I.J.Modern Education and Computer Science, **2012, 1, 33-39**
Published Online February 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2012.01.05

Finding Vulnerabilities in Rich Internet Applications (Flex/AS3) Using Static Techniques

Sreenivasa Rao Basavala
Dept. of Computer Science & Engineering, CMJ University, Shillong, India
Email: basavala@gmail.com

Dr.Narendra Kumar
Dept. of Computer Science & Engineering, Ace college of Technology and Management, Agra, India
Email: narendra.ibs@gmail.com

Dr.Alok Agarrwal
Dept. of Computer Science & Engineering, JIIT University, Noida, India
Email: alok289@yahoo.com

***Abstract*--- The number and the importance of Rich Internet Applications (RIA) have increased rapidly over the last years. At the same time, the quantity and impact of security vulnerabilities in such rich internet applications (RIA) have increasing as well. Since manual code reviews are time consuming, error prone and costly and it need skilled developers or programmers to review the manual source code review, the need for automated solutions has become evident.**

In this paper, we address the problem of application security vulnerable detection in Adobe Flex (Rich Internet Applications) platform in web 2.0 applications by means of static source code analysis. To this end, we present precise analysis targeted at the unique reference semantics commonly found in RIA based web applications or widgets (small applications which will run on fly i.e. drag and drop) developed in Adobe Flex Framework or Action Script 3.0. Moreover, we enhance the quality and quantity of the generated vulnerability reports.

***Index Terms*: Applications security Analysis, Static Analysis, Taint Analysis, Vulnerability Detection in Flex, Static Techniques, Action Script Security.**

I. Introduction

With Web 2.0 technologies like Adobe Flex/Flash and web services being all the rage, Rich Internet Applications (RIAs) are popping up everywhere. More developers are creating rich apps called widgets in-house.

Software vulnerabilities provide a way to an attacker attack or hack the application. Vulnerabilities are the well-known and well understood software flaws by the application developers. For example Cross-Site Scripting vulnerability is most common and well known of security issue. In order to identify these kinds of vulnerabilities in a web application a comprehensive analysis (either dynamic or static) is required to develop standard solutions against security vulnerabilities.

As vulnerability or security issue refers to a weakness in software. Now the question will arises that what is a weakness of software or an application? The main reason of vulnerabilities is due to improper development of a software developer; an attacker can take advantage from this vulnerability of an application and execute or run the commands of the desktop or web application or bypass some access control list (ACL).

To identify these kinds of security issues (vulnerability) we may need some techniques and tools to discover and/ or to remove vulnerabilities in a desktop or web applications. For example Fortify, Web Inspect, Rational Appscan, SWF Scan etc.

Vulnerabilities are present at least to some extent in every web or desktop applications, so it could not be neglected. Vulnerability can be identified and prevented and/or removed in application source code using different techniques.

Basically there are two major approaches to detect or prevent vulnerabilities such as static (source code) and dynamic (penetration testing) analysis of an application. Some tools can directly apply to the source code so they either solve or at least provide warn about presence of such vulnerabilities in the application source code. These tools are called static tools for example Fortify. The other types of tools called as dynamic analysis tools that check the software at runtime against any known vulnerability as we discussed in section VI.

