



Web application architecture  
vulnerabilities: is the functionality  
of your application secure?  
White paper



## Table of contents

|  |   |
|--|---|
| <b>Introduction</b> .....                              | 2 |
| <b>Defining architecture web vulnerabilities</b> ..... | 2 |
| <b>Scanning to find vulnerabilities</b> .....          | 2 |
| Administration sections .....                          | 2 |
| Mailers .....  | 3 |
| Wikis .....  | 5 |
| Forums .....   | 6 |
| Heavy load applications .....                          | 6 |
| <b>Conclusion</b> .....                                | 6 |
| <b>Appendix: references</b> .....                      | 7 |

## Introduction

Web application architecture vulnerabilities affect many websites, but few webmasters categorize them as such or recognize them. These problems are designed into a web application and are easy to find, but you may not notice them, because they can blend into the functionality of the application. Although developers, quality assurance (QA) engineers and security professionals may not identify them, hackers do. This white paper informs all participants in the software development lifecycle about the unintended, potential security implications of web applications.

## Defining architecture web vulnerabilities

Architecture web vulnerabilities are inherent vulnerabilities in the design of an application. Attackers can exploit them by using standard functionality in a malicious way. Without proper measures, attackers can use functionality in ways never envisioned by developers. You cannot fix the vulnerabilities with proper input validation as with most security vulnerabilities that exist on the Internet today. Architecture vulnerabilities require that you design security into the application from the beginning.

Remember, you need to think like a hacker to prevent hackers from launching successful exploits. Can you use functionality to break the application or manipulate it into revealing information for launching more damaging attacks?

Most people do not notice architecture vulnerabilities. For example, a product manager may request new functionality for a website, such as a subscription mailer. A developer, who may be security conscious, adds the functionality to the site. A QA professional analyzes it

and determines whether it is functional and secure from cross-site scripting (XSS), SQL injection and other risks. A security engineer does not see security threats, and in most cases, excludes the new functionality from scans.

## Scanning to find vulnerabilities

Why would someone exclude areas of a website from a scan? The answer is simple: Scanning a specific page may exploit a vulnerability and damage the site. So, the scanner is set up to avoid that page. Therefore, vulnerabilities usually appear after a scan occurs unless the scan includes that area. Security professionals have known for years that these vulnerabilities exist but have not classified them as vulnerabilities. They call these vulnerabilities “gotchas” or areas of the website to avoid.

Web scanners crawl through an application, click on links, submit form data and attack everything they find. Scanners find architecture vulnerabilities by submitting too much data or by sending input too many times to a form that cannot accept it. When a scanner accesses an area that it shouldn't or does something unexpected, the application is designed improperly.

Architecture vulnerabilities can come in many forms, including: administration sections, mailers, wikis, forums and heavy load applications.

### Administration sections

Some applications have an administration section that resides on the same server and same port as the application that you are trying to assess. Some administration sections disable or enable parts of the application, uninstall modules, add or remove users, change passwords, run batch jobs and even shut the server down. Some administration sections reside on the same server through a section that has no link to it. A

directory with a random name houses the controls. In this case, security through obscurity is not effective: A hacker can guess paths and targets in the application and try to return data.

Other sections use a web form to log into the administration section. The form asks for a user name and password. This can be a good approach if the user name and password are strong. In some cases, the log-in page may have a SQL injection vulnerability. For example, a hacker can get into the application by entering OR '5'='5.

Ideally, you should combine several steps:

1. Put the administration section on a separate server, or at a minimum, use a different host name.
2. If possible, restrict the other server and host name so that only internal IP addresses from your company and domain can access them.
3. If it must be external, the page that performs the administration should use web server authentication, such as NTLM or Kerberos, not Basic. This authentication can guard against attacks and is easy to implement.
4. Make sure you protect the directory above the administration section against attack:
  - Configure it to return an HTTP status code of 404 NOT FOUND instead of 403 FORBIDDEN, 401 UNAUTHORIZED or 200 OK. The status code of 404 NOT FOUND makes the directory disappear to the hacker and he or she will move on. If you are running Internet Information Services (IIS), set the physical directory on the web server to "hidden," which makes IIS return a 404 status code, but you can still access the files below the directory.

- Give the directory an uncommon name, not easily guessed by a dictionary of attacks. Don't use "admin" or variations of it.

An example is:

<http://www.yoursite.com/prodahr1/ubcontroller.aspx>

<http://www.yoursite.com/prodahr1/> is configured to protect the directory against attack.

### Mailers

Many people complain when a scanner hits a mailer form. Mailers are common on the Internet, but without proper measures, attackers can misuse them. A web application typically has an input box where a user enters either an e-mail address or contact information, and an e-mail is sent. There are different types of mailers:

- **E-mail subscription forms:** The user can enter an e-mail address to receive updates.

Figure 1. E-mail subscription form

The image shows a web form titled "Subscribe". Below the title, there is a message: "To subscribe, please enter a valid e-mail address below. All other fields are optional and will only be used for personalization of your newsletter." The form itself is titled "Sign-Up Form" and contains the following elements: "First Name:" followed by an input box; "Last Name:" followed by an input box; "\* Email Address:" followed by an input box; "Email Format:" followed by a dropdown menu currently showing "HTML"; and a "Subscribe" button at the bottom right.

