VULNERABILITIES IN WEB APPLICATIONS
PAINFUL LESSONS; ADVICE (UN)HEEDED?

Raymond Ankobia
Mathematics Department, Royal Holloway, University of London
E-mail: r.k.ankobia@rhul.ac.uk, rkya@aol.com

Abstract

The Internet has made the world smaller. In our routine usage we tend to overlook that "www" really does mean "world wide web" making virtually instant global communication possible. It has altered the rules of marketing and retailing. An imaginative website can give the small company as much impact and exposure as its much larger competitors. In the electronics, books, travel and banking sectors long established retail chains are increasingly under pressure from e-retailers. All this, however, has come at a price – ever more inventive and potentially damaging cyber crime. This paper aims to raise awareness by discussing common vulnerabilities and mistakes in web application development. It also considers mitigating factors, strategies and corrective measures.

Key words: Security, Internet, Application, Vulnerability, Risk, Standards, RFC, PKI, Countermeasures, SSL

1. INTRODUCTION

The Internet has become part and parcel of the corporate agenda. But does the risk of exposing information assets get sufficient management attention?

Extension of corporate portals for Business-to-Business (B2B) or developments of websites for Business-to-Customer (B2C) transactions have been largely successful. But the task of risk assessing vulnerabilities and the threats to corporate information assets is still avoided by many organisations. The desire to stay ahead of the competition while minimising cost by leveraging technology means the process is driven by pressure to achieve results. What suffers in the end is the application development cycle; - *this is achieved without security in mind*.

Section 1 of this paper introduces the world of e-business and sets the stage for further discussions. Section 2 looks at common vulnerabilities inherent in web application development. Section 3 considers countermeasures and strategies that will minimise, if not eradicate. some of the vulnerabilities. Sections 4 and 5 draw conclusions and look at current trends and future expectations.

1.1 Underlying Infrastructure

The TCP/IP protocol stack, the underlying technology is known for lack of security on many of its layers. Most applications written for use on the Internet use the application layer, traditionally using HTTP on port 80 on most web servers.

The HTTP protocol is stateless and does not provide freshness mechanisms for a session between a client and server; hence, many hackers take advantage of these inherent weaknesses. TCP/IP may be reliable in providing delivery of Internet packets, but it does not provide any guarantee of confidentiality, integrity and little identification.

As emphasised in [1], Internet packets may traverse several hosts between source and destination addresses. During its journey it can be intercepted by third parties, who may copy, alter or substitute them before final delivery. Failure to detect and prevent attacks in web applications is potentially catastrophic.

Attacks are loosely grouped into two types, passive and active. Passive attackers [6] engage in eavesdropping on, or monitoring of, transmissions. Active attacks involve some modification of the data stream or creation of false data streams [6].

2. COMMON VULNERABILITIES

This is by no means an exhaustive list but an indication of some serious flaws exploited by hackers. *Hacking Exposed: Web Applications (ISBN 007222438X*) as a good source for the subject area.

2.1 Buffer Overflow Attack

Usually perpetrated in a form of stack, heap or format string attack [3]. Without doubt, one of the oldest problems exposed by poor programming; yet attacks continue to be perpetrated on large scale, simply due to lack of rigorous security routines in web applications. To get the system to run their own code, attackers construct an input string sometimes with other malicious code that is long enough to overrun memory space assigned to it [7]. By doing so, this spills over and overwrites the stack below, overwriting what was initially in that address space. If the code contains malicious payload, it may subvert the system and escalate any privileges it may have garnered.

2.2 SQL Injection Attack

Most e-commerce web sites use dynamic content to attract and appeal to potential customers by displaying their wares using dynamic SQL queries and front-end scripts. An attacker could inject special characters and commands into a SQL database and modify the intended query. Chaining additional commands with intent of causing unexpected behavior could alter the meaning to a query. Not only could the attacker be able to read the entire database, but also in some circumstances, alter prices of these commodities.

2.3 Cross Site Scripting Attacks (XSS Attacks)

This attack is executed by embedding malicious message in an HTML form [4] [3] and posting it as a message to say a newsgroup or bulletin board. By viewing the message, the user unintentionally gets the code interpreted and executed by the web browser triggering its associated payload.

2.4 Input Validation Attack

Typically used by most active attackers to check for client side validation of fields and if successful then try to escalate privileges gained [3]. Poorly validated client-side (typically a web browser) allows an attacker to tamper with parameters sent to the server. Server-side may also compromised if trust is implicit and validation poorly executed from the client-side.

