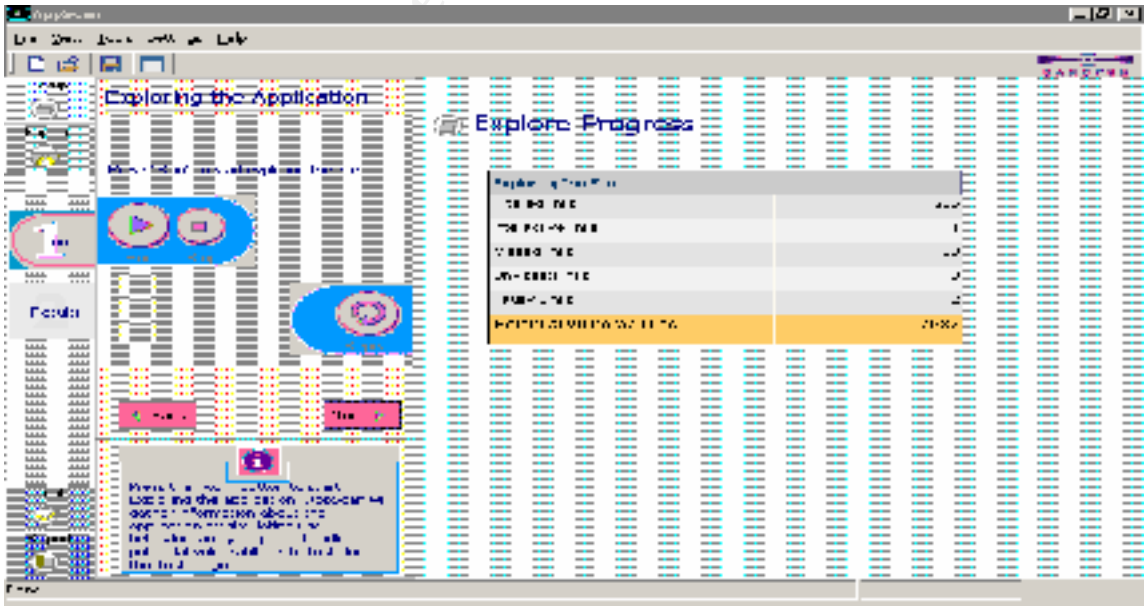
The Setup stage is now complete. We now are ready to start the actual 3-stage scan. These stages will use our setup information to run the scan. Appscan gives you the opportunity to change these settings during the scan stages as well.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

The Explore stage. Please refer to figure 5 below.
This is the 1st part of the 3-part scan. Explore will discover and map your site by crawling the site, and its links. This can be done automatically or manually or a combination of both depending on your selections. In a manual or interactive mode you would be in control of where you want Appscan to Explore by where you navigate in the site with your mouse clicks. This is useful when emulating a path the user may take. You can also switch to automatic from manual at any time. The automatic scan will visit all links or you can specify filters and how deep you want Appscan to Explore. As it explores, it creates a database of potential vulnerabilities based on the information it discovered from the links and responses to requests that it sends while discovering your site. This information is then used to produce a list of test attack requests for potential vulnerabilities that will be run in the Test stage. These vulnerabilities include known vulnerabilities and vulnerabilities associated with the specific application configuration.

By clicking on *Run*, the scan begins, using the information that was supplied in the setup stage. From the URL you supplied, the site and its links are scanned and a model of the site is built along with the database of potential vulnerabilities. We have instructed Appscan to explore automatically and to go to a links depth of 5. If we had chosen a manual Explore, we would control the Explore process by where we clicked, and the links we visited.
The results window on the right below summarizes the Explore progression. It shows a summary of the links and the potential vulnerabilities found. These will be tested in the next stage to insure that they are in deed real vulnerabilities.

Figure 5, the Explore stage.

