SPI LABS
secure. protect. inspect.

Security at the Next Level

Are Your Web Applications Vulnerable?

Introduction

Application Security Is the Trend of the Future.

It began with desktop security when the only means of compromising your data was by inserting a contaminated floppy disk into your PC. That was the age of Anti-Virus. It evolved with the Internet as more corporations developed internal and external networks. That was the age of Network Security. Now as corporations leverage the power of the World Wide Web, information security has reached its 3rd age, the age of Application Security.

“One of the biggest vulnerabilities of a corporation’s network is the widespread access to its applications. To date, Internet security solutions have not been designed to handle perhaps the most crucial part of the transaction — that is, the application and its core data. To address the new requirements, we believe firms will need to implement vulnerability assessment programs and application security software. We believe that application security is a critical element in network security.”

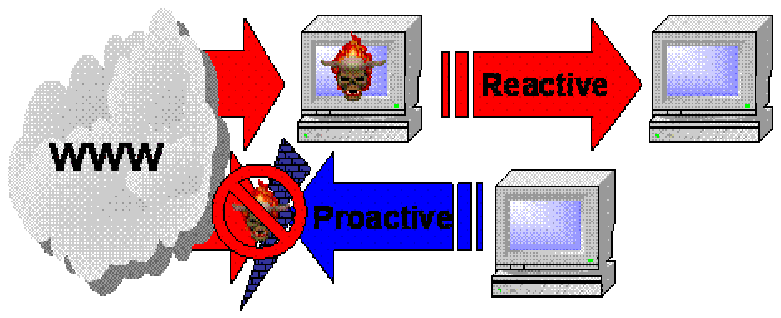
- Bear Stearns, “Internet Security”, June 2001

Web Applications can take many forms – an informational website, an e-commerce website, an extranet, an intranet, an exchange, a search engine, a transaction engine, an e-business. All of these applications are linked to computer systems that contain weaknesses that can pose risks to a company. Weaknesses exist in system architecture, system configuration, application design, implementation configuration, and operations. The risks include the possibility of incorrect calculations, damaged hardware and software, data accessed by unauthorized users, data theft or loss, misuse of the system, and disrupted business operations.

As the digital enterprise embraces the benefits of e-business, the use of Web based technology will continue to grow. Corporations today use the Web as a way to manage their customer relationships, enhance their supply chain operations, expand into new markets, and deploy new products and services to customers and employees. However, successfully implementing the powerful benefits of Web based technologies cannot be achieved without a consistent approach to **Web Application Security**.

Everyone gets hacked, from large consumer e-commerce sites and portals, such as Yahoo!, to government agencies, such as NASA and the CIA. In the past, the majority of security breaches occurred at the network layer of corporate systems. However, today hackers are manipulating web applications *inside* the corporate firewall, enabling them to access and sabotage corporate and customer data. Given even a tiny hole in a company's web-application code, an experienced intruder armed with only a web browser (and a little determination) can break into most commercial websites.

The problem is much greater than industry watchdogs realize. Many U.S. businesses do not even monitor online activities at the Web application level. This lack of security permits even attempted attacks to go unnoticed. It puts the company in a **reactive security** posture, as seen in figure 1, in which nothing gets fixed until after the situation occurs. Reactive security could mean sacrificing sensitive data as a catalyst for policy change.



Why isn’t the Web Environment secure?

As an ever-increasing number of businesses move to take advantage of the Internet, they discover that the web is not just a new market or distribution channel but that it is also a new operating environment. In this new environment conventional security measures are not only outdated but also frequently ineffective.

A new level of security breach has begun to occur through continuously open Internet ports (port 80 for general web traffic and port 443 for encrypted traffic). Because these ports are open to all incoming Internet traffic from the outside, they are gateways through which hackers access secure files and proprietary corporate and customer data. While rogue hackers make the news, there exists a much more likely threat in the form of online theft, terrorism, and espionage.

In addition to the vulnerabilities inherent in the new Internet operating environment, negligence also accounts for a portion of the risk to a company’s data. According to the *SANS Institute* there are **7 management errors that lead to computer security vulnerabilities**:

7 – Pretend the problem will go away.

