AJAX – A TESTER'S POINT OF VIEW

Aakash Vakil, CSTE

Executive Summary

Rapidly shifting market conditions have necessitated changes in the mode in which applications are offered to users. Innovative ways to present have come in place of static old HTML pages. Embracing "Rich Internet Application" is becoming a need for continued existence for just about any corporation that has web presence.

AJAX acceptance is on the rise swiftly, but the testing done is not adequate for the magnitude of security risks that AJAX faces. Black box testers are testing the functionality, but security concerns that are the focal area of apprehension are still left untested.

This paper presents AJAX overview – what a black box tester should know before he leaps into AJAX testing and the security risks that AJAX applications open up. We will see what the market trends are for AJAX adoption and its prospective growth potential.

Finally we will have a look at some of the black box test ideas and recommendations that test two main security concerns in AJAX implementation – Data Validation & Session Management.

AJAX Overview:

Is it a tool or a technique?

When we talk about developing internet applications, we are often presented with a conflicting choice between developing 'Rich Internet Applications (RIA)' and developing applications with 'Quick Response Time'.

We see a high degree of variance in performance of web applications to their client server counter parts. Lately, we are seeing enormous progress in the way applications are presented to users. This is what gives web application the look and feel of a desktop application without compromising on the ease of use and accessibility.

One such interesting and popular technique is AJAX.

AJAX is an acronym for '**Asynchronous Java Script and XML'**. It is a new approach to web development that combines several existing technologies in a unique way. As the name suggests, it's a combination of Java Script and XML, but what's the deal with 'Asynchronous'?

'Asynchronous' is the state of not being 'synchronized' – simply put, AJAX allows you to refresh & update part of the web page without re-loading the complete page. In techie terms, it avoids 'post backs' which is nothing but sending request to server every time page needs updated data. It uses Java script for local processing instead of exclusively using server side scripting.

AJAX aims to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is where "Asynchronous" comes into play – the data exchange is done by the AJAX engine and is transparent to the end user.

We should note that AJAX by itself is not a programming language or a development environment. It is a set of programming techniques used in conjunction with a programming language to enhance the user's web application experience.

Popular examples include our favorite search engine – 'Google' that has a personalize option that uses AJAX to provide drag and drop portlet's that users can choose and the very informative 'Google Maps' that renders images smoothly without refreshing the page when you navigate to a different location

Let's move on to see where market is headed and how web application development is going to adapt itself to the new flavor.

Market Trends:

What is driving this wave?

Drivers for AJAX adoption

AJAX applications are gaining popularity and are changing the way organizations do business. The question that comes to mind is what the drivers are for this massive adoption of AJAX enabled applications.

Let us look at the top reasons why organizations are going in for Rich Internet Applications using AJAX.

1) **Greater acceptance among online users:**

   Almost 50% of online users have responded positively to the RIA applications like Google Maps stating that the web experience has been improved greatly.

2) **Overcomes limitations posed by HTML:**

   Improved visualization and data rendering are the main advantages that one gets over HTML when RIA is used.

3) **Immediate results:**

   70% of organizations that measure the result of RIA implementation reported immediate effect on the top line growth which they categorized as "far more successful then expected"

RIA / AJAX Adoption Projection

CIO's are spending big money on getting their front end web sites & applications AJAX enabled. That's the message that we are getting from CIO's of fortune 500 companies.

Only 25% of all organizations have their web applications that implement AJAX. We have a tremendous opportunity available – three fourths of market will be open to capture in the next 12-24 months.

Refer to figure 1.0 to get an idea of the number of companies planning to spend on RIA / AJAX implementations in next 12-24 months.

