Web developers ignore security flaws at customers' peril

By Jon Brodkin, Network World, 10/04/07

Web security is at the top of customers’ minds after many well-publicized personal data breaches, but the people who actually build Web applications aren’t paying much attention to security, experts say.

“They’re totally ignoring it,” says IT consultant Joel Snyder. “When you go to your Web site design team, what you’re looking for is people who are creative and able to build these interesting Web sites… That’s No. 1, and No. 9 on the list would be that it’s a secure Web site.”

The biggest problem is designers aren’t building walls within Web applications to partition and validate data moving between parts of the system, he says.

Security is usually something that’s considered after a site is built rather than before it is designed, agrees Khalid Kark, senior analyst at Forrester.

“I’d say the majority of Web sites are hackable,” Kark says. “The crux of the problem is security isn’t thought of at the time of creating the application.”

That’s a big problem, and it’s one the nonprofit Open Web Application Security Project (OWASP) is trying to solve. An OWASP report called “The Ten Most Critical Web Application Security Vulnerabilities” was issued this year to raise awareness about the biggest security challenges facing Web developers.

Click to see: Top 10 vulnerabilities list
Web vulnerabilities

Experts say most Web applications can be hacked. Here are the top ten vulnerabilities that could put your Web site at risk.

1. Cross site scripting
2. Injection flaws
3. Malicious file execution
4. Insecure direct object reference
5. Cross site request forgery
6. Information leakage and improper error handling
7. Broken authentication and session management
8. Insecure cryptographic storage
9. Insecure communications
10. Failure to restrict URL access

SOURCE: OWASP (the Open Web Application Security Project)

The first version of the list was released in 2004, but OWASP Chairman Jeff Williams says Web security has barely improved. New technologies such as AJAX and Rich Internet Applications that make Web sites look better also create more attack surfaces, he says. Convincing businesses their Web sites are insecure is no easy task, though.

“It’s frustrating to me, because these flaws are so easy to find and so easy to exploit,” says Williams, who is also CEO and co-founder of Aspect Security. “It’s like missing a wall on a house.”

Here is a summary of OWASP’s top 10 Web vulnerabilities, including a description of each problem, real-world examples and how to fix the flaws.

1. Cross site scripting (XSS)

The problem: The “most prevalent and pernicious” Web application security vulnerability, XSS flaws happen when an application sends user data to a Web browser without first validating or encoding the content. This lets hackers execute malicious scripts in a browser, letting them hijack user sessions, deface Web sites, insert hostile content and conduct phishing and malware attacks.

Attacks are usually executed with JavaScript, letting hackers manipulate any aspect of a page. In a worst-case scenario, a hacker could steal information and impersonate a user on a bank’s Web site, according to Snyder.

Real-world example: PayPal was targeted last year when attackers redirected PayPal visitors to a page warning users their accounts had been compromised. Victims were redirected to a phishing site and prompted to enter PayPal login information, Social Security numbers and credit card details. PayPal said it closed the vulnerability in June 2006.
**How to protect users:** Use a whitelist to validate all incoming data, which rejects any data that’s not specified on the whitelist as being good. This approach is the opposite of blacklisting, which rejects only inputs known to be bad. Additionally, use appropriate encoding of all output data. “Validation allows the detection of attacks, and encoding prevents any successful script injection from running in the browser,” OWASP says.

**2. Injection flaws**

**The problem:** When user-supplied data is sent to interpreters as part of a command or query, hackers trick the interpreter — which interprets text-based commands — into executing unintended commands. “Injection flaws allow attackers to create, read, update, or delete any arbitrary data available to the application,” OWASP writes. “In the worst-case scenario, these flaws allow an attacker to completely compromise the application and the underlying systems, even bypassing deeply nested firewalled environments.”

**Real-world example:** Russian hackers broke into a Rhode Island government Web site to steal credit card data in January 2006. Hackers claimed the SQL injection attack stole 53,000 credit card numbers, while the hosting service provider claims it was only 4,113.

