THE TWELVE MOST COMMON APPLICATION-LEVEL HACK ATTACKS

By Steve Orrin, VP of Security and Technology

A whitepaper from Watchfire

TABLE OF CONTENTS

200 West Street | Waltham, Massachusetts | USA 02451 | T 1-781-810-1450 | F 1-781-890-2087
1 Hines Road, Suite 200 | Kanata, Ontario | Canada K2K 3C7 | T 1-613-599-3888 | F 1-613-599-4661
211 Piccadilly |London, United Kingdom | W1J 9HF | T +44 (0) 20 7917 2962 | F +44 (0) 20 7917 2963

www.watchfire.com

INTRODUCTION

While it is very difficult to secure applications on your website, application hacking is quite simple. A hacker typically spends a few hours getting to know the web application by thinking like a programmer and identifying the shortcuts he would have created had he built the application. Then using nothing more than the web browser, the hacker attempts to interact with the application and its surrounding infrastructure in malicious ways.

Audits performed by Sanctum, acquired by Watchfire on July 26, 2004, on over 500 leading websites containing over 2000 applications revealed that over 97% of the sites had *major* application level problems that could be exploited in only a few hours or less. In almost all of the cases, these companies had done a good job of implementing security at the network level (i.e. firewalls and encryption), but these sites were still highly vulnerable because they allowed hackers to access valuable customer and corporate information through the application layer.

THE 12 MOST COMMON ATTACKS

Based on our experience working with some of the leading enterprise customers from a wide range of regulated industries including financial services, government, and pharmaceuticals, we have recognized that the following are the most common types of attacks occurring on the Internet today:

Hack attack	What hackers use it for	How it's done
1. Cookie Poisoning	Identity theft/ Session Hijack	By manipulating the information stored in a browser cookie, hackers assume the user's identity and have access to that user's information. Many Web applications use cookies in order to save information (user-id, timestamp, etc.) on the client's machine. Since cookies are not always cryptographically secure, a hacker can modify them, thus fooling the application to change their values by "poisoning the cookie." Malicious users can gain access to accounts that are not their own and perform activities on behalf of that user.
2. Hidden Field Manipulation	eShoplifting	Hackers can easily change hidden fields in a page source code to manipulate the price of an item. These fields are often used to save information about the client's session, eliminating the need to maintain a complex database on the server side. Because e-Commerce applications use hidden fields to store the prices of their

Hack attack	What hackers use it for	How it's done
		merchandise, our auditors were able to view the sites' source codes, find the hidden field and alter the prices. In a real world scenario, no one would have discovered the change and the company would have shipped the merchandise at the altered prices and may even have sent a rebate.
3. Parameter Tampering	Fraud	Changing information in a site's URL parameters. Because many applications fail to confirm the correctness of CGI parameters embedded inside a hyperlink, parameters can be easily altered to, for example, allow a credit card with a $500,000 limit, skip a site login screen, and give access to alternate orders and customer information.
4. Buffer Overflow	Denial of Service/ Closure of business	By exploiting a flaw in a form to overload a server with excess information, hackers can often cause the server to crash and shut down the website.
5. Cross-Site Scripting	Hijacking/ Identity Theft	When hackers inject malicious code into a site, the false scripts are executed in a context that appears to have originated from the targeted site, giving attackers full access to the document retrieved, and maybe even sending data contained in the page back to the attacker.
6. Backdoor and Debug Options	Trespassing	Oftentimes, programmers will leave debug options in the code to test the site before it goes live. Sometimes, in haste, they will forget to close the holes, giving hackers free access to sensitive information.
7. Forceful Browsing	Breaking and Entering	By subverting the application flow, hackers can access information and parts of the application that should normally be inaccessible, such as log files, administration facilities and application source code.
8. HTTP Response Splitting	Phishing, Identity theft and eGraffiti	Hackers can poison web caches both at the site and on intermediate systems to change web pages in the cache and perform a variety of attacks against users with enhanced abilities to conceal their activities.
9. Stealth	Concealing Weapons	Often hackers conceal dangerous