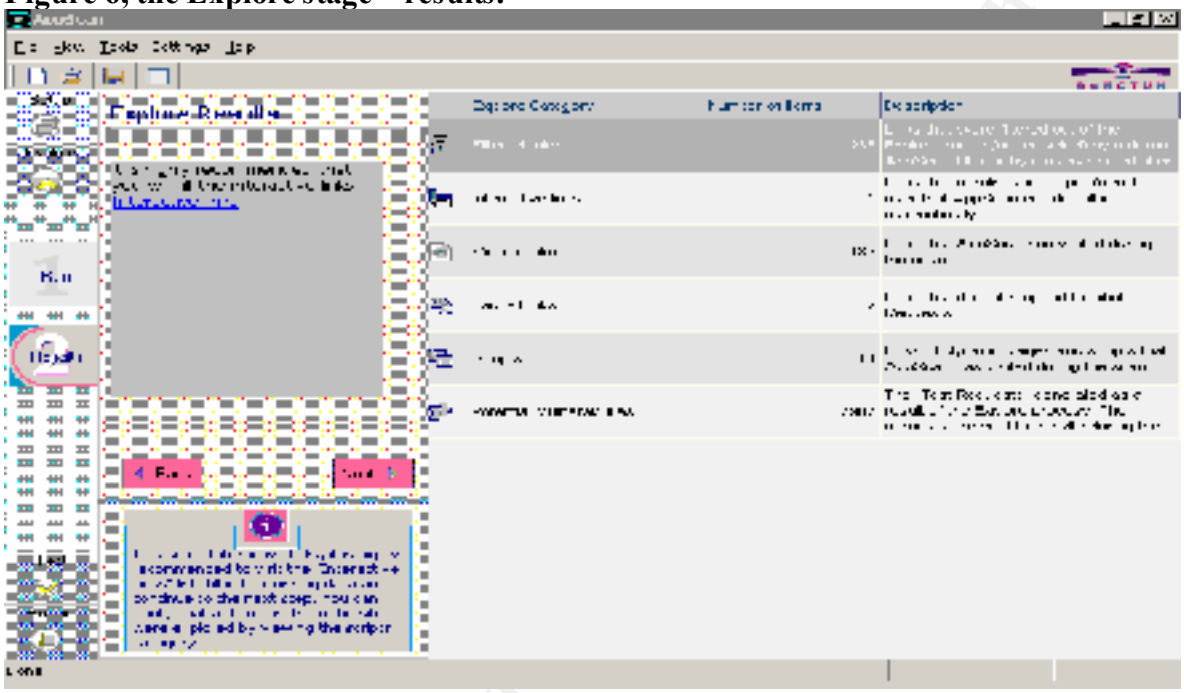


The Explore results, please refer to figure 6 below.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

We now have the opportunity to examine the Explore results and filtered links in detail and make any changes, if needed. Otherwise we proceed to the Test stage. Note that Appscan has identified 7987 test Requests for potential vulnerabilities. These are test requests that will be run in the Test stage to pinpoint actual vulnerabilities for this site.

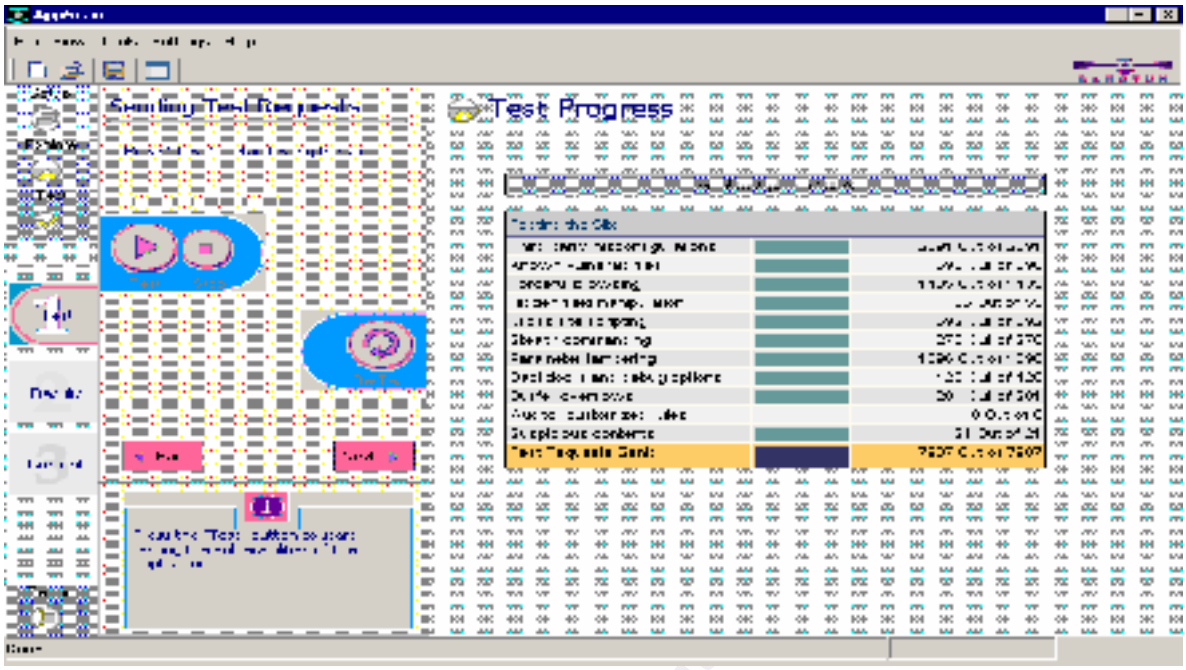
Figure 6, the Explore stage – results.