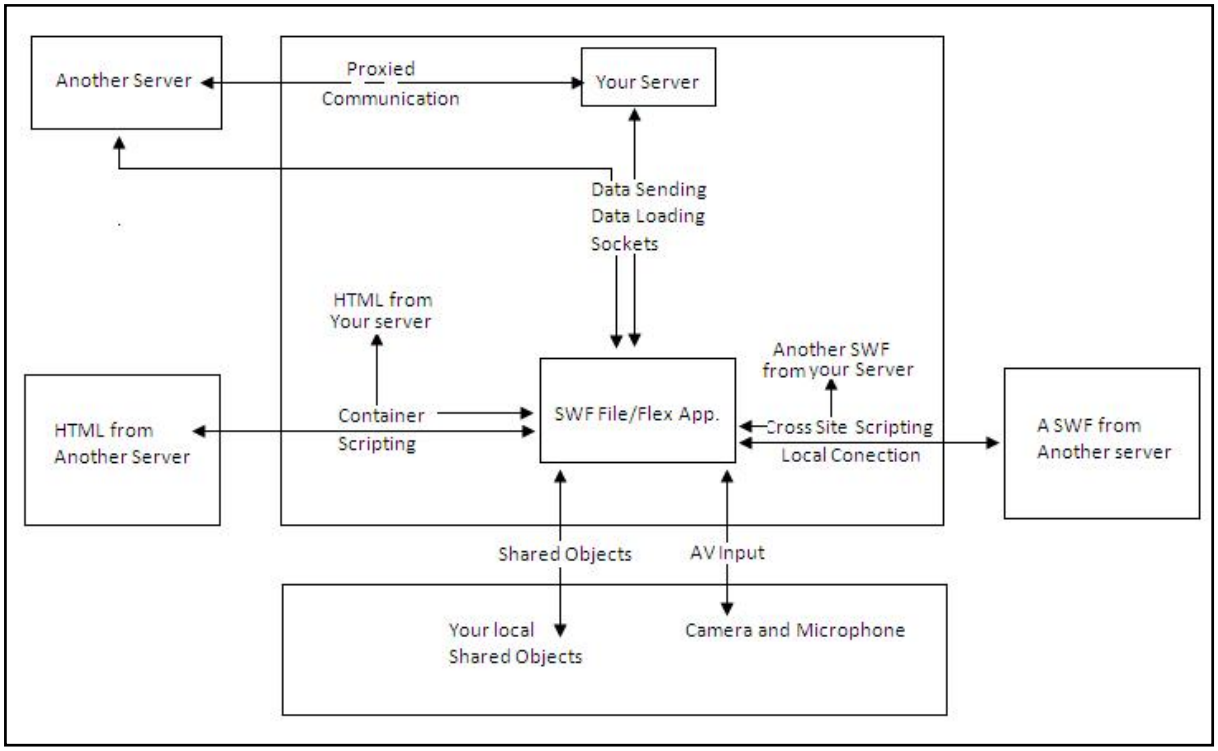

Figure 1: Security vulnerabilities in Adobe Flex applications

II. ADOBE FLEX

Adobe Flex [15] is basically a software development kit (SDK) (similar to JDK) released by Adobe Systems for the development Rich Internet Applications (RIA) [16] based on the Adobe flash platform. Adobe Flex applications can be written by using Adobe Flash Builder [17] called integrated and development environment (IDE) or by using the freely available flex compiler.

A Rich Internet Application [18] (RIA) is a normal web application that has many of the characteristics of web or desktop applications like drag and drop, rich in feel; rich in look and very impressive to the end users (feature from in web 2.0 technologies). In the recent years RIAs dominate in online gaming as well in the market.

II. RELATED WORK

Currently, exists only few approaches that deals with static detection or analysis of web application vulnerabilities. Huang et al. [11] were the first to address this issue in the PHP based applications. They used a lattice-based algorithm derived from type systems and type state, and compared it to a technique based on bounded model checking in their follow-up paper [12]. In this paper we have chosen regex (regular Expressions) patterns that deals with static detection of RIA web applications vulnerabilities developed in Adobe Flex framework with action script 3.0. This document continues to our effort in the area of rich internet applications security static analysis.

Thus, the analysis provided in this document aimed to evaluate the regex patterns in static analysis our original intention was to provide automatic testing procedures or patterns for the best practices. As far as we know, our analysis is the first analysis of this kind in RIA framework. However, there are several analyses of bug finding static tools for Java but not in rich internet applications such as Adobe Flex.

III. FLEX API

In Adobe Flex Framework the API which is listed below leads to security vulnerabilities [7] is as follows:

A. .htmlText property
B. externalInterface.call()/addcallBack()
C. navigateToURL()
D. SharedObject.getLocal()/getRemote()
E. trace() debug statement

Motivating Example: As we listed all vulnerable methods in the adobe flex framework. Some examples of how developer uses these APIs while developing web applications are given below.

A. .htmlText property: data displayed in the .htmlText property is vulnerable to attacks like XSS [10] [4] (cross site scripting), etc. To prevent these attacks, data should be encoded before it is assigned to the htmlText attribute. The following component property may leads to XSS attack in the flex in RIA application:

Functions that display HTML code:

- TextField.htmltext
- Text.htmltext
- Lable.htmltext
- TextArea.htmltext

Syntax:
<mx:Text htmlText=" your HTML content here">
<mx:Label htmlText=" your HTML content here">
<mx:TextArea htmlText=" your HTML content here">