2.5 “Phishing” Attack

This attack is mainly executed due to vulnerability in some versions of web browsers. Attackers are able to create bogus websites and masquerade as legitimate commercial ones. They normally operate by sending spoofed emails to unsuspecting customers, advising them to visit their bank’s website to reactivate or update their accounts. The embedded addresses in these emails tend to have some hidden characters cleverly constructed to make the page appear to be a legitimate one.

On clicking the embedded website address, the unsuspecting user is redirected to a fake website where the credentials and details of bank accounts are taken and later used to empty the accounts.

[4] This anomaly is due to obfuscation techniques used by the URL to parse information. URL may be parsed in different ways using decimal, hexadecimal and dWord format. A particular vulnerability in Internet explorer allowed an attacker to construct and hide information by simply using the “@” symbol in ways that makes it possible to redirect traffic to bogus sites.

2.6 Mobile code

Most common languages used for developing mobile code include Java, ActiveX control and Shockwave. Traditionally the programme gets downloaded from a web server onto the customer’s machine. Environments used for execution include Virtual Machines (in browsers) or downloadable plug-ins. These programmes could be maliciously crafted to subvert the security and system functionality by causing crashes and disruption of normal operating environment.

2.7 Insecure Configuration Management

The communicating parties end points, especially their web servers, are poorly configured. Often ignored, but the area most attacked by hackers as a way of bypassing security offered by encryption and other security mechanisms [4]. Apache and IIS dominate commercial deployment of web servers and some of the earlier releases are riddled with bugs. Simply installing these applications with default settings is a bad practice. Poorly programmed sample scripts are exploited by attackers who may easily take control of the server resources.

2.8 Google Hacking

Google's search engine traverses the Internet, crawling websites, and taking snapshots of each web page it examines and caches its results. Next time a query is received, the search is performed on these cached pages, allowing for faster retrieval [4]. Hackers exploit these caches for vulnerable sites. The mechanism used by Google is explained in great depth in a white paper written by Foundstone (www.foundstone.com) called *SiteDigger,*. Tools such as these are the "Swiss army knives" of hackers. Using search engines, hackers find vulnerability scanning reports and intrusion detection alerts and log files. These are then used to find suitable targets to exploit.

3. STRATEGIES And COUNTERMEASURES

This section discusses remedial strategies and countermeasures (not in any order) that will alleviate threats and vulnerabilities commonly found in web application development.

3.1 Security Management Programmes

A security policy drafted and implemented from a holistic viewpoint with full approval of senior executives. There must be security education and awareness campaigns for the development team and administrators to foster a secure development lifecycle. Policies will ensure secure configuration of web servers and back end databases. Key amongst education campaigns is social engineering [8][7] where the attacker deceitfully extracts information directly from authorized people.

3.2 Deployment of Application Firewalls

This is a fairly new concept that offers use of gateways that specifically operate at the application layer. These are stateful, intelligent and content driven programmes/appliances that operate by checking web content. This allows for evaluation of attack signatures and exploits and prevents them from impacting on the targets.

They look out and allow legitimate requests of users to reach the backend servers and databases whilst preventing, logging and alerting administrators of malicious activities. Even though these may be able to do a far better job of analysing application content including graphics, they are not a panacea and the battle is far from over. Malicious and encrypted content will still get through firewalls [6].

3.3 Using SSL/TLS (HTTPS) Protocol

SSL/TLS has become the de-facto protocol for deploying secure web applications running on HTTP. It is based on Public Key Technology and X509 certificates, and defined by the Internet Engineering Task Force (IEFT) RFC 2246. This is supported in most web browsers and provides a secure tunnel between the client and the server. The server side almost always authenticates to the client by making available its public key to the client for verification; thereby offering a mechanism to identify rogue servers that impersonate by spoofing IP addresses with wrong DNS entries [8][7].

In most situations, the client side authentication is optional. This is due largely to the overhead of requiring every client to have a public key. This provides confidentiality, integrity and authenticity of transactions between both ends of the traffic. However, it must be emphasised that hackers concentrate on attacking the endpoints'; poor deployment and implementation of applications and databases make easy break-ins.

Poor implementation of a secure protocol does not make it any better. Attention to detailed instructions from these specifications is imperative to get it right.

3.4 Sandboxing and Code Signing

This idea for using sandboxes and signing of code (especially mobile code) is to introduce trust and assurance to the end user as to the origin of the application in question. Sandboxes are restricted and non-privileged operating environments [2][1]. Java Applets use this approach by encapsulating permissions and rights to resources within the programme itself. This provides a safer environment as the Java Virtual Machine (embedded in most browsers) consults the security manager for any violations or privileged system calls that may compromise the local computer. The author of a code may digitally sign it to give some authenticity and confidence to the end user; allowing that signature to be publicly verified using a certified public directory.