Test stage, please refer to figure 7 below.
The database of possible vulnerabilities discovered from the Explore stage are tested to make sure that they apply. Simulated attacks using hacker techniques are used to insure that there are no false positives reported. Appscan will then determine the level of severity and rank them for the next step. This stage can be set up for an automatic test or can be run with user intervention against a particular vulnerability. Manual retesting is also possible if desired for verification. The test progress shows you the status of the testing. In this case Appscan has run 7987 test requests that were identified in the Explore stage. These are broken down by the listed categories.

Figure 7, the Test stage.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



The Reporting stage. Please refer to figures 8 thru 10 below.
This is the final stage of the scan process. Reports are generated with the results of the Testing stage. Appscan allows you can generate 2 different types of reports. An Executive report and a Detailed report.
The Executive report is a high level report that summarizes the Detailed scan report. The Detailed scan report lists the vulnerability, the needed fix, and the advisory of this vulnerability. It further lists the test, which Appscan ran for this. You have the ability to go back and retest the vulnerability at any time. The reports are listed in order of severity. This is determined by the potential destructiveness of the vulnerability. The Detailed report also includes helpful links for additional information on vulnerabilities. Both reports allow you to customize a heading page to identify the scan and provide you with the ability to add comments.
The reports can be saved in many formats including word as well as exported in csv, or Crystal reports. For this scan we ran both an Executive and a Detailed report. The Executive report below describes the type and number of Vulnerabilities found at a high level. The Detailed report for this scan was extensive with cross site scripting, buffer overflows, and misconfiguration vulnerabilities.
I have included several pages of the Detailed report ,(figure 10), to give you a sample of the detail for identifying vulnerabilities and the helpful recommendations.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Figure 8, the Report stage- configuration.