Sample code:
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml">
<mx:Script>
 <![CDATA[

     private function myTest():void  {
    test1.htmlText = "Hello
<script>alert(document.cookie)</script>;
     }
 ]]>
</mx:Script>
 <mx:VBox>
  <mx:TextArea id="test1" />
 </mx:VBox>
</mx:Application>
```

Figure 2: Syntax and sample code of .htmlText property

B. ExternalInterface.call()/addcallBack(): To communicate between ActionScript and the container of an application can take either ActionScript can call code (such as a JavaScript function) defined in the container, or code in the container can call an ActionScript function that has been designated as being callable. In either case, information can be sent to the code being called, and results can be returned to the code making the call.

Functions that communicate with the web browser:

- ExternalInterface.call()
- ExternalInterface.addCallBack()

Here "Sample java script to code" verify for the word *to:*

The java script function needs name as input parameter/argument as follows:

```
<script type="text/javascript">
        function submitInfo(name) {
                document.write(name);
        }
</script>
```

Syntax of ExternalInterface.call():

flash.external.ExternalInterface.call(function_name:String[, arg1, ...]):Object;

By using ExternalInterface.call () method for invoking java script code as:

```
Protected function clickHandler(name:String):void {
        ExternalInterface.call('submitInfo', name);
}
```

Sample Code:
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml" >
<mx:Script>
        <![CDATA[
protected function clickHandler(name:String): void
        {
ExternalInterface.call('submitInfo', name);
}  ]]>
</mx:Script>
<mx:FormItem label="First Name: ">
<mx:TextInput id="firstNameTextInput"/>
</mx:FormItem>
mx:Button label="Submit"
click="clickHandler(firstNameTextInput.text)"/>
</mx:Form>
</mx:Application>
```

Figure 3: Sample code of ExternalInterface.call ()

The code will execute in the browser and the output of the above code shall be as follows:

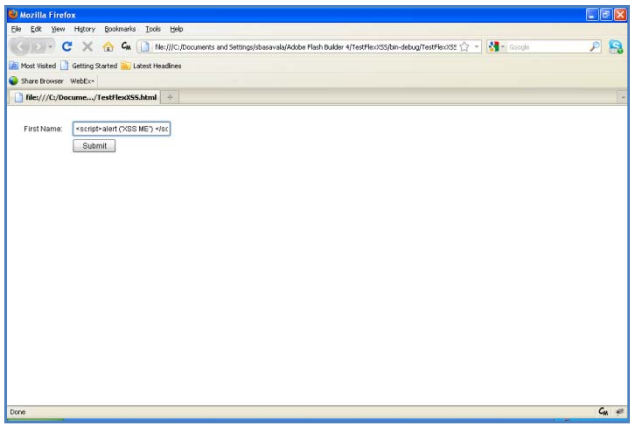

Figure 4: Browser view of MXML content with ExternalInterface.call ()

In the textbox we have entered *"<script>alert ('XSS ME') </script>"*. Script will execute and display the popup message as shown below:

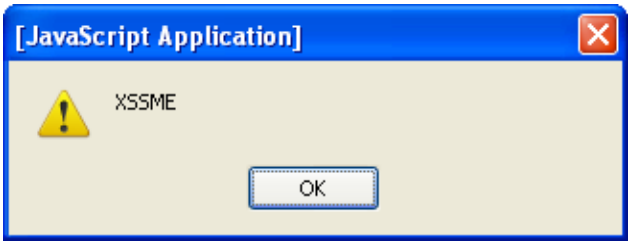

Figure 5: popup message or value of vulnerable API

.

C. *navigateToURL()*: The navigateToURL() method loads a document from a specific URL into a window or passes variables to another application at a defined URL. Also you can use this method to call JavaScript functions in the HTML page that encloses an application. These functions may prone to URL redirection or forward issue. Functions that make networking calls are:
   - navigateToURL()
   - sendToURL()

   Functions and objects that accept URLs:
   - URLRequest()

Syntax & Sample Code:
navigateToURL (request: URLRequest, window:String):void

```
<fx:Script>
 <![CDATA[
    import flash.net.*;
    public function openNewWindow (event:
    MouseEvent):void {
    var url:URLRequest = new URLRequest
                "http://www.website.com/wel");
     navigateToURL(url,"_blank");
    }  ]]>
</fx:Script>
```

Figure 6: Syntax and sample code of navigateToURL()

D. *SharedObjects:* Shared objects, same as browser cookies in normal web application and stored at client system. SharedObject class is used to store data (cookie) on the user's local hard drive in clear text and will call that data during the same session or in a later. Web applications can be able to access only their own SharedObject data or cookie and only if they are running on the same domain. Functions for accessing SharedObjects:
   - SharedObject.getLocal()
   - SharedObject.getRemote()