- **Registration forms:** After a user creates a user name and password to access a site, an e-mail is sent to verify the information.

Figure 2. Registration form

The screenshot shows a registration form titled "MEMBER CENTER" with a "Sign In | Register" link. The form includes the following fields and options:

- Register - BE THE FIRST TO KNOW WHEN NEWS BREAKS:** Get instant access to [redacted] BREAKING NEWS E-MAIL ALERTS. Plus, get instant register now or sign in below. [Read more about member benefits.](#)
- Select your country:** United States of America (dropdown menu)
- First name:** [text input field]
- Enter your e-mail address:** [text input field]
- ZIP/Postal Code:** [text input field]
- Job responsibility:** Select your job responsibility (dropdown menu)
- Privacy Policy:** CNN.com values and maintains your privacy. The information you submit is subject to  Yes, I have read and agree to [redacted] [privacy policy](#)
- Breaking News Alerts:**  [redacted] Breaking News Alerts. Be the first to know when there's breaking financial news with timely alerts delivered to your inbox.
- Yes, I would like to receive occasional [redacted] member updates about new features.
- Age Confirmation:** By submitting this form, I agree that I am at least 13 years of age.
- Buttons:** Submit, Cancel

- **"Contact us" forms:** An inquiry e-mail message is sent to a general address, such as info@company.com, or a form with an issue is e-mailed to the customer support team.

Figure 3. "Contact us" form

The screenshot shows a "Contact Us form" with the following fields and options:

- Topic or category:** Select (dropdown menu)
- Affiliation:** General Public (dropdown menu)
- Subject:** [text input field]
- Message:** [large text area]
- Do you want a reply?**  Yes  No
- If yes, how should we respond?**  E-mail  Letter  Fax
- E-mail address:** [text input field]
- Mailing address, line 1:** [text input field]
- Mailing address, line 2:** [text input field]
- Mailing address, line 3:** [text input field]
- Fax number:** [text input field]

For all of these types of mailers, a web application works with a mail server. If a scanner hits a mailer, the mailer appears as another form to scan. When testing this form, the scanner submits different values for each input method and sees what the web application returns. Nothing may appear in the scan; however, the scanner submits thousands of requests, which in turn clog the mail server with a backlog of e-mails to send. For one customer, a mail server sent e-mails three hours after a scan stopped.



**CAPTCHA stands for “Completely Automated Public Turing test to tell Computers and Humans Apart.” Basically, it is one of those pictures with a word contained inside, that the user has to enter in before continuing. It is quite difficult to create a script to automatically enter to text to continue, but there is some research taking place on how to break them, so if you decide to use this method make sure to use a strong CAPTCHA. The research even suggests that there is a way to create strong CAPTCHAs which would be nearly impossible to bypass automatically.**

If a user can submit the same form over and over again and clog your e-mail server or send hundreds of thousands of e-mails, you have an architecture vulnerability. If the user can select an e-mail address for sending the e-mail, your architecture vulnerability is worse. A hacker can use your site to target and fill any victim’s mailbox and make the e-mail appear as if it came from your site.

Misusing a mailer can have more serious repercussions. Many automated spam controls exist on the Internet today with white lists and black lists in effect. With only a few e-mails that look like spam, filters take action. If your company sends thousands of e-mails, filters can automatically block users. The spam filters can block all e-mails from your company or your entire domain. It can take hours or days to remove you from the different black lists.

Some websites only allow a certain amount of outbound traffic or e-mails a month. If that limit is exceeded, extra charges apply. If a hacker can generate a large amount of outbound data, either through a large number of outbound e-mails or a large amount of data in each e-mail, you can spend your entire website budget.

Many companies use several popular yet inadequate solutions to this problem. Many block a user from submitting e-mail multiple times, usually based on session information. Hackers can simply remove the session information so they cannot be tracked. Many prevent an IP address from making multiple submissions. However, an IP address can be randomized easily. If a form does not require returning data to a user before sending e-mail—and most do not—hackers can randomize the return IP, send a request to the server, randomize a new IP, send another request and continue repeating this process.

The best solution is to use something like a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) to prevent attackers from writing scripts for web forms. While they can still submit some e-mails manually, it is too tedious to submit thousands of e-mails, and hackers move on.

### **Wikis**

Wikis are web applications that let users quickly and easily edit the content on a website. While Wikipedia has provided the name for wikis, they have existed for a while and can be architecture vulnerabilities if you do not take proper measures.

Wikis are vulnerable when all of the following are true:

- The content on a site is important.
- Anonymous users can edit the site.
- There is no verification step.

You should prevent one of these from happening. The best way to prevent someone from adding unacceptable content is to prevent outside users from editing a wiki site. Only allow employees and trusted individuals to change content, and grant everyone else privileges to view the content. This may not always be possible. Wikis are popular because they bring together knowledge from many different users.

You can also require verification and let everyone edit a wiki. However, don’t post the changes until an employee or other trusted individual has verified them.