6 – Authorize reactive, short-term fixes so problems re-emerge rapidly.

5 – Fail to realize how much money their information and organizational reputations are worth.

4 – Rely primarily on a firewall and IDS.

3 – Fail to deal with the operational aspects of security: make a few fixes and then not allow the follow through necessary to ensure the problems stay fixed.

2 – Fail to understand the relationship of information security to the business problem – they understand physical security but do not see the consequences of poor information security.

1 – Assign untrained people to maintain security and provide neither the training nor the time to make it possible to do the job.

Today the hackers are one step ahead of the enterprise.

While corporations rush to develop their security policies and implement even a basic security foundation, the professional hacker continues to find new ways to attack. Most hackers are using "out of the box" security holes to gain escalated privileges, or execute commands on a company's server. Simple mis-configurations of off-the-shelf Web applications leave gaping security vulnerabilities in an unsuspecting company's web site.

It's not a case of IF your site will be attacked, but of WHEN.

Attacks on web-connected servers are becoming more common. Everyday there is news of another major corporation whose security has been breached. For example, attackers stole credit card numbers from Western Union's site, and a computer hacker broke into a Walt Disney Company computer, stealing sensitive guest information. Not to mention the many cases of brand deterioration such as the time when Ford's pristine Web site was defaced. In each of these highly publicized incidents, attackers used security holes in web-based computer applications to access and steal proprietary data and sensitive information or to make changes to a corporate system.

Passwords are not enough.

Passwords are only as secure as the people using them. If you rely only on passwords to protect access to your data, then you are relying entirely on the people creating and using those passwords. Historically, the easiest way to penetrate a company's defenses is with inside knowledge. One seemingly unbelievable yet real-world example: if a hacker knows that "Bob" works for your company and if "Bob" has a family website where information such as the names and ages of his children can be found, odds are that a hacker may be able to determine Bob's password and be inside your company's defenses very quickly. Even if the integrity of your passwords remains intact, a hidden back-up file, an upload file, a form, or even an application running on your web server might allow hackers to identify or bypass your password security.

SSL and Data-encryption are not enough.

The SSL protocol and data encryption may protect your information during transmission, but when this information is used by your systems it must be in a readable form. If the applications that process data on your site store information for their own purposes, odds are that they do not store it in an encrypted format. It is surprisingly easy to retrieve data from many Web-based applications, and if your site is vulnerable, then so is your data.

Firewalls are not enough.

When medieval architects designed castles, they spent more time on the gates and the moat than on any other single feature. They knew that any defensive system is only as strong as its weakest point. To be useful to the people with legitimate business inside, walls must have openings to the outside. Those openings provide potential vulnerabilities. The challenge is to make sure that you only lower your drawbridge to friendly forces. One of the hardest attacks to recognize and defend against is one that uses your own programs and systems against you. This Trojan-horse type of attack manipulates the features of your own software in order to force it to divulge information. Firewalls do not prevent this from happening.

Standard scanning programs are not enough.

Most scanners run a series of routine checks against your network searching only for known vulnerabilities on standard servers and applications. Most of these products evaluate your security based on a static list of potential vulnerabilities. All that they can tell you is that access to your data may be possible. They do not check website content to ensure the security of your entire Web presence – e.g., Web pages, Web scripts, proprietary applications, cookies, and other Web servers. As a result, they do not tell you what data a hacker might be able to access or what damage he could inflict upon your site and your company by going in through the Web application.

A chain is only as strong as its weakest link.