Syntax:
SharedObject.getLocal("objectName" [, pathname]): SharedObject

Sample Code:

```
Var  my_so:SharedObject = SharedObject
                              .getLocal("superfoo");

var my_so:SharedObject = SharedObject.
                              getRemote ("superfoo");
```

Figure 7: Syntax and sample code of SharedObject

E. *Trace ():* Trace function is used for debugging the application which prints the message on the console when application runs in debugging mode. Function to debug the application

Syntax: trace();

Sample code:

```
 <mx:Script>
<![CDATA[
        import mx.controls.Alert;
        public function init():void{
        trace("Init started.....");
        myText.text = "Button Clicked.";
        trace("Init ended.....");
        }
```

Figure 8: Syntax and sample code of trace()

IV. TECHNIQUES

Basically the techniques used to identifying, detecting and/or removing vulnerabilities in two categories so called as static analysis [9] (source code analysis) and dynamic analysis (penetration testing). In this paper we will more concentrate about static techniques or analysis and then we will discuss a brief description of how these static techniques to identify, detect, and prevent or remove security vulnerabilities in the web or desktop applications.

Software program analysis (also called static code analysis) is the analysis of computer software (web application) that is running without actually executing programs built on that software. All most all the times the analysis is performed on some part of the source code of an application and in the other part is some form of the binary code. Finally system will provide the vulnerable report with lot of false positives. An application security engineer needs to review the

report and will identify the actual vulnerabilities in the computer software or a web or desktop application.

Dynamic analysis is also called black box testing to analyze computer software programs and applications without knowing the functionality of applications. To make dynamic analysis is more effective, the target application or test program must be executed with sufficient test data as inputs to produce sufficient output and behavior of the application as well. Using different software testing techniques such as code coverage (static program analysis) helps to ensure that an application behavior and set of possible behaviors with respect to security has been observed closely.

A. Static Techniques

In static code technique [9] or analysis basically an application source code is analyzed in order to find security vulnerabilities in application. The source code is scanned against the known vulnerabilities in the application level and a tool should implement with static techniques to detect the existing vulnerabilities in the source code. In the static code analysis there are two disadvantages of this approach is that someone in the security team to keep an update and maintain the database of programming flaws (known vulnerability) to test for, and since the static tools only detect vulnerabilities the user should knows how to fix the problem once a warning has been issued. In static code analysis source code is scanned before compilation against known vulnerabilities. The static code analysis techniques will addresses problems something like array bound check, uninitialized variables, improper error handling, sensitive information in the log messages, unreachable code, syntax problems, undeclared variables, parameter type mismatch, uncalled function and procedure, non-usage of function results etc.

Static source code analysis [9] [14] do not covers all vulnerabilities within the source code because of tool implementation depends on pattern and there are blind to certain types of vulnerabilities and problems that are only apparent at application runtime. To fill this gap, dynamic analysis will help us from past few years, as well as hybrid analysis which combine both static and dynamic analyses [14].

Dynamic and static analyses [14] to identify vulnerabilities are complementary technologies with different strengths and intended for slightly different audiences.

As we are aware of that static analysis is growing commercial use in the verification of properties of software used in safety and critical computer systems and identifying potentially vulnerable code in the computer software, web or desktop based applications.

a. Pattern Matching

Pattern matching is technique is used to find out of all pattern matches of given string to match or identify the risk. For example .htmlText that displays HTML (Hyper Text Markup Language) code in the source code.

A pattern matching is cannot identify partly from source code and is easily fooled by unexpected white space. Using this technique we will get more false positive results. Searches the given input files for lines containing a match to a given pattern list. When it finds a match in a line, it copies the line to standard output or file. Tool that implements this pattern match technique is security flaw finder which detects or identifies vulnerability by using pattern matching technique. Tool analyzes the source code and checks it to figure out XSS issues, format string or integer vulnerabilities, insecure cookies etc. by using its database for Flex functions and produce a list of flaws sorted by risk level.

b. Lexical Analysis

Lexical analysis creates an identifier stream of the source code in order to make a difference between variables of a function and to identify arguments of a function. These tokens are matched with existing vulnerabilities. Basically lexical analysis is used to improve the accuracy and correctness of pattern matching technique, because a lexer can handle irregular whitespace and code formatting. Lexical analysis techniques are fast and simple, but very limited since they do not take into account of the syntax or semantic technique of the program. The benefits of lexical analysis are very less and it's reported the more number of false positives by this technique.