The best method is one that Wikipedia describes. Wikipedia lets everyone make incremental changes. If you try to make a change that is too large, it does not make the change until a trusted person can verify it. If you create a user name for making the change and you make many changes that are not rejected over a



Tom Cross did some great work on how wiki information can be best displayed to show the reliability of it. His report shows how you can color-code text to show how long it has gone without change, which can be interpreted to mean that it is reliably accurate.

period of time, you can request to be an administrator. Administrators have more power, can control other users and must approve large changes. Wikipedia also uses MediaWiki, free wiki software that may be suitable for your organization.

### Forums

A forum is a web application that allows users to post thoughts, opinions or questions to a round table where others can comment on them. Most forums are susceptible to attacks because they have architecture vulnerabilities.

Many forums require that users register before they can post something on the site. Additionally, a forum may verify user information by sending users e-mail with a link to confirm their e-mail addresses: This is a mailer architecture vulnerability. After the account is created, some sites allow unlimited postings. Attackers can write scripts to flood the server with posts and responses to fill the site with their content until the site is “cleaned up” by one of the administrators.

There are many ways to prevent malicious posts on forum sites. Adding CAPTCHAS to every posting prevents script attacks. You can also assign user-based trust levels, based on the number of postings submitted without a warning. For example, set new users to low trust levels so that they can only post twice a day. After 30 days or 30 posts, whichever comes first, they can post four times a day, and so on.

### Heavy load applications

Many applications on the Internet are heavy load applications. They take a long time to process a request that comes in, even if it is a valid request and not a buffer overflow attack. A heavy load application can be a flight reservation page or a simple search box. Hackers may make 20,000 requests a minute to the application, and each appears as a distinct request from a valid user. You may consider this a denial of service attack, but the underlying problem is an architecture vulnerability.

If your application takes a long time to process a valid request, then your application may be vulnerable. When scanners hit these applications, valid users may see the site operate very slowly. Bandwidth is not affected—the web application causes the problem. A request may take a long time to run, or a database may need to retrieve a lot of data.

To prevent heavy load applications, make sure that you perform load testing on the site. Several load testing applications are available, including open source applications. Make sure that your QA organization performs tests, including performance analysis, and make sure that you conduct load testing correctly.

### Conclusion

For a while, penetration testers have reviewed web applications prior to scanning and found areas that scanners mishandle, creating a problem for application administrators. You need to thoroughly examine these areas, because they are vulnerabilities in the web application. If a scanner destroys parts of an application, creates many e-mails or shuts down servers, you have a vulnerability, and you need to address the problem. Remember, if a scanner can do something, so can a hacker.

## Appendix: references

### CAPTCHAs

#### **CAPTCHA information from Wikipedia**

<http://en.wikipedia.org/wiki/Captcha>

#### **ASP.NET CAPTCHA code project**

<http://www.codeproject.com/aspnet/CaptchaImage.asp>

#### **CAPTCHA project**

<http://www.captcha.net/>

#### **CAPTCHA decoder (breaking CAPTCHAs)**

<http://sam.zoy.org/pwntcha/>

### Wikis

#### **Wikipedia, the best-known wiki**

<http://www.wikipedia.org>

#### **A history of wikis**

<http://en.wikipedia.org/wiki/Wiki#History>

#### **Administration FAQs for Wikipedia**

[http://en.wikipedia.org/wiki/Wikipedia:Administration\\_FAQ](http://en.wikipedia.org/wiki/Wikipedia:Administration_FAQ)

#### **MediaWiki site**

<http://www.mediawiki.org/wiki/MediaWiki>

#### **Register article on wiki abuse**

[http://www.theregister.co.uk/2006/09/05/defra\\_wiki\\_abuse/](http://www.theregister.co.uk/2006/09/05/defra_wiki_abuse/)

#### **Tracking website attacks and defacements**

[http://www.zone-h.org/component/option,com\\_attacks/Itemid,44/](http://www.zone-h.org/component/option,com_attacks/Itemid,44/)

(This is a restricted website. You need to log in.)

### Load testing

#### **Best practices for Java™ load testing**

[http://developers.sun.com/appserver/reference/techart/best\\_practices/best\\_practices\\_3.html](http://developers.sun.com/appserver/reference/techart/best_practices/best_practices_3.html)

#### **Load testing using Visual Studio Team System**

<http://blogs.msdn.com/billbar/pages/517081.aspx>

#### **Comprehensive list of load and performance test tools**

<http://www.softwareqatest.com/qatweb1.html#LOAD>

#### **HP LoadRunner software**

<http://www.mercury.com/us/products/performance-center/loadrunner/>

#### **IBM Rational Performance Tester**

<http://www-306.ibm.com/software/awdtools/tester/performance/index.html>

#### **RadView WebLOAD Open Source**

<http://www.radview.com/products/LoadTesting-WebLOAD-Overview.aspx>

---

To learn more, visit [www.hp.com/go/software](http://www.hp.com/go/software)

© Copyright 2007 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein. Java is a U.S. trademark of Sun Microsystems, Inc.

4AA1-5394ENW, October 2007

