Top 10 Web 2.0 attack vectors

Shreeraj Shah
Founder, Net Square
shreeraj@net-square.com

Web 2.0 is the novel term coined for new generation Web applications. *start.com*, *Google maps*, *Writely* and *MySpace.com* are a few examples. The shifting technological landscape is the driving force behind these Web 2.0 applications. On the one hand are Web services that are empowering server-side core technology components and on the other hand are AJAX and Rich Internet Application (RIA) clients that are enhancing client-end interfaces in the browser itself. XML is making a significant impact at both *presentation* and *transport* (HTTP/HTTPS) layers. To some extent XML is replacing HTML at the presentation layer while SOAP is becoming the XML-based transport mechanism of choice.

WEB 2.0 Security concerns – Reshaping the industry

This technological transformation is bringing in new security concerns and attack vectors into existence. *Yamanner*, *Samy* and *Spaceflash* type worms are exploiting "client-side" AJAX frameworks, providing new avenues of attack and compromising some of the confidential information. On the "server-side", XML based Web services are replacing some of the key functionalities and providing distributed application access through Web services interfaces. These remote capabilities to invoke methods over GET, POST or SOAP from the Web browser itself provide new openings to applications. On other side, RIA frameworks running on XML, XUL, Flash, Applets and JavaScripts are adding new possible sets of vectors. RIA, AJAX and Web services are adding new dimensions to Web application security.

Here is the list of 10 attack vectors along with a brief overview of each:

1. Cross-site scripting in AJAX

In last few months, several cross-site scripting attacks have been observed, where malicious JavaScript code from a particular Web site gets executed on the victim's browser thereby compromising information. A recent example is the Yamanner worm that exploited cross-site scripting opportunities in *Yahoo mail*'s AJAX call. Another recent example is the Samy worm that exploited MySpace.com's cross-site scripting flaw. AJAX gets executed on the client-side by allowing an incorrectly written script to be exploited by an attacker. The attacker is only required to craft a malicious link to coax unsuspecting users to visit a certain page from their Web browsers. This vulnerability existed in traditional applications as well but AJAX has added a new dimension to it.

2. XML poisoning

XML traffic goes back and forth between server and browser in many of the WEB 2.0 applications. Web applications consume XML blocks coming from AJAX clients. It is possible to poison this XML block. Not uncommon is the technique to apply recursive payloads to similar-producing XML nodes multiple times. If the engine's handling is poor this may result in a denial of services on the server. Many attackers also produce malformed XML documents that can disrupt logic depending on parsing mechanisms in use on the server. There are two types of parsing mechanisms available on the server side – SAX and DOM. This same attack vector is also used with Web services since they consume SOAP messages and SOAP messages are nothing but XML messages. Large-scale adaptation of XMLs at the application layer opens up new opportunities to use this new attack vector.

XML external entity reference is an XML property which can be manipulated by an attacker. This can lead to arbitrary file or TCP connection openings that can be leveraged by an attacker. XML schema poisoning is another XML poisoning attack vector which can change execution flow. This vulnerability can help an attacker to compromise confidential information.

3. Malicious AJAX code execution

AJAX calls are very silent and end-users would not be able to determine whether or not the browser is making silent calls using the `XMLHTTPRequest` object. When the browser makes an AJAX call to any Web site it replays cookies for each request. This can lead to potential opportunities for compromise. For example, John has logged in to his bank and authenticated on the server. After completing the authentication process he gets a session cookie. His bank's page has a lot of critical information. Now he browses other pages while still logged in to his bank's account Web page and lands at an attacker's Web page. On this page the attacker has written silent AJAX code which makes backend calls to his bank without John's consent, fetches critical information from the pages and sends this information to the attacker's Web site. This leads to a security breach and leakage of confidential information.