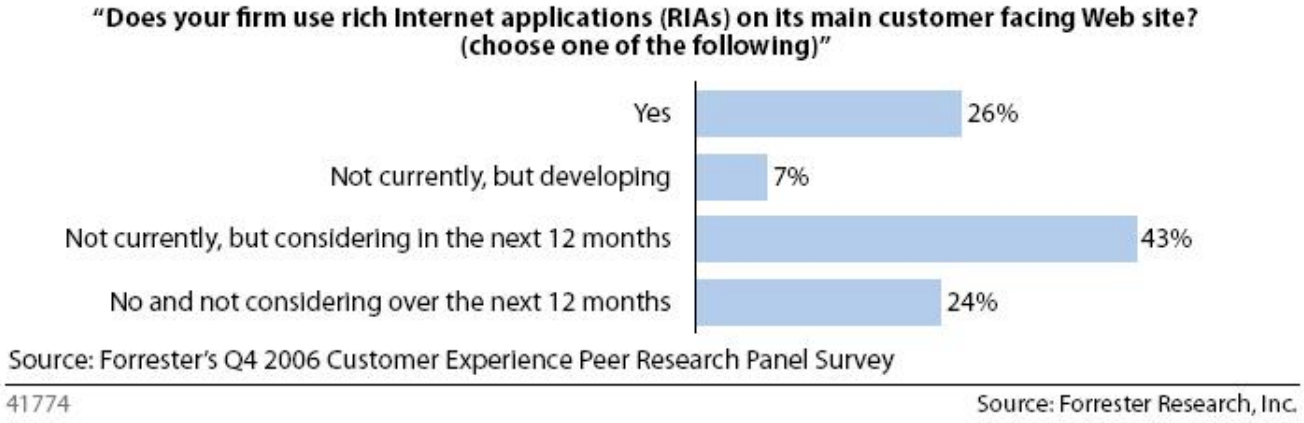

Figure 1.0: RIA Adoption Projections in 12-24 months

AJAX implementation: Major Risks

What gives Mr. CIO sleepless nights

No – we are not talking about insomnia!! We are talking about the risks that AJAX application present. AJAX applications have a lot of promise and immense potential to become the next killer application. However, along with the positives, there are risks that come along with AJAX adoption. AJAX implementation is new – but the technology that is combined to make AJAX applications is not at all new. It is a known set of technology with know risks. However, when these technologies are combined together the collective risk is much higher than earlier.

These risks are so colossal that, if not tackled in an effective way, can leave an organization in extremely vulnerable state. It is something no organization can afford to ignore.

The following two risks pose severe threat and can negatively affect companies image and user experience:

Security Compromise

This is the number one question in every CIO's list. Technology experts should be prepared to answer the following questions that haunt stake holders and give them sleepless nights.

- Will my application security be compromised when I make it AJAX compliant?
- How is my application responding to security risks?

Mitigating security risks needs considerable investment in terms of time and money.

Java script in itself has known security risks. However, AJAX implements client side java script which becomes even more risky business. AJAX, if not implemented thoughtfully, opens up multiple windows from where security of an application can easily be compromised.

Some of the more prominent security threats in AJAX applications are as follows:

- Lack of data validation
- XMLHttpRequest Vulnerabilities
- SQL Injection
- Cross Site Scripting
- AJAX Bridging
- Cross Site Request Forgery (CSRF)
- Denial of Service

Don't worry if you are not able to comprehend the security threats written above. It will sound latin and greek for most of us who have done mostly black box testing throughout our careers.

We will look at the test strategy to mitigate these risks once we go to the testing recommendations later on in the paper.

Performance Compromise

Performance has always been a challenge for a web application. Statistics say that users do not wait for more then a few seconds before they move on to the next site. You can win or loose a customer in few seconds.

AJAX applications have a risk of slowing down the application performance due to high volume of processing happening at client side along with the rich interface that can take considerable time in page loading.

Main concerns of stake holders on application performance are:

- Application performance degradation after AJAX implementation
- Sustainability of the current goals and visitor to customer conversion rates

Opportunity Assessment

What testing opportunities exists

AJAX is going to change the way business takes place – it is going to make it easy and intuitive for the customers and end users. However, the main business model still remains as it was before AJAX implementation.

So the question now is - what additional testing opportunities are presented with this model? Well, the basic business model is not going to change – therefore we will have a lot of additions made to the testing scope.

In Figure 2.0, we see that we build upon the traditional black box testing that we currently carry out and add tests to target specific risks posed by AJAX applications – Security and Performance.

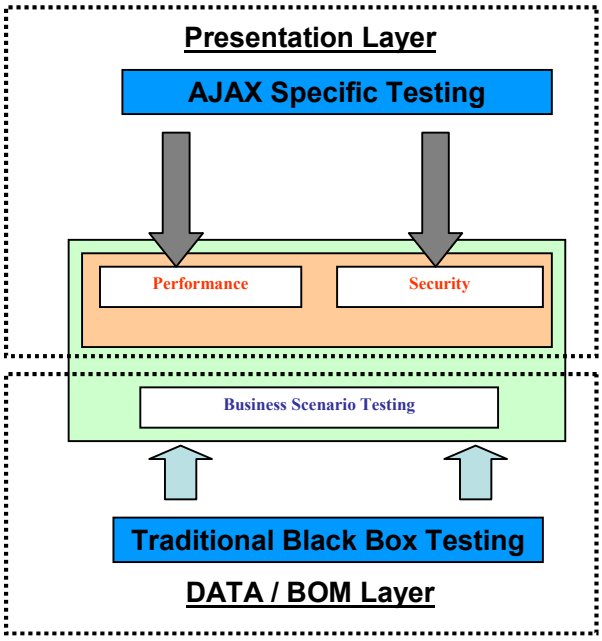

Figure 2.0: AJAX testing opportunities in security and performance space

Lots of organizations are coming up with support for RIAs. Microsoft has come up with its own flavor for 'AJAX' and Adobe is contemplating going the open source way for its 'Flex' toolkit. These two are the two most visible organizations declaring support for RIA.