Parsing technique or syntactical analysis is used to parses or analyzing the source code and it builts an abstract syntax tree representation of the source code. The abstract syntax tree allows us to analyze not only the syntax but also the semantics of a program. The pattern matching technique can be significantly improved by matching abstract syntax trees instead of sequences of matches or characters. This technique creates a complete and easy to navigate representation of a program.

c. Regular Expressions

Regular expression is also called as regex or regexps is special text string for describing a search pattern. A regular expression may contain search string with wildcards and special chars. e.g. regex "\b[A-Z0-

9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b" will search email-id in the source code. Typically a similar regular expression can be used to check if the user entered a properly formatted email address or not. One line of code, whether that program is written in Perl, PHP, Java, a .NET language or any of other programming languages. When writing computer software code regex is used to find/replace text in some code, computer programs can use regexps to translate code from one form into another form.

In computing regular expressions to provide a concise and flexible for identifying search strings of text such as particular characters, words, or patterns of characters. Regular expressions are written in a formal language that can be interpreted by a regular expression processor, or a program that either serves as a parser or examines text and identifies parts that match the provided specification.

A regular expression to detect or identifying the vulnerable code or security risk during static code analysis (SCA) of Adobe Flex/Flash or action script 3.0 applications is as follows:

a. htmlText=\\s*\\bhtmlText\\b\\s*=
b. ExternalInterface.call=\\s*\\=\\s*\\bExternalInterface\\b\\.\\bcall\\b\\s*\\(
c. ExternalInterface.addCallBack=\\s*\\=\\s*\\bExternalInterface\\b\\.\\baddCallBack\\b\\s*\\(
d. URLRequest()=\\s*new\\s*\\bURLRequest\\b\\s*\\(
e. navigateToURL()=\\s*\\bnavigateToURL\\b\\s*\\(
f. URLLoader=\\s*new\\s*\\bURLLoader\\b\\s*\\(
g. SharedObject=*new*\\bSharedObject\\b\\s*\\
h. trace=\\s*\\btrace\\b\\s*\\(

d. Comparison and Evolution

As we deal with static analysis using regular expressions to detect or identifying security vulnerabilities in Adobe Flex application. Static code techniques have several advantages over run-time techniques. Static techniques find errors by analyzing the source code and do not require running the program. They do not incur run-time overhead and they narrow down the vulnerabilities specific to the source program being analyzed, yielding a more secure program before it is deployed. However, a pure static analysis can produce many false alarms due to the lack of vulnerabilities related information.

V. OTHER VULERABILITIES IN FLEX

In this paper we described only static analysis to detect, identify security issues in RIA (developed using adobe flex and action script). But there are other security issues in web application developed in Flex. These vulnerabilities we can exploit through the dynamic analysis (security testing or penetration testing) of an application at run time. Here is the list other vulnerabilities which we can identify, detect or exploit using dynamic analysis as follows:

a) Cross-domain privilege escalation
b) Spoofing
c) Malicious data injection
d) Script injection into the browser
e) Insufficient authorization restrictions
f) Unauthorized access to data in transit
g) Unauthorized local data access
h) Cross-site request forgery
i) DNS rebinding
j) Click jacking

Along with these vulnerabilities in adobe flex or as3 applications there are other PDNF (Potentially Dangerous Native Function) functions.

V. CONCLUSION

It is very difficult that a static tool completely detect all the security vulnerabilities without false positives. As more and more software's are developing day by day vulnerabilities are also growing. Although significant work is done to cope with XSS [4] (cross site scripting) issue, insecure cookies, invalidated URLs and other security vulnerabilities but we still need a satisfactory solution. Security can be improved if vulnerability detection tools either static or dynamic analysis is used as a part of software development lifecycle. We have discussed different static techniques and tools to detect XSS issues, URL redirection and forwards, cookies or shared objects and debug statements. We will explore section V vulnerabilities in my next paper through the dynamic analysis with penetration testing.

VI. ACKNOWLEDGEMENT

I owe a great many thanks to great many people who helped and supported me during writing this paper and exclusively to my colleague Satya Kumar.

My deepest thanks to my co-author Dr.Alok Agarrwal and my guide Dr.Narendra Kumar for guiding and correcting various parts of this document with attention and at most care. He has taken care to go through this paper and make necessary correction as and when needed.

REFERENCES

[1] Gagan Agarwal, Jinqian Li, and Qi Su. Evaluating a demand driven technique for call graph

construction. In *Proceedings of the International Conference on Compiler Construction*, May 2002.