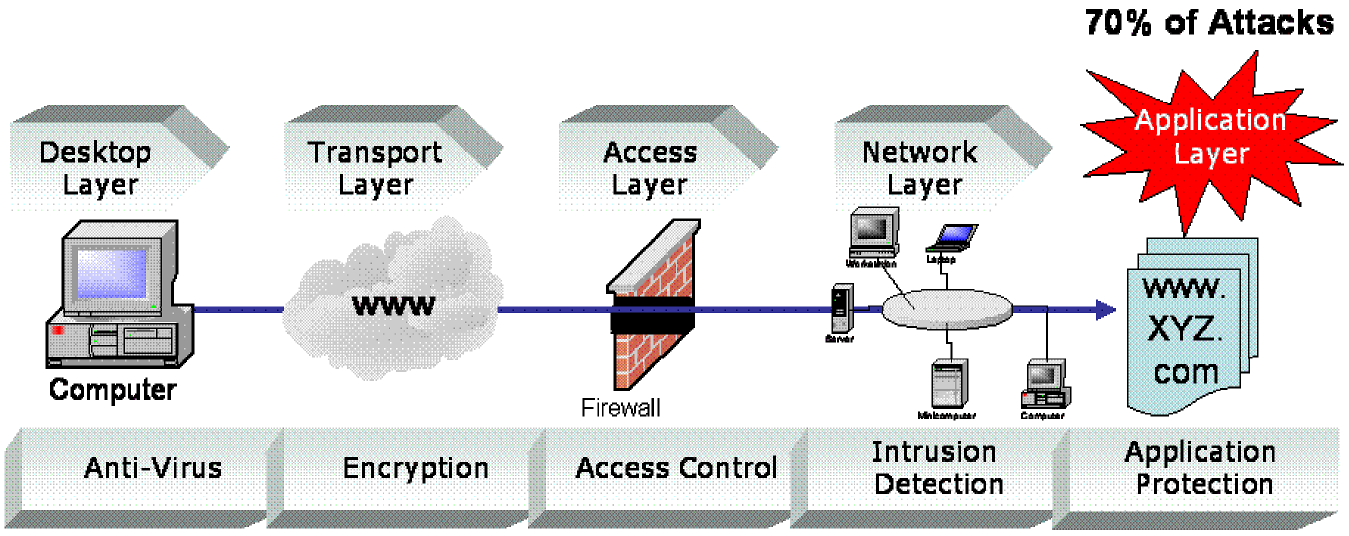
Even if you have the best internal controls, what about other links in your Internet supply chain - ISPs, ASPs, tool vendors, and development partners? Can you rely on them to take the same care with your data and information that you do? One small third-party example is software licensing agreements—users deploying any software product are generally required to sign a document stating that any damage that a third-party software product may cause is not the responsibility of the product vendor. In essence, third parties are telling you up front, “Security is your problem.”

It’s in the code.

Programmers typically don’t develop Web applications with security in mind. What’s more, most companies continue to outsource the majority of their website or Web application development using 3rd party development resources. Whether these development groups are individuals or consultancies the fact is that most programmers are focused on the “feature and function” side of the development plan and assume that security is embedded into the coding practices. However, these 3rd party development resources typically do not have even core security expertise. They also have certain objectives, such as rapid development schedules, that do not lend themselves to the security scrutiny required to implement a “safe solution”.

Manipulating a Web application is simple.
It is often relatively easy for a hacker to find and change hidden fields that indicate a product price. Using a similar technique a hacker can also change the parameters of a CGI script to search for a password file instead of a product price. If some components of a web application are not integrated and configured correctly, such as *search* functionality, the site could be subject to buffer-overflow attacks that could grant a hacker access to administrative pages. Today's web-application coding practices largely ignore some of the most basic security measures required to keep a company and its data safe from unauthorized access.

A firewall, an IDS, cryptography, and access control are just not enough. Figure 2 shows the progression of the professional hacker through the hacker's value chain. If the hacker appears to be a "normal user" he can pass all the regular security checks and end up engaged at the application layer. (A visitor to your company's public Web site appears like a "normal user".) Once he has reached the application layer, he begins his attack.



How do you protect your site?

Nowhere is the dynamic nature of the Web more apparent than in the area of security. The market is constantly introducing new software and new standards to the Web. Each of these innovations introduces potential weakness that hackers can exploit in order to compromise your network's integrity. In the rush to bring new software products to market few companies even test their products from a security perspective, yet users rely on these products to do business every day.