Authenticode is the approach by Microsoft for digitally signing code to provide trust and authenticity of origin. Developers of ActiveX controls/programmes may likewise sign the code to give similar level of trust and authenticity. However, discretion is left entirely to the user to check the authenticity of the digital signatures. [2] Clearly declares, *"A digital signature does not, however, provide any guarantee of benevolence or competence*". The Sandboxing (by Sun Microsystems) approach offers better assurance since it comes with a built-in security reference monitor that checks the access controls of the objects. These architectures are designed with Public Key Infrastructure (PKI) in mind and require education and awareness programmes on key management and certification authorities.

3.5 Use of Honey pots

These are used to lure potential crackers / hackers. The principle is one of falsifying information and placing it where hackers will eventually find it. The original concept seem to have come from [9] where he managed to bait hackers with falsified information which eventually led to their capture. This allows for the footprints of malicious activities to be logged, monitored and analysed. They help analyse the weak points that may are exposed with subsequent introduction of countermeasures that will seal any weaknesses that may be exploited. Use of this technology does have some legal implications. There is a debate as to whether this is enticement or entrapment and may require legal interpretation before use.

3.6 Using SiteDigger

This is a tool developed by Foundstone Professional Services to help web application developers and administrators test the efficacy of security measures incorporated during design. It works in conjunction with certain API's which will need to be downloaded from Google's website (http://www.google.com/apis/). This tool will help the web application developer or administrator to scan and generate reports of any leakages on a particular website.

3.7 ISO/IEC 17799 (Part I)

This was originally a British code of practice for Information Security Management and was later adopted by ISO as a Standard [5]. This has many facets for compliance and one of them is Systems Development and Maintenance. Part II of this, is for accreditation (currently being vetted by ISO for standardisation). It engages the certifying party through a rigorous compliance process, which includes the integration of controls and audit trails built into application systems. It encourages stringent checks and controls, Input data validation, message authentication to guard against unauthorised changes, output validation to ensure correct input and processing (the old adage "Garbage In, Garbage out), and the use of cryptographic controls to protect the confidentiality and integrity of information.

It also envisages strict and secure change control procedures and principle of least principle, by making sure that support developers are only given access to areas of their domain.

3.8 Security Audit

Self-Hack Audit [1]. The self-hack audit is an approach that uses methodology used by developers to identify and eliminate security weaknesses in an application before they are discovered and compromised. This will include checking login prompts, brute forcing passwords and setting up limits for login attempts.

Penetration Testing. Particular mention is made of The Open Web Application Security Project (OWASP), which is an Open source platform used as a benchmark for testing web application vulnerabilities. Below is a diagram (fig. 1) taken from their website

