

# Ajax Security

## Whitepaper

### Learn about:

- Client-side security
- Cross-Site Scripting issues
- Caching Security
- JSON Weaknesses
- and more

## Introduction

---

With security in the sights of IT executives like never before, it's understandable that new technologies like Ajax are being examined carefully for potential security pitfalls. Is the technology behind Yahoo Mail, Google Gmail and Google Map introducing new security threats into the enterprise? The quick answer is no. There were security vulnerabilities before Ajax, and the issues remain the same with Ajax.

Ajax doesn't introduce new security holes or require special security over and above industry-standard best practices. This paper will highlight some of those best practices and illustrate how they are implemented when developing Ajax solutions using the Backbase framework.

## Understanding Ajax

AJAX (Asynchronous JavaScript + XML) combines existing technologies to allow Web page content to be updated dynamically. With Ajax you no longer need to refresh a page to update data. In an Ajax-enhanced Web page, data is typically displayed using HTML, XSLT, or JavaScript, or a combination of the three. Data is typically transferred from the Web server and processed in the browser.

In Backbase's implementation, the Ajax engine receives data from the Web server and renders it in the way designed by the application. The mechanism for performing asynchronous data transfer is XMLHttpRequest (XHR), a feature embedded in all modern Web browsers.

## Where is the security problem?

Some security commentators assert that AJAX opens new security vulnerabilities. But a clear understanding of Ajax technology dispels most of these alleged problems.

Concerns about cross-site scripting reflect an incomplete understanding of the security precautions built into the XHR specification. Likewise, concerns about caching show a misunderstanding of how browsers handle cached files. Other concerns are based on invalid assumptions — for example, trusting requests to the server. In practice today, production servers do not trust an unauthenticated request, so Ajax use would not change that standard. And, while security concerns about JSON are valid, they do not apply to Backbase widgets which do not use JSON.

Let's examine each of these concerns in detail, identifying root causes and highlighting security best practices for Ajax solutions.

## Client-side security

As a rule of thumb, websites should never trust a client or Web browser. Developers should always be mindful that basic HTTP proxies can modify a request en route. This means that server security remains essential. Ajax critics posing the question, "What if the Ajax client makes a malicious request?" are presupposing that this rule of thumb no longer applies. Authentication and validation of user input on the server-side is still mandatory.

## Cross-Site Scripting (XSS) Attacks

JavaScript has been used for many years and, when it's enabled in the browser, it opens a variety of security vulnerabilities. These vulnerabilities — and the security best practices that reduce the risks — are well known and widely practiced. Cross-site scripting is one example, and these attacks existed before AJAX was introduced. However, these attacks are not normally possible using the XHR, because the XHR doesn't allow contact with any server other than the originating Web server. This constraint is a deliberate security feature designed to eliminate the risk of cross-site scripting.

## A larger attack surface

One criticism getting a lot of ink currently is that Ajax increases an application's attack surface. The attack surface is composed of the points in a system that are vulnerable to attack. It is true that Ajax applications tend to encourage a service-oriented architecture, replacing a single monolithic Web application on the server with a set of simple services, thus increasing the number of access points. But this argument fails to consider the complexity of each point.

A complex monolithic application is likely to have many more undetected vulnerabilities than a set of simple services. Simple services are easier to test, audit, and lock down. Considering only the number of access points is therefore misleading. To date, automatic crawling and black-box website testing remain the best ways to assess attack surface risk on a server.

## Caching Security

Browsers normally cache the files they load to avoid reloading them each time they're needed, thus allowing pages to load faster. Files are stored using their URLs, so that `www.google.com/index.html` is treated as a separate file from `www.yahoo.com/index.html`.

The Backbase engine is a JavaScript file that is loaded like any other script. Once it is cached it doesn't need to be loaded again for other Backbase-enabled pages on the same site. This cached copy of the Backbase engine won't be used for another Backbase-powered site because the URL it's loaded from is different. And this cached file isn't accessed when the users browse to other sites.

While it is theoretically possible to access the Backbase engine from the browser cache and modify it, this requires physical access to the user's computer. If the attacker has physical access to the user's computer we have far more serious problems. An attacker could, in a matter of minutes, install a key-logger or other spyware to collect passwords and private information.

## JSON Weaknesses

JSON (JavaScript Object Notation) is sometimes used as an alternative to XML for communication between client and server. Essentially, the server responds to a request with literal JavaScript that is evaluated immediately by the browser, trusting that the script has not been maliciously altered. In addition to this so-called evaluation vulnerability, JSON opens other vulnerabilities like cross-site request forgery. For these reasons, Backbase uses XML in preference to JSON.

## Server Vulnerability

The introduction of Ajax clients is often accompanied by the introduction of a service-oriented architecture (SOA) on the server-side. As a result, some have mistakenly attributed the risks of SOA to Ajax. While these technologies are clearly complementary, they are nevertheless distinct. The security risks and best practices for SOA design are well-documented and are not changed by the addition of Ajax on the client-side.

For example, one SOA security principle is that services must not trust requests. A service must assume that all input data can be maliciously crafted and must validate all data before using it. This principle applies whether the front-end is a plain HTML page or an Ajax-enabled rich user interface.

## Ajax Engine Vulnerability

Another common question is, whether the Backbase Ajax engine can be compromised on the server. This isn't possible, however, unless the attacker compromises the production server, a severe security breach that should be guarded against by standard corporate security policies.

## Application Developer Security Exposure

One threat agent that's often overlooked is the application developer. Using Backbase doesn't change the risk of malicious code being introduced into applications. Although attacks by internal developers are difficult to detect, most organizations have already established standard security disciplines and have protections in place to deal with this threat.

## Network Security

Whenever information travels the Internet, there is the possibility of interception and man-in-the-middle attacks, whereby the contents of a request or a response is modified en route. Using SSL (https://) to secure the connection eliminates much of this risk as well as keeping the content of communications private.

## Summary

---

- Ajax doesn't introduce new security issues
- Backbase highly recommends using XML instead of JSON
- Ajax clients often work with a service-oriented architecture, for which security best practices must be implemented
- Using SSL (https://) protects against many Web application vulnerabilities

## The AJAX Top security tips:

Follow best practices from sites like the Open Web Application Security Project ([www.owasp.org](http://www.owasp.org)). These guidelines include checking for Access Control and Input Validation flaws and sending sensitive information over SSL rather than in the clear.

For a security audit of your application, or security best practices in your application design and development stages, contact a Backbase Ajax Security Expert.

---

## Contact Backbase, The Ajax Company:

**Americas:**  
635 Mariners Island Blvd Suite 200  
San Mateo, CA 94404  
USA

Tel: +1 866 800 8996  
Email: [sales-us@backbase.com](mailto:sales-us@backbase.com)

**Europe and Asia:**  
Stephensonstraat 19  
1097 BA Amsterdam  
The Netherlands

Tel: +31 20 465 8888  
Email: [sales-eu@backbase.com](mailto:sales-eu@backbase.com)