As the industry matures we will have a host of organizations providing sophisticated development tools to create RIA applications. This is all very good for the industry, but at the same time, a nightmare for security and performance analysts who are unsure of the stability and reliability of the development tools used to create RIAs.

In this paper we will discuss the Security testing opportunities that are presented to us – we will look at the market demand in terms of specific services and tester skill set.

Security Testing

It will be worth noting that 'Security' concerns contributed 69% of the total vulnerabilities reported in a research conducted by Forrester. More and more organizations are moving towards B2B and B2C model of sales. This makes it increasingly important that these organizations pay a lot of attention to security of their applications along with focusing on improving user experience.

Organizations will spend up to 45% of their total budget for getting RIA ready in making sure that their application is secured and does not compromise on security concerns. In pure dollar terms the world wide security testing market for Rich Internet Applications is pegged at around USD 250 Million over next 3 years.

It makes business wisdom to focus on this niche and provide specialized security testing consulting services customized for Rich Internet Applications.

AJAX Black Box Testing Recommendations

Adapting to change is vital to survival

Testing AJAX applications needs amalgamation of different testing styles. We cannot afford to sign-off the application with just black box business scenario testing. As I mentioned earlier in this paper, black box testing is one side of the coin – the other side is in-depth security testing.

We have briefly seen the security threats earlier when we spoke about security concerns – it may not be possible for black box testers to fully understand and test for these security threats. Most of the test teams that test AJAX implementations are doing it the traditional black box way with no formal security testing planned.

However, there are tests that a black box tester can perform to check the security threats. These are not very sophisticated techniques like the ones white box testers use – but when you compare the confidence levels between a product with no security testing and some security testing – we would definitely prefer the later.

Let us look at the test ideas a black box tester can deploy to test for security threats. This is from a tester's point of view and is expected to serve as a basic guide for anyone just starting on AJAX testing journey. Once you get the initial idea, you will start getting alternative ways in which you can test – with such new technologies there is one key to success and that's – experimentation with your testing.

So lets start – I will focus on two main threats that can be tested with black box approach:

AJAX Data Validation:

Let's start by first seeing how data validation becomes a threat in the first place. Actually, I should rather say, how 'lack of data validation' becomes a threat.

I will keep it as simple as possible so that everyone understands – if the seriousness of threat is not understood, then we possibly cannot get enough test ideas.

We all use forms and edit boxes to gather user input in an application – it can be authentication form, search engine or a sophisticated eCommerce website. Sometimes we forget to fill in data in a field or enter data that is not within acceptable limits – what happens - the application throws up an error message – "Dude, enter the data / correct the data before you can submit this form". Alright, the error message is much more formal that this – but you get the idea.

That's data validation that got the error message to popup. This ensures that your application does not get any invalid data, intentional or unintentional, as inputs that can invite you trouble.

The hacker will try to find out what data validations are not in place and try to exploit that to pass in invalid inputs and if he is lucky, your system will give him some output which can be anything from directory structure, email folders to password files.

The hacker can get the data displayed in the web page itself, get it mailed to his id or worse – give some command to the operating system and processes to restart, format etc.

Testing Strategy

Here are some of the test ideas to start off with data validation testing. Please not that these do not eliminate the need for full fledged security testing – it's just a means for a black box tester to test something instead of nothing.

- **Test the pillars** – here you test all the field & form level validations for the following validation:
  - Mandatory fields - are they really mandatory
  - Invalid data input – enter text for a field that is supposed to accept numbers only
  - Test field dependency – city field becomes mandatory if you select data for country field
  - Check the validation fired upon Submit / Save – any event that is supposed to pick data from the form and send it to server
- **SQL Keyword test ideas** – once you are satisfied with the first step, carry out some SQL injection attacks on the form and observe the result
  - Enter the following special characters into the field:
    - Comma (,)
    - Semi Colon (;)

- Apostrophe (‘)
- Double Dash / Hyphen (--)
- Ampersand (&)
- Forward Slash (/)
- Asterisk (*)

Each of these special characters has special meaning in relation to a database. When you pass these in fields to the server, it can throw up database error if not handled smartly by the developer. Ideally these characters should be ‘filtered’ out by the developer before the field data is submitted to the database.