The cost of poor application security can be far greater than most organizations can imagine. Not only are you risking your brand and precious customer data but common denial of service attacks alone can prevent a company from doing business at all.

Even with the best conventional security systems available today, your company is very likely to be vulnerable to web-based application hacking.

What does your company need to do?

Developers and security professionals must be able to detect holes in both standard and proprietary applications. They can then evaluate the severity of the security holes and propose prioritized solutions, enabling your organization to protect existing applications and implement new software quickly. A typical process involves evaluating all applications on web-connected devices, examining each line of application logic for existing and potential security vulnerabilities.

Unfortunately, most security products cannot adequately examine the applications that reside on your web server, yet these applications often provide back-end access to confidential data. This means you must take a proactive approach to protecting your critical Web applications.

To successfully prevent an application attack you must first understand how a hacker thinks. We describe the anatomy of a Web application attack on the following pages.

Act 1: The Scan

The hacker starts by running a port scan to detect the open HTTP and HTTPS ports for each server and retrieving the default page from each open port.

Act 2: Information Gathering

The hacker then identifies the type of server running on each port, and each page is parsed to find normal links (HTML anchors). This enables the hacker to determine the structure of the site and the logic of the application.

Then the attacker analyzes the found pages and checks for comments and other possibly useful bits of data that could refer to files and directories that are not intended for public use.

Act 3: Testing

The hacker goes through a testing process for each of the application scripts or dynamic functions of the application, looking for development errors to enable him to gain further access into the application.

Act 4: Planning the Attack

When the hacker has identified every bit of information that can be gathered by passive (undetectable) means, he selects and deploys attacks. These attacks center on the information gained from the passive information gathering process.

Act 5: Launching the Attack

After all of these procedures, the hacker engages in open warfare by attacking each Web application that he identified as vulnerable during the initial review of your site.

The Result.

The hacker has been successful at manipulating your Web application. The results of the attack could be lost data, content manipulation, or even theft and loss of customers. The average corporation does not have the mechanisms to detect such attacks and can spend significant resources just trying to diagnose the implication of an attack.

The potential for loss is significant. A hacker could easily copy sensitive corporate information, such as proprietary customer databases or records, and disseminate that information to competitors, or even to the general public, without your knowledge.

What specific techniques can a hacker use to exploit your web application? There are a variety of techniques that a hacker can use to exploit your Web application. For example:

- **Parameter manipulation** can be something as simple as an invalid value passed to the web application in order to coax the application into revealing some internal data about itself, or it can be something as complex as passing a hidden SQL statement that could access useful data from your database.

- **A Forced Parameter** is an attempt to exploit the programming rather than the application, by attempting to determine debugging and testing flags. When these flags are present, they might be used to enable special, normally hidden modes within the application.

- **Cookie Tampering** involves manipulating the contents of cookies passed between the user and the Web application. This tampering can result in an application permitting access to an otherwise unauthorized user or the application may mishandle the contents of the cookie and return restricted data.

- **Common File Query** involves looking for files that have been inadvertently left accessible by developers, administrators, or default application configurations. The result could be the exposure of sensitive information that otherwise should have been removed from the application.

These are just a few examples of the types of attacks that the professional hacker may attempt.

Common Application Attack Methodologies. Attacks primarily fall into two types, Static Vulnerability Attacks and Dynamic Vulnerability Attacks. Static Vulnerability Attacks are commonly known attack methods, while Dynamic Vulnerability Attacks are harder to detect and protect against because they are launched from deep within the application logic. A more complete list of attacks and the details of their operation is provided below along with suggested remediation techniques.

- Static Vulnerability Attacks
    - Known Exploits
    - Directory Enumeration
    - Web Server Testing

- Dynamic Vulnerability Attacks
    - Link Traversal
    - Path Truncation
    - Session Hijacking
    - Hidden Web Paths
    - Java Applet reverse engineering
    - Backup Checking
    - Extension Checking
    - Parameter Passing
    - Common File Checks
    - Cross Site Scripting
    - SQL Injection

Static Vulnerability Attacks
These are attacks that are known by the hacker community at-large and are the result of poor application coding practices, sub par administration processes, or application mis-configurations.