**How to protect users:** Avoid using interpreters if possible. “If you must invoke an interpreter, the key method to avoid injections is the use of safe APIs, such as strongly typed parameterized queries and object relational mapping libraries,” OWASP writes.

**3. Malicious file execution**

**The problem:** Hackers can perform remote code execution, remote installation of rootkits, or completely compromise a system. Any type of Web application is vulnerable if it accepts filenames or files from users. The vulnerability may be most common with PHP, a widely used scripting language for Web development.

**Real-world example:** A teenage programmer discovered in 2002 that Guess.com was vulnerable to attacks that could steal more than 200,000 customer records from the Guess database, including names, credit card numbers and expiration dates. Guess agreed to upgrade its information security the next year after being investigated by the Federal Trade Commission.

**How to protect users:** Don’t use input supplied by users in any filename for server-based resources, such as images and script inclusions. Set firewall rules to prevent new connections to external Web sites and internal systems.

**4. Insecure direct object reference**

**The problem:** Attackers manipulate direct object references to gain unauthorized access to other objects. It happens when URLs or form parameters contain references to objects such as files, directories, database records or keys.

Banking Web sites commonly use a customer account number as the primary key, and may expose account numbers in the Web interface.

“References to database keys are frequently exposed,” OWASP writes. “An attacker can attack these parameters simply by guessing or searching for another valid key. Often, these are sequential in nature.”
Real-world example: An Australian Taxation Office site was hacked in 2000 by a user who changed a tax ID present in a URL to access details on 17,000 companies. The hacker e-mailed the 17,000 businesses to notify them of the security breach.

How to protect users: Use an index, indirect reference map or another indirect method to avoid exposure of direct object references. If you can’t avoid direct references, authorize Web site visitors before using them.

5. Cross site request forgery

The problem: “Simple and devastating,” this attack takes control of victim’s browser when it is logged onto a Web site, and sends malicious requests to the Web application. Web sites are extremely vulnerable, partly because they tend to authorize requests based on session cookies or “remember me” functionality. Banks are potential targets.

“Ninety-nine percent of the applications on the Internet are susceptible to cross site request forgery,” Williams says. “Has there been an actual exploit where someone’s lost money? Probably the banks don’t even know. To the bank, all it looks like is a legitimate transaction from a logged-in user.”

Real-world example: A hacker known as Samy gained more than a million “friends” on MySpace.com with a worm in late 2005, automatically including the message “Samy is my hero” in thousands of MySpace pages. The attack itself may not have been that harmful, but it was said to demonstrate the power of combining cross site scripting with cross site request forgery. Another example that came to light one year ago exposed a Google vulnerability allowing outside sites to change a Google user’s language preferences.

How to protect users: Don’t rely on credentials or tokens automatically submitted by browsers. “The only solution is to use a custom token that the browser will not ‘remember,’” OWASP writes.

6. Information leakage and improper error handling

The problem: Error messages that applications generate and display to users are useful to hackers when they violate privacy or unintentionally leak information about the program’s configuration and internal workings.

“Web applications will often leak information about their internal state through detailed or debug error messages. Often, this information can be leveraged to launch or even automate more powerful attacks,” OWASP says.

Real-world example: Information leakage goes well beyond error handling, applying also to breaches occurring when confidential data is left in plain sight. The ChoicePoint debacle in early 2005 thus falls somewhere in this category. The records of 163,000 consumers were compromised after criminals pretending to be legitimate ChoicePoint customers sought details about individuals listed in the company’s database of personal information. ChoicePoint subsequently limited its sales of information products containing sensitive data.

How to protect users: Use a testing tool such as OWASP’S WebScarab Project to see what errors your application generates. “Applications that have not been tested in this way will almost certainly generate unexpected error output,” OWASP writes.