Hack attack	What hackers use it for	How it's done
Commanding		commands via a Trojan horse with the intent to run malicious or unauthorized code that is damaging to the site.
10. 3rd Party Misconfiguration	Debilitating a Site	Since vulnerabilities are posted and patches made available on public websites (such as www.securityfocus.com), hackers are alerted to new vulnerabilities as they arise. For example, through a configuration error a hacker could create a new database that renders the existing one unusable by the site.
11. Known vulnerabilities	Taking control of the site	Some technologies used in sites have inherent weaknesses that a persistent hacker can exploit. For example, Microsoft Active Server Page (ASP) technology can be exploited to gain the administrators' passwords and take control of the entire site.
12. XML & Web Services Vulnerabilities	New layers of attack vectors and malicious use	New infrastructures and protocols supporting XML-based applications, both independent and integrated into existing web applications, pose significant business risk by introducing vulnerabilities into the infrastructure, protocols and content. In addition, new types of attacks can now take advantage of XML's flexibility and richness to affect all of the above-mentioned negative outcomes. Some of the vulnerabilities in XML include: entity expansion, SQL injection in XQuery, XPath injection, as well as various methods of denial of service attack.

CONCLUSION

Why do these vulnerabilities exist? The continuous cycle of auditing applications and trying to keep up with the latest patches is a constant battle against hackers who are armed with automated tools to scout out the newest vulnerabilities. While virtually all sites today attempt to achieve application-level security manually and ultimately fail, new automated tools have recently become available that allow auditors, developers and QA professionals to perform vulnerability assessments and ethical hacks that catch the vulnerabilities before the hackers do.

GUIDELINES TO DEVELOPING SECURE APPLICATIONS

With the explosion of web-enabled applications, a new reality has emerged. While you want users to come to your site and interact with the application, to protect your organization you need to be paranoid and build your site assuming the user will try to manipulate the application. Only information that is derived from the enterprise (i.e. the server) can be trusted and everything else (i.e. the client) is always potentially dangerous.

Therefore, the **first guideline** for developing a secure web application is: *Never trust any information that comes from the client, and never assume anything about it.* All security decisions must be made with the underlying assumption that anything that can theoretically be manipulated by the user will be manipulated in reality. Never assume that just because a user supposedly uses a specific tool, that this tool will put any constraints on his actions. For example, the fact that the browser does not show the hidden fields in the HTML of the page does not mean they cannot be seen by looking at the traffic, and manipulated when sent back to the server.

The **second guideline** is**:** *It is always easier to secure simple logic than complex logic.* This actually complements the first guideline as it relates to the usage of logic on the client. The use of client-side logic (such as scripts written in JavaScript or VBScript) might enhance the user experience with the application, but it has serious effects on the site's security. Using client-side logic as a security mechanism by checking information on the client is easily subverted since the mechanism can be bypassed. For example, having JavaScript verification of constraints on form parameters can easily be bypassed by directly creating the request outside the browser. In addition, any changes to the application logic or flow by changing links or parameter values, creates a complex program with multiple sources of flow instead of a single unified point or a simple flow structure.

RECOMMENDED READING

For more information about developing secure web applications to combat these top 12 hack attacks, please download our whitepaper, "Developing and Deploying Secure Web Applications", at https://www.watchfire.com/securearea/whitepapers.aspx?id=6

ABOUT WATCHFIRE

Watchfire helps organizations manage their online business by providing software solutions to minimize online risk and maximize channel effectiveness. These solutions let ebusiness, marketing, legal, compliance, and IT departments measure, manage, improve and secure their online channels. We analyze online content and applications and generate management intelligence dashboards that report on visitor behavior, quality, data privacy, web application security, and accessibility issues.

Watchfire recently acquired Sanctum, the pioneer in web application security testing and firewall software. Sanctum's web application security testing software, combined with Watchfire's Privacy solutions, help organizations institute comprehensive online risk management programs.

More than 400 enterprise organizations and Government agencies, including AXA Financial, SunTrust Banks Inc., Veteran's Affairs, United States Postal Service, and Dell rely on Watchfire to monitor, measure and manage all aspects of the online business including quality, privacy, web application security, accessibility, user experience and visitor behavior. Watchfire's alliance and technology partners include IBM Global Services, PricewaterhouseCoopers, TRUSTe, Microsoft, Interwoven, Documentum and Mercury Interactive. Founded in 1996, Watchfire is headquartered in Waltham, MA.

To evaluate Watchfire® AppScan™, please visit
https://www.watchfire.com/securearea/appscanauditdownload.aspx