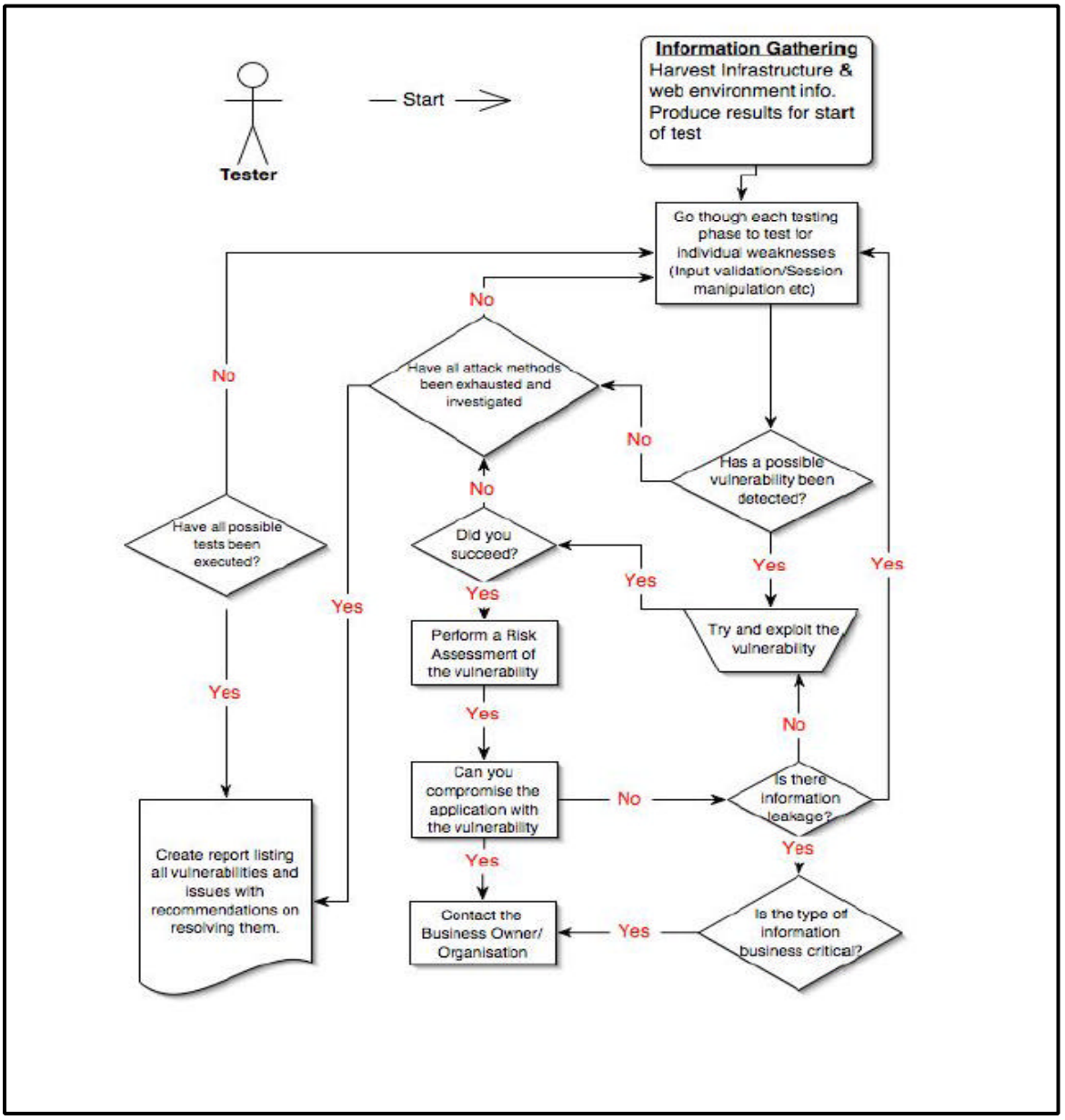

Fig 1. The OWASP Web Application Penetration Check

4. CONCLUSION

Internet transactions must reassure customers by maintaining confidentiality and integrity to the same extent as transactions using traditional manual procedures. Authentication mechanisms must ensure that customers are protected adequately so that their credentials are not compromised through false pretences.

Another key issue is that of non-repudiation. The procedures and mechanisms implemented for transactions over the web must make provisions for the protection of parties involved in a transaction. These can be achieved through use of Third Party services such creation of certificates. Currently, in most web transactions only the sever side (web servers) authenticate to the client (in this case, the customer using a browser). Implicitly, the use of credit cards is deemed enough to authenticate the customer. In highly secure transactions however, there may be the explicit use of certificates for both parties. This perhaps, will usher in the era of PKI with its key management challenges

5. THE FUTURE

The uptake of websites for e-business has now prompted a surge in digital content and media streaming services. Downloading MP3's is becoming the preferred choice for music lovers and the fight to protect such media content and piracy has only started. Movements such as DMCA are becoming more aggressive in pursing copyright violations on digital content.

E-commerce will continue to boom but so will attacks on infrastructure used in providing these services. Explosion of PKI and identity based crypto systems will provide mechanisms for ensuring confidentiality, integrity and availability of these services, which currently is being provided using traditional point of sale and accounting systems

REFERENCES

[1]. H. F. Tipton and Micki Krause, **Information Security Management Handbook,** Boca Raton, CRC Press LLC, 2003.

[2]. S Bosworth and M.E. Kabay, Computer **Security Handbook 4e,** Wiley & Sons, 2002

[3]. G Hoglund and Gary McGraw, **Exploiting Software: How to break code**, Addison Wesley, Pearson Education, 2004

[4]. Stuart McClure, et al **Hacking Exposed, network security secrets & solutions 4e,** McGraw-Hill / Osborne

[5]. ISO/IEC 17799 Part I – **Code of Practice for Information Security Management**, First Edition 2000-2001.

[6]. William Stallings, **Network Security Essentials, Applications and Standards 2e**, Prentice Hall, 2003

[7]. Dorothy E Denning, **Information Warfare and Security**, Addison Wesley, 1999

[8]. Ross Anderson, **Security Engineering**, Wiley, 2001

[9]. Cliff Stoll, **The Cuckoo's Egg**, Simon & Schuster Inc, 2000