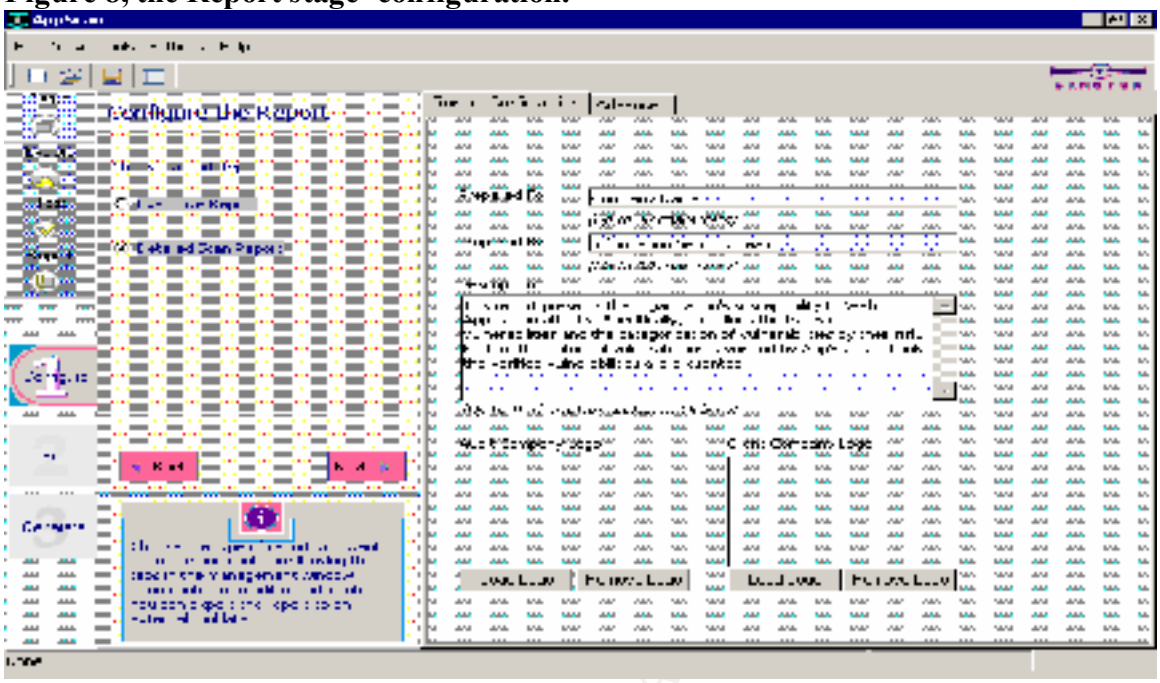
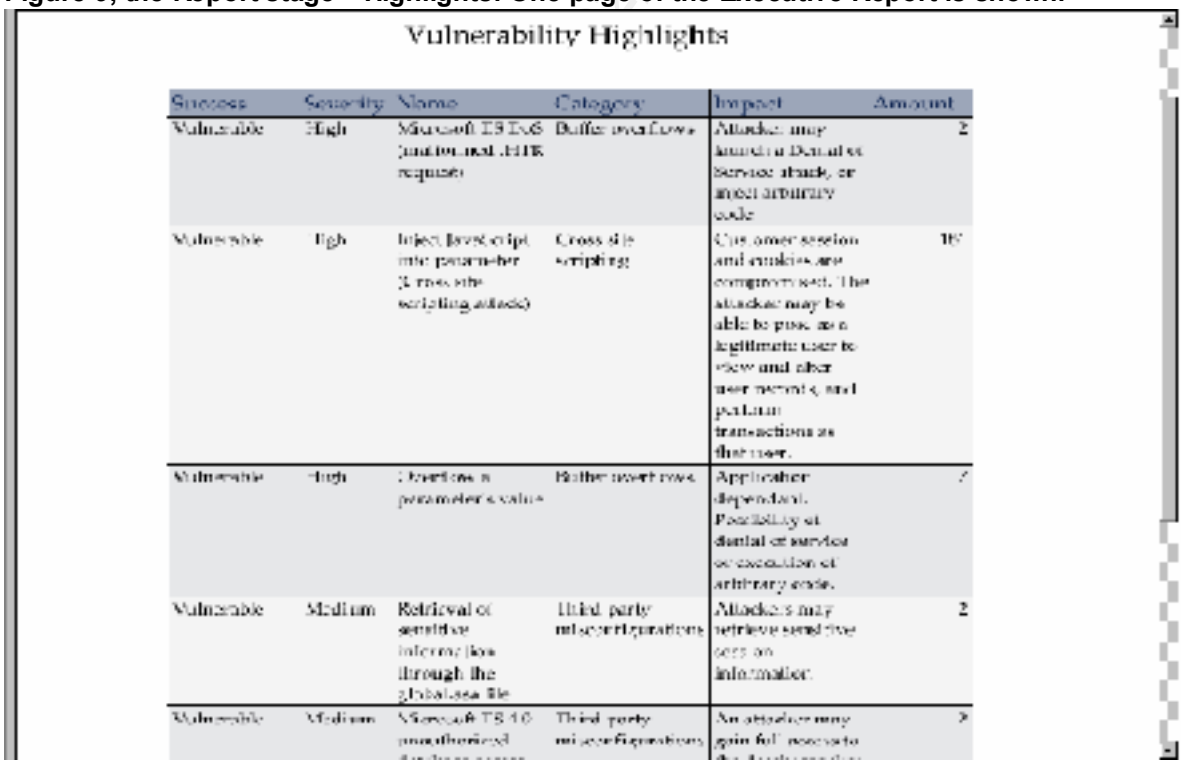


Figure 9, the Report stage – Highlights. One page of the Executive Report is shown.