[2] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley.

[3] Ken Ashcraft and Dawson Engler. Using programmer-written compiler extensions to catch security holes. In *Proceedings of the Symposium on Security and Privacy*.

[4] Amit Klein. Cross site scripting explained. *http://crypto.stanford.edu/cs155/CSS.pdf*

[5] Yao-Wen Huang, Fang Yu, Christian Hang, Chung-Hung Tsai, Der-Tsai Lee, and Sy-Yen Kuo. Securing Web application code by static analysis and runtime protection. In *Proceedings of the Conference on World Wide Web*. May 2004.

[6] *http://www.adobe.com/devnet/flex/articles/flex_enterprise_security.html* by Adobe.

[7] Nenad Jovanovic, Christopher Kruegel, and Engin Kirda. Precise alias analysis for syntactic detection of Web application vulnerabilities. *In Proceedings of the Workshop on Programming Languages and Analysis for Security*, June 2006.

[8] Brian Chess and Gary McGraw. *Static analysis for security. IEEE Security and Privacy*, 2(6):76–79, 2004.

[9] Jeremiah Grossman. Cross-site tracing (XST): the new techniques and emerging threats to bypass current Web security measures using TRACE and XSS. *http://www.cgisecurity.com/whitehat-mirror/WhitePaper screen.pdf*

[10] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo. *Securing web application code by static analysis and runtime protection*. In WWW '04: *Proceedings of the 13th International Conference on World Wide* Web, 2004.

[11] Ss Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D. T. Lee, and S.-Y. Kuo. *Verifying web applications using bounded model checking*. In DSN, 2004.

[12] *Industrial Perspective on Static Analysis. Software Engineering Journal* Mar. 1995: 69-75Wichmann, B. A., A. A. Canning, D. L. Clutterbuck, L. A. Winsbarrow, N. J. Ward, and D. W. R. Marsh. http://www.ida.liu.se/~TDDC90/papers/industrial95.pdf

[13] http://www.slideshare.net/eladnyc/top-security-threats-to-flashflex-applications-and-how-to-avoid-them-4873308

[14] http://www.cc.gatech.edu/~orso/papers/halfond.choudhary.orso.STVR11.pdf

[15] http://en.wikipedia.org/wiki/Adobe_Flex

[16] http://en.wikipedia.org/wiki/Rich_Internet_application

[17] http://en.wikipedia.org/wiki/Adobe_Flash_Builder

[18] *Rich* Internet Applications: The Next Frontier of Corporate Development" by Larry Seltzer. 2010-08-25. eWeek.http://www.eweek.com/c/a/Security/Rich-Internet-Applications-The-Next-Frontier-of-Corporate-Development-732651

[19] Laszlo: An Open Source Framework for Rich Internet Applications

Mr. Sreenivasa Rao Basavala, M.Sc., M.Phil. M.Tech. SCJP, SCWCD, IBM-ACSE is a Sr.Application Security Engineer in Department of Yodlee Security Office. He has over 11 years of experience in IT industry and Academic. His areas of interests are Web Application Security, Software Engineering, Computer Networks, Cryptography, Mobile Application Security, Information Security, Database Security, DBMS and RDBMS. His area of research interest is Web Application Security, Mobile Application Security, Security code reviews and penetration (security) testing in various domains. Currently he is researching in Web Application Security under guidance of Dr.Narendra Kumar from CMJ University, Shillong, India.

Dr. Narendra Kumar, M.Tech., Ph.D., is former professor and Head, Department of Computer Science and Engineering. He has over 18 years of teaching experience. His areas of interests are Theory of Computation, Wireless Communication, Cryptography, Computer Graphics, Securities and Applied Mathematics. He has good number of research publications in international/ national journals. He is member editorial board and referee for various journals. He is associated with reviewing books of international publications. His area of research interest is Wireless Communication, Security in various domains and Mathematical modeling. He is research supervisor for more than a dozen students in various universities. He worked as visiting faculty for various engineering colleges/ universities.

Dr. Alok Aggarwal, received his graduation and post graduation degrees in Computer Science and Engineering in 1995 and 2001 respectively. He received Ph.D. degree in Mobile Computing discipline in 2010 from IIT Roorkee. Currently he is with Computer Science and Engineering/Information Technology Dept. of JIIT University, Noida. He has edited three books and about 50 research papers published in various International/National journals, conference proceedings. He has a mix experience of Industry, Research and Teaching of about 15 years. His area of interest is Mobile Computing and Object Oriented Programming.