Let’s take an example how lack of such validation can help a hacker.

On our form we have a field known as “Age”. This field is supposed to accept age details of the participants who are going to attend a program.

Our developer has not developed any filtering code to ensure that no unwanted data is passed to the database server. When we pass the any of the above mentioned data, the database does not know what result it is supposed to deliver. It does not even know what type of data is coming in as it’s programmed to accept numbers and only numbers.

We get the following error message (*Reference - OWASP Testing Guide – Page # 152*)

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark before the
character string ''.
/target/target.asp, line 113
```

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the
varchar value 'test' to a column of data type int.
/target/target.asp, line 113
```

These error messages tell us a lot about the data base schema and field, its validation and any user details if available. With this information, a hacker can easily modify the input data to generate a condition that fools the database in showing up some confidential data on screen – or mailing it to the hacker’s email id.

When ever we see such error messages or unhandled error states, we need to highlight it to the development team of a potential security risk.

AJAX Session Management

What is a 'Session'? This is something all of us have heard at some point of time – in any sphere of life. Internet chat sessions, tennis sessions and yoga sessions – all these depict your relation to the specific activity for a specific period of time – and this is what we call a 'Session'.

Let's take a real life example. Your algebra class occurs between 2 – 3 PM. You reach the class at 5PM. Your professor checks your name in his log and then checks the batch time. He finds that you are not in the 5PM batch and you are not allowed entry in the class – this is because your session has ended at 3PM.

Internet application sessions are no different from the real life example that we just saw. Only difference is that your professor has got the decision making power to decide if the session rules can be overridden. Internet applications have lot more checks and validations and are not as sparing as a human when it comes to security – once you are out, you are out.

Internet application sessions are designed in a very sophisticated manner – code named: "Cookies". These cookies are obfuscated like a classified code word from a James Bond thriller so that no one can easily interpret the data!! If this is so hi-tech, how does a black box tester test these sessions without any idea of algorithm and underlying code?

Let us have a look at some black box test ideas that can be used for basic session testing.

Testing Strategy

We can test session cookies using black box techniques, but as with data validation testing, there are lots of things more to session testing that cannot be tested without getting hands dirty with code.

Here are some of the test ideas for black box testers that can be used when there is no formal plan / budget for professional security testing.

Test Ideas

- Test for timeout of active session
- Test for number of users that can login at a time
- Any actions under which users is thrown out of a system
- Try accessing a secured page from a bookmark
- Test the 'remember me' or 'remember my user name' options
- Test the 'login automatically' functionality
- Test the URL for passing variables directly
- Test the URL for session ID display
- Test if session ID is abstracted and unpredictable
- Test if the session IDs can be reproduced
- Test if the same user logs in twice, the session id for each login is unique
- Test if the static part of session ID are easily recognizable
- Test than no sensitive information is passed as clear text in session ID
- Test to find if same inputs produce same session IDs
- Try to identify a pattern in session IDs for same login

The answer to all of these questions should ideally be in negative – stating that the basic session security mechanism is in place. The intent of such testing is to validate that session IDs are generated in a protected and a non-predictable manner. An attacker who is able to sniff out the details from a session ID can easily gain access to sensitive information of users and customers.

Conclusion

Security testing of RIA's is of most important and we should have in-depth security testing done for such applications. If we don't have talent or budget for such testing, we can do some security testing using black box methods.

This paper showed some of the most basic test ideas a functional tester can try out to catch vulnerabilities in their RIA applications. This testing is basic and given a chance should be supported with professional security testing services.

References:

These are some of the documents and blogs that I referred while working on this article. These all have some excellent information on AJAX and RIA applications.

1. http://en.wikipedia.org/wiki/Ajax_(programming)
2. http://www.forrester.com
3. http://www.owasp.org/index.php/AJAX_Testing_AoC
4. http://blogs.zdnet.com/Hinchcliffe/?p=65

About the author

Aakash Vakil is a CSTE with over six years experience in functional and automated testing. He has worked as test lead, test specialist and test engineer servicing large banking and financial service industry clients.

Aakash like to research on latest developments in technology areas with a focus on testing, search engine optimization and web analytics.

When he is not testing, he likes to blog about his views and observations about everyday life focusing around testing.

His blogs can be read at www.sqablogs.com/aakashvakil and he can be contacted @ avakil@deloitte.com