Another tip: disable or limit detailed error handling, and don’t display debug information to users.
7. Broken authentication and session management

The problem: User and administrative accounts can be hijacked when applications fail to protect credentials and session tokens from beginning to end. Watch out for privacy violations and the undermining of authorization and accountability controls.

“Flaws in the main authentication mechanism are not uncommon, but weaknesses are more often introduced through ancillary authentication functions such as logout, password management, timeout, remember me, secret question and account update,” OWASP writes.

Real-world example: Microsoft had to eliminate a vulnerability in Hotmail that could have let malicious JavaScript programmers steal user passwords in 2002. Revealed by a networking products reseller, the flaw was vulnerable to e-mails containing Trojans that altered the Hotmail user interface, forcing users to repeatedly reenter their passwords and unwittingly send them to hackers.

How to protect users: Communication and credential storage has to be secure. The SSL protocol for transmitting private documents should be the only option for authenticated parts of the application, and credentials should be stored in hashed or encrypted form.

Another tip: get rid of custom cookies used for authentication or session management.

8. Insecure cryptographic storage

The problem: Many Web developers fail to encrypt sensitive data in storage, even though cryptography is a key part of most Web applications. Even when encryption is present, it’s often poorly designed, using inappropriate ciphers.

“These flaws can lead to disclosure of sensitive data and compliance violations,” OWASP writes.

Real-world example: The TJX data breach that exposed 45.7 million credit and debit card numbers. A Canadian government investigation faulted TJX for failing to upgrade its data encryption system before it was targeted by electronic eavesdropping starting in July 2005.

How to protect users: Don’t invent your own cryptographic algorithms. “Only use approved public algorithms such as AES, RSA public key cryptography, and SHA-256 or better for hashing,” OWASP advises.

Furthermore, generate keys offline, and never transmit private keys over insecure channels.

It’s pretty common to store credit card numbers these days, but with a Payment Card Industry Data Security Standard compliance deadline coming next year, OWASP says it’s easier to stop storing the numbers altogether.

9. Insecure communications

The problem: Similar to No. 8, this is a failure to encrypt network traffic when it’s necessary to protect sensitive communications. Attackers can access unprotected conversations, including transmissions of credentials and sensitive information. For this reason, PCI standards require encryption of credit card information transmitted over the Internet.
Real-world example: TJX again. Investigators believe hackers used a telescope-shaped antenna and laptop computer to steal data exchanged wirelessly between portable price-checking devices, cash registers and store computers, the Wall Street Journal reported.

“The $17.4-billion retailer's wireless network had less security than many people have on their home networks,” the Journal wrote. TJX was using the WEP encoding system, rather than the more robust WPA.

How to protect users: Use SSL on any authenticated connection or during the transmission of sensitive data, such as user credentials, credit card details, health records and other private information. SSL or a similar encryption protocol should also be applied to client, partner, staff and administrative access to online systems. Use transport layer security or protocol level encryption to protect communications between parts of your infrastructure, such as Web servers and database systems.

10. Failure to restrict URL access

The problem: Some Web pages are supposed to be restricted to a small subset of privileged users, such as administrators. Yet often there’s no real protection of these pages, and hackers can find the URLs by making educated guesses. Say a URL refers to an ID number such as “123456.” A hacker might say ‘I wonder what’s in 123457?’ Williams says.

The attacks targeting this vulnerability are called forced browsing, “which encompasses guessing links and brute force techniques to find unprotected pages,” OWASP says.

Real-world example: A hole on the Macworld Conference & Expo Web site this year let users get “Platinum” passes worth nearly $1,700 and special access to a Steve Jobs keynote speech, all for free. The flaw was code that evaluated privileges on the client but not on the server, letting people grab free passes via JavaScript on the browser, rather than the server.

How to protect users: Don’t assume users will be unaware of hidden URLs. All URLs and business functions should be protected by an effective access control mechanism that verifies the user’s role and privileges. “Make sure this is done … every step of the way, not just once towards the beginning of any multi-step process,’ OWASP advises.