Known Exploits.
Hackers are continuously posting the latest discovered Web application attacks on a variety of forums and "underground" Web sites that cater to the black-hat hacker community. The number of postings is in the thousands, with many more being discovered and posted on a daily basis. A hacker can use any of the known exploits to attack your Web application. For example, a hacker can use such exploits as:

- RDS Exploit
- Code Red Worm Exploit
- Nimda Virus

Remediation: Apply all general application patches if available. If not available a secure code audit and review is required in order to detect the specific application code level vulnerability.

Directory Enumeration. In this attack the hacker attempts to map your entire web-site hierarchy and directory structure by looking for common directory names that are hidden from public view but still left accessible. These directories can contain administrative pages or sensitive information that can help the attacker further his access into the application.

Remediation: Ensure that your Web root of the Web server is kept clean by removing hidden directories and that the server only contains content that needs to be viewed by the public. If hidden directories are needed insure that they are protected by proper authentication mechanisms.

Web Server Testing.

Many Web servers have vulnerabilities that allow attackers to submit malformed requests to the server. These can result in unauthorized access to the system.

> **Remediation:** If the server is known to have a patch make sure you have installed the latest updates.

Dynamic Vulnerability Attacks

This is where the majority of available security products stop being effective because they don't include these types of security checks. The majority of security products detect known vulnerabilities when they are in a certain location or within a fixed path (for example, the /cgi-bin directory). A Dynamic Vulnerability Attack will uncover vulnerabilities even if they are deep within your Web structure.

Example:
Standard fixed vulnerability products search for vulnerabilities within a fixed path (ex. /cgi-bin/formmail.pl).

But what if formmail.pl was, instead, located in /scripts/mail/formmail.pl?

A hacker will find formmail.pl —no matter where it is located in your web structure. The attacker will know that it is an exploitable script even though it is not in its default location.

> **Remediation:** A directory tree review is necessary to ensure that vulnerable files are not exposed through your Web application.

Link Traversal. A hacker will "crawl" your Web application in order to define the structure and logic flow of the application. This is an information gathering attack and is usually the initial step in a series of attacks. This enables a hacker to identify URL's that may no longer be in production but which are still referenced in commented-out sections of your Web application.

> **Remediation:** Analyze your link structure and ensure that any unnecessary links are removed from public access.

Path Truncation.
Another information gathering attack, the hacker will request only the directories found in a site and not the specific files. If a Web server does not have a default page located within a directory or specified in the Web server configuration, the result is that the contents of the directory are returned to the attacker. This enables the attacker to gain a significant amount of useful information about the application and its structure. Like the Link Traversal Attack, this is also used to mount additional attacks against the application.

Example:
A link exists on your site with /customers/id/993/details.html

The hacker will begin truncating the path looking for vulnerabilities during each truncation as follows:

First truncation: /customers/id/993/

Second truncation: /customers/id/

Third truncation: /customers/

This frequently uncovers vulnerabilities that neither the development staff nor the network administrators ever knew existed.

> **Remediation:** Ensure that there is a default file located within each directory and disable directory listings in the Web server configuration files.

Session Hijacking.
The hacker is able to "hijack" another user's session by intercepting or predicting any cookies sent by the site. This enables the hacker to impersonate an authenticated user's identity and review any and all information that the authenticated user would review.

> **Remediation:** This is typically a design issue within the application caused by the application generating predictable session ID's or cookies. Review application code and prohibit predictable session ID's and cookies.

Hidden Web Paths.

The hacker finds hidden paths or references in the source code or comments within a Web application. This information could provide access to restricted areas of your Web application.

Example: <!— my old path /webroot/old/bleh.asp —>

Remediation: Always keep HTML comments on a BETA server and remove them before migrating the application into production. Ensure that the HTML source code is free of any references or comments that relate to the design features of the Web application code.

Java Applet Reverse Engineering.

A common use for a Java Applet is client side log-in. The vulnerability exists because a Java Applet can easily be decompiled. This enables the hacker to find paths or links in Java Applet code and to use that information to try to obtain unauthorized information from your Web site.