Vulnerability Highlights

Success	Severity	Name	Category	Impact	Amount
Vulnerable	High	Microsoft IIS Ex6 [malformed .HTR request]	Buffer overflows	Attacker may launch a Denial of Service attack, or [illegible] arbitrary code	2
Vulnerable	High	Inject JavaScript into parameter [Cross site scripting attack]	Cross site scripting	Customer session and cookies are compromised. The attacker may be able to pose as a legitimate user to view and alter user records, and perform transactions as that user.	16
Vulnerable	High	Overflow a parameter's value	Buffer overflows	Application dependent. Possibility of denial of service or execution of arbitrary code.	7
Vulnerable	Medium	Retrieval of sensitive information through the globalasa file	Third party misconfigurations	Attackers may retrieve sensitive data on [illegible]	2
Vulnerable	Medium	Microsoft IIS 4.0 [malformed [illegible]	Third party misconfigurations	An attacker may gain full access to the [illegible]	2

Figure 10, a sample of the Detailed report below. Although one cross site scripting vulnerability is shown below, this report produced many different links with this problem, which are not shown here. The link information has been removed for security

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

reasons. Note that this report includes the recommended fix.

6 Client side execution of malicious scripts (cross-site scripting)

Full Explanation

Impact

Customer session and cookies are compromised. The attacker may be able to pose as a legitimate user to view and alter user records, and perform transactions as that user.

Affected Products

General

Technical Description

There are three parties involved in this attack:
(A) - is an attacker. He/she may know the identity of "B", and the structure of site "C".
(B) - is the victim user (of web-site "C").
(C) - is the vulnerable web-site.
The attack is basically a privacy violation. The attacker (A) gains the victim user (B)'s credentials at the vulnerable site (C). When the site involved is vulnerable, it is possible to steal credentials from its users. It is not possible to gain information regarding other sites, so (C)'s vulnerability affects only (C)'s customers.

The attack hinges on the fact that the web-site (C) has a script that returns user input (usually a parameter value, but variants are discussed below) in an HTML page without first sanitizing the input. This allows an input consisting of JavaScript code to be executed by the browser when the script returns this input in the response page. As a result it is possible to form links to the site (C) where one of the parameters consists of malicious JavaScript code. This code will be executed (by (B)'s browser) in (C) site context, granting it access to cookies (B) has for site (C), and other windows in site (C) at browser (B).
The attack proceeds as following: The attacker (A) lures the legitimate user (B) to click on a link that was produced by the attacker. When the user clicks on the link, this generates a request to the web-site (C) containing a parameter value with malicious JavaScript code. If the web-site (C) embeds this parameter value into the response HTML page (this is the essence of the site vulnerability), the malicious code will run in the user's browser (B).
Possible actions that can be performed by the script are:
[1] Sending the attacker the user cookies for the legitimate site
[2] Sending the attacker the current URLs of the legitimate site in which the user has an open window
This information is sent to the attacker (A), and thus the victim user "C"'s security (privacy) is compromised.

Some notes:
[1] Although the attacked web-site (C) is involved, it is not in itself compromised (in the narrow sense). It is only used as a jump station for the malicious script (sent by the attacker) to return to the victim's browser (B) as if it is legitimate.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

However, since the privacy of the victim (B) is breached in the context of site (C), and since site (C) is directly responsible, it is considered a security flaw in site (C) (much like a weak session token would have been).
[2] The malicious link can be provided by (A) by via a web-site link (if (A) maintains a site that is visited by (B)), or via email (if (A) knows (B)'s email address, and if (B)'s email client uses the browser to render the HTML message).
[3] While user input is most commonly found in form field values (i.e. URL parameters), there are known attacks where the malicious code is embedded in the path, or in the HTTP Referer headers, and even in cookies

Fix Recommendation

Sanitize user input and filter JavaScript code. We suggest that you filter the following characters: < > " ' % ;) (& +