4. RSS / Atom injection

This is a new WEB 2.0 attack. RSS feeds are common means of sharing information on portals and Web applications. These feeds are consumed by Web applications and sent to the browser on the client-side. One can inject literal JavaScripts into the RSS feeds to generate attacks on the client browser. An end user visits this particular Web site loads the page with the RSS feed and the malicious script – a script that can install software or steal cookies – gets executed. This is a lethal client-side attack. Worse, it can be mutated. With RSS and ATOM feeds becoming integral part of Web applications, it is important to filter out certain characters on the server-side before pushing the data out to the end user.

5. WSDL scanning and enumeration

WSDL (Web Services Definition Language) is an interface to Web services. This file provides key information about technologies, exposed methods, invocation patterns, etc. This is very sensitive information and can help in defining exploitation methods. Unnecessary functions or methods kept open can cause potential disaster for Web services. It is important to protect WSDL file or provide limited access to it. In real case scenarios, it is possible to discover several vulnerabilities using WSDL scanning.

6. Client side validation in AJAX routines

WEB 2.0 based applications use AJAX routines to do a lot of work on the client-side, such as client-side validations for data type, content-checking, date fields, etc. Normally, these client-side checks must be backed up by server-side checks as well. Most developers fail to do so; their reasoning being the assumption that validation is taken care of in AJAX routines. It is possible to bypass AJAX-based validations and to make POST or GET requests directly to the application – a major source for input validation based attacks such as SQL injection, LDAP injection, etc. that can compromise a Web application's key resources. This expands the list of potential attack vectors that attackers can add to their existing arsenal. AJAX routines are

7. Web services routing issues

Web services security protocols have WS-Routing services. WS-Routing allows SOAP messages to travel in specific sequence from various different nodes on the Internet. Often encrypted messages traverse these nodes. A compromise of any of the intermediate nodes results in possible access to the SOAP messages traveling between two end points. This can be a serious security breach for SOAP messages. As Web applications move to adopt the Web services framework, focus shifts to these new protocols and new attack vectors are generated.

8. Parameter manipulation with SOAP

Web services consume information and variables from SOAP messages. It is possible to manipulate these variables. For example, "`<id>10</id>`" is one of the nodes in SOAP messages. An attacker can start manipulating this node and try different injections – SQL, LDAP, XPATH, command shell – and explore possible attack vectors to get a hold of internal machines. Incorrect or insufficient input validation in Web services code leaves the Web services application open to compromise. This is a new available attack vector to target Web applications running with Web services.

9. XPATH injection in SOAP message

XPATH is a language for querying XML documents and is similar to SQL statements where we can supply certain information (parameters) and fetch rows from the database. XPATH parsing capabilities are supported by many languages. Web applications

consume large XML documents and many times these applications take inputs from the end user and form XPATH statements. These sections of code are vulnerable to XPATH injection. If XPATH injection gets executed successfully, an attacker can bypass authentication mechanisms or cause the loss of confidential information. There are few known flaws in XPATH that can be leverage by an attacker. The only way to block this attack vector is by providing proper input validation before passing values to an XPATH statement.

10. RIA thick client binary manipulation

Rich Internet Applications (RIA) use very rich UI features such as Flash, ActiveX Controls or Applets as their primary interfaces to Web applications. There are a few security issues with this framework. One of the major issues is with session management since it is running in browser and sharing same session. At the same time since the entire binary component is downloaded to the client location, an attacker can reverse engineer the binary file and decompile the code. It is possible to patch these binaries and bypass some of the authentication logic contained in the code. This is another interesting attack vector for WEB 2.0 frameworks.

Conclusion

AJAX, RIA and Web services are three important technological vectors for the WEB 2.0 application space. These technologies are promising and bring new equations to the table, empowering overall effectiveness and efficiency of Web applications. With these new technologies come new security issues, and ignoring them can lead to big disasters for the corporate world. In this article, the discussion was restricted to only ten attacks but there are several other attack vectors as well. Increased WEB 2.0 security awareness, secure coding practices and secure deployments offer the best defense against these new attack vectors.