Remediation: A Java Applet should not be used as the sole means of authentication or access control. Also, obfuscation should be used to make it difficult for an attacker to decompile the Applet.

Backup and Extension Checking.

The hacker works from a list of commonly used backup folder names and file extensions. If any of these folders exist on the site, the hacker will search for a mirror of the original site within that folder. If a mirrored site is found, the hacker will parse through the site, looking for valid information that could lead to a security breach.

Example:
Such directories as /backup and /oldsite could contain old revisions of your company's site.

Though these sites are no longer used, they could contain pages or scripts that may lead to information that could compromise your site.

Remediation: Delete all backup files from production servers and disable any mechanisms that automatically create backup files on a production server. If backups are required on the production server move the files out of the Web tree to prevent access by unauthorized users.

Parameter Passing.

Many vulnerabilities exist because application developers and site designers fail to consider how hostile users might input data into various Web page form fields. The hacker evaluates the parameters used by scripts and inputs invalid or user-specified values.

Example:
Given /contact.asp?email=bleh@bleh.com&subject=Send+me+stuff,

The hacker then requests many variations of this script as follows:
/contact.asp?email='&subject=
/contact.asp?email=|ls&subject=
/contact.asp?email=/../../../etc/passwd&subject=/../../../etc/passwd

This technique can also be used to check for buffer overflows in the application.

Remediation: Validate all input from the user on server side. Do not rely on client Java Script validation.

Common File Checks.

Many sites have common file names or common directory names within all their site directories. For instance: many administrators use WS-FTP to upload updated files to their web sites. However, WS-FTP leaves a file called WS_FTP.LOG in the directory to which the files were uploaded.

These common files provide information that attackers can use to compromise a site. A hacker searches through every directory for these common files and uses this information to attack your site.

Remediation: Remove these files from the production server Web tree.

Cross Site Scripting.

A hacker forces a Web server to serve Java Script that was not originated by the Web site authors. This malicious Java Script can be used to steal a user's cookies and even compromise a user's computer.

Remediation: Deny any HTML input to your Web application.

SQL Injection.

SQL injection is a technique for exploiting web applications that use client-supplied data in SQL queries without stripping illegal characters first. The hacker inputs SQL commands into Web page forms or parameters. The attacker may be able to run any SQL commands on your database which may lead to compromise of the database server. Despite being remarkably simple to protect against, there is an astonishing number of production systems connected to the Internet that are vulnerable to this type of attack.

Remediation: Parse and filter all input.

In the end, it's what you know.

The remedy for all corporate security issues cannot be described in just one paper. This document is meant to be a starting point to a better understanding of the critical issues that exist when your corporation engages the power of the Internet. Keeping current on all aspects of security is a must if you intend to stay ahead of the professional hacker and fully protect your critical data assets.

The sooner your company is able to detect, assess, and remediate your Web application security vulnerabilities the less likely your organization will become a victim of online fraud. A proactive approach will enable your organization to rest assured that its intellectual property is not at risk. In a nutshell, you must know what the enemy knows.

The Business Case for Application Security

Whether a security breach is made public or confined internally, the fact that a hacker has laid eyes on your sensitive data is of concern to your company, your shareholders, and most importantly, your customers.

SPI Dynamics has found that the majority of companies that take a proactive approach to application security, and that continuously engage in the application security process, are better protected. In the long run, these companies enjoy a higher R.O.I. on their e-business ventures.

About SPI Dynamics, Inc.

SPI Dynamics mission is to develop security products and services that systematically detect, prevent, and communicate Web application vulnerabilities and intrusions for any online business and provide intelligent methodological approaches for resolution of discovered vulnerabilities. SPI Dynamics products are used in a wide variety of industries, including financial management, manufacturing, healthcare, telecommunications, and government.

For further Information please contact:

SPI Dynamics, Inc.
115 Perimeter Center Place N.E.
Atlanta, GA 30346
Toll-free: 1-866-SPI-2700
Direct: 678-781-4800
Fax: 678-781-4850
sales@spidynamics.com