References

CERT Advisory CA-2000-02
http://www.cert.org/advisories/CA-2000-02.html

Microsoft HOWTO: Prevent Cross-Site Scripting Security Issues (Q252985)
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q252985

Microsoft Technet "Cross-site Scripting Overview"
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/csoverv.asp

Affected Links	Inject JavaScript into parameter (Cross site scripting attack)	
No.1		
Success	Vulnerable	
Severity	High	
Original Link	//xxxx/xxxx_xxx.asp	**LINKS HAVE BEEN SANITIZED FOR
Link	//xxxx/xxxx_xxx.asp	SECURITY REASONS**

As you can see from the Detailed report sample above, Appscan displays the links that have the cross site scripting vulnerabilities. This report had over 100 links that had this problem. The recommended fix above advises what characters that the developer needs to filter, (edit), out. This report exposed many other problems including the vulnerabilities noted in the Executive report shown on Figure 9 above.

Conclusion:

Web applications that that need to be accessible over common ports, using the HTTP, and HTTPS protocols, will continue to have risks that attackers can exploit due to insecure code, misconfigurations, and known vulnerabilities. Network layer security products such as Firewalls and IDS are not effective to protect web applications, which operate on the application layer. The challenges that must be dealt with in securing web applications include education and training for the developers to insure that they understand and employ best practices for writing secure code. Companies need to prioritize the use of best practices by creating and enforcing policies. Implementing a security code review with Appscan is a very effective way to insure that the web applications are free of

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

insecure coding, misconfigurations and known vulnerabilities. Appscan is a valuable, easy to use tool that can add a tremendous benefit to companies in conjunction with a security code review. The benefits include thoroughness, consistency and timesavings in identifying the problems. Once identified, Appscan's suggested fix(s) instructs the staff how to correct the problems, helping to mitigate the risks of web applications and the application layer.

For more detailed information on Appscan:

http://www.sanctuminc.com/solutions/appscan/index.html

References:

Bar-Gad, Izhar, "Auditing and Securing Web-enabled Applications", Vol. #5, 15 April 2002, URL: http://www.theiia.org/itaudit/index.cfm?fuseaction=forum&fid=435 , (30 June 2002).

Ollmann, Gunter , "Bug Watch: Developers at fault", 27 June 2002, URL: http://www.vnunet.com/News/1133057 , (30, June, 2002).

Sanctum Security Adviser, Cyberterrorism:The New Enemy, 2002, URL: http://www.sanctuminc.com/news/advisory/index.html , (30, June 2002).

Sanctum white papers, "The Ten most Common Application Level Hacker Attacks" , URL: http://www.sanctuminc.com/solutions/whitepapers/index.html ,(July 2002).

Andress, Mandy, "Web apps are Trojan horses for hackers", InfoWorld, 5 April 2001, URL: http://www.infoworld.com/articles/tc/xml/01/04/09/010409tcwebsec.xml , (24 June 2002).

Pettet, Steve, , "Anatomy Of A Web Application: Security Considerations", July, 2001, URL: http://www.sanctuminc.com/solutions/appscan/details/index.html , (05 June 2002).

Hulme, George V., "Hackers Sneak Through Open Doors In Applications", InformationWeek, 25 Feb 2002, URL: http://www.informationweek.com/story/IWK20020221S0025 , (28 June 2002).

OWASP, The Open Web Application Security Project, "A Guide to Building Secure Web Applications and Web Services", (2002), URL:

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

http://www.owasp.org/guide/ , (12 June 2002).

Schneier, Bruce, "Secrets and Lies, Digital Security in a Networked World", New York,

Wiley Computer Publishing, 2000, 353-361.

Dyck, Timothy, "AppScan Rethinks Application Security", eWeek,18 March 2002, URL: http://www.eweek.com/article2/0,3959,35242,00.asp (14 June 2002).

Forristal, Jeff, "Maintaining Secure Web Applications", Network Computing, 20 March 2000, URL: http://www.networkcomputing.com/1105/1105ws1.html , (20 June 2002).

Peterson, Scott, and Fisher, Dennis, "Web apps pose security threat", January 2001,eWeek, URL: http://www.zdnet.com.au/newstech/security/story/0,2000024985,20150718,00.htm, (22 June 2002).

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.