International Journal of Computer and Electrical Engineering, Vol. 1, No. 4, October, 2009
1793-8163

A Platform Specific UML model for Web application self defense through an Aspect Oriented Approach

Dhanya Pramod, Vinay Vaidya

***Abstract* – Invention of internet has paved a way for popularity of Web based applications. Web application vulnerabilities are a major concern for critical applications. We bring forth the idea of making applications self defendable through aspect oriented approach of code modification. Aspect Oriented programming and modeling has been accepted as it facilitates integration of cross cutting concern without any change to the existing application and also facilitates separation of nonfunctional concerns in applications that are under development. We propose a Platform Specific Model{PSM) in java using aspect oriented approach for securing web applications against most popular cross site scripting, sql injection, authentication, authorization parameter tampering and session hijacking attacks. The main focus in this paper is the description of various classes in the PSM model. Each and every security countermeasure is modeled as class in the PSM model and act as an aspect in aspect oriented modeling. We have shown the relationships between various other components of aspects like pointcut and advice. The paper also presents an excerpt of the model implementation of aspect oriented countermeasure using aspectJ. The paper also gives an excerpt of algorithm devised for session protection. Running web applications were tested before and after the aspect injection and test results are given to prove the approach.**

***Index Terms* – Web application, Aspect Oriented Approach, Platform Specific Model, Countermeasure.**

I. INTRODUCTION

The internet and web-based applications allow groups of users in different locations to harness the massive computing power collaboratively for the benefit of humanity. In this era of collaborative computing and networked and shared web applications, ensuring the safety and privacy of data stored in computers and transmitted over the internet has become critically important.

Both the at-large attacks on computers and targeted attacks on specific computer networks have become so sophisticated that general-purpose IDS (Intrusion Detection System) mechanisms like signature-based antivirus packages have been rendered inadequate. The mutating and polymorphic varieties of viruses can defeat purpose-built IPS (Intrusion Protection System) mechanisms and with hackers developing the ability to launch zero-day attacks using easily available exploit frameworks, the time has come to confront the security problems by improving the immunity of the target software itself.

It can be assumed that the Operating System (OS) and Network Protocol-level software is relatively secured. However the same cannot be said about application software. If we split application software into end-user and web-based, it is easy to see that while security breaches in end-user applications can cause corruption of data on the computers where the applications are installed, security breaches in web-based applications have the potential of affecting large swathes of networked computers alarmingly quickly over the internet. Cross-site scripting, parameter tampering, buffer overflows, SQL injection are some common causes that result in authentication and authorization breaches in web-based applications.

This paper concentrates on the problem of securing web-based applications (called 'web applications' henceforth). It is proposed to treat security as an aspect that is to be incorporated at the design stage of development of the web application. This approach to secure web applications can be applied to existing applications also. These security concerns are to be integrated throughout the software and thus Aspect Oriented Modeling is a better way to do this. In this approach we design the security model separately and then weaved to the base model. It is further proposed to use a model-driven development approach for this purpose. We have developed a Platform independent UML class diagram to model the solution. Also conversion to Platform Specific Model is done. ie. For applications developed in Java servlet technology aspects have been designed using aspectJ.

The following part of this paper contains overview of web application attacks, countermeasures modeled as aspects represented using UML model and excerpt of aspectJ implementation of aspects.

II. METHODOLOGY

Web applications security threats that are reported as the top vulnerabilities are injection attacks (cross site scripting, sql injection), authentication, authorization, parameter tampering, buffer overflows and session hijacking. We use Aspect Oriented approach to handle the above mentioned vulnerabilities.

Manuscript received June 1, 2009.

Dhanya pramod is a research student at Symbiosis International University and working as an assistant Professor at Indira Institute of Management, Pune-India.

Vinay Vaidya is an advisor at Symbiosis International University, Pune-India.

A. *Aspect Oriented Approach*

Aspect Oriented Modeling has aspect, advice, point cut and joinpoint as the fundamental elements. Aspect is a nonfunctional concern that is scattered across multiple modules or classes of a system and is put together with other functional concerns. Joinpoint is a well defined point in the structure or execution of a system and can be called as a static joinpoint and dynamic join point respectively. Collection of joinpoints is referred to as point cut. Advice is the behavior that is injected by the aspect to the joinpoints. The beauty of AOM is the injection of advices before, after and around joinpoints during compile time or runtime. This is accomplished by a process called weaving. In this paper we describe these elements of security aspects.

B. *Overview of countermeasure aspects.*

Table 1 shows the list of aspects we have designed to address top listed attacks. For every attack there is an aspect class which has the pointcut. Every advice is modeled as another class.

C. *Model Driven Approach*

In order to facilitate compliance to model driven engineering an aspect model that is independent of the specific technological platform used to implement it, has been defined. This would help those who wants to use a Model Transformation Language to transform a Platform-independent model into a Platform-specific model.

a. *PIM Aspect meta model*

The PIM model contains various components that constitute the security countermeasure aspect.

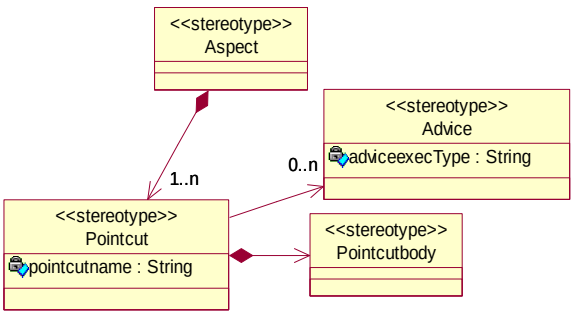


Figure (1) Platform Independent metamodel

b. PIM Aspect model using UML

The PIM model defines 12 aspect classes and 12 advice classes . Every aspect class contains one or more pointcut classes. Following are the aspect classes and related pointcuts

c. *Pointcut identification*

A general method has been identified to find various pointcuts of every aspects that are defined.

i. SqlInjectionAspect

The following pointcuts are identified

- All user input points
- All database access points. Methods that retrieve or update database contents

ii. LoginTracer

The following pointcuts are identified

- Login verification method
- Home page/Error page firing method

iii. CredentialCheck

The following pointcuts are identified

- Method that update user password
- Method that takes new password

iv. DataValidation

The following pointcuts are identified

- All user input points

v. RuleValidation

The following pointcuts are identified

- Data process/populate method

vi. BufferSize

The following pointcuts are identified

- All input points

vii. XssAsp

The following pointcuts are identified

- All input points

viii. EnsureConfidentiality

The following pointcuts are identified

- All page request handling methods
- All page requests
- Session load/create methods
- Setting cookie as header information
- Setting cookie using methods
- Logout method

ix. SessionAspect

The following pointcuts are identified

- All page request handling methods
- Session start/load method
- Login verify method
- Form response method
- Logout method

x. FileCheck

The following pointcuts are identified

- All file upload methods

xi. Encryption

The following pointcuts are identified

- Database Update method

xii. Decryption

The following pointcuts are identified

- Data retrieve method

.

d. *Platform Specific Model*

The platform Independent Model which has been proposed in the previous section can be transformed in to a model for any specific platform. The platform specific model is then used to implement the system in the desired technological platform (programming language, operating system, database).

Figure(2) shows the PSM model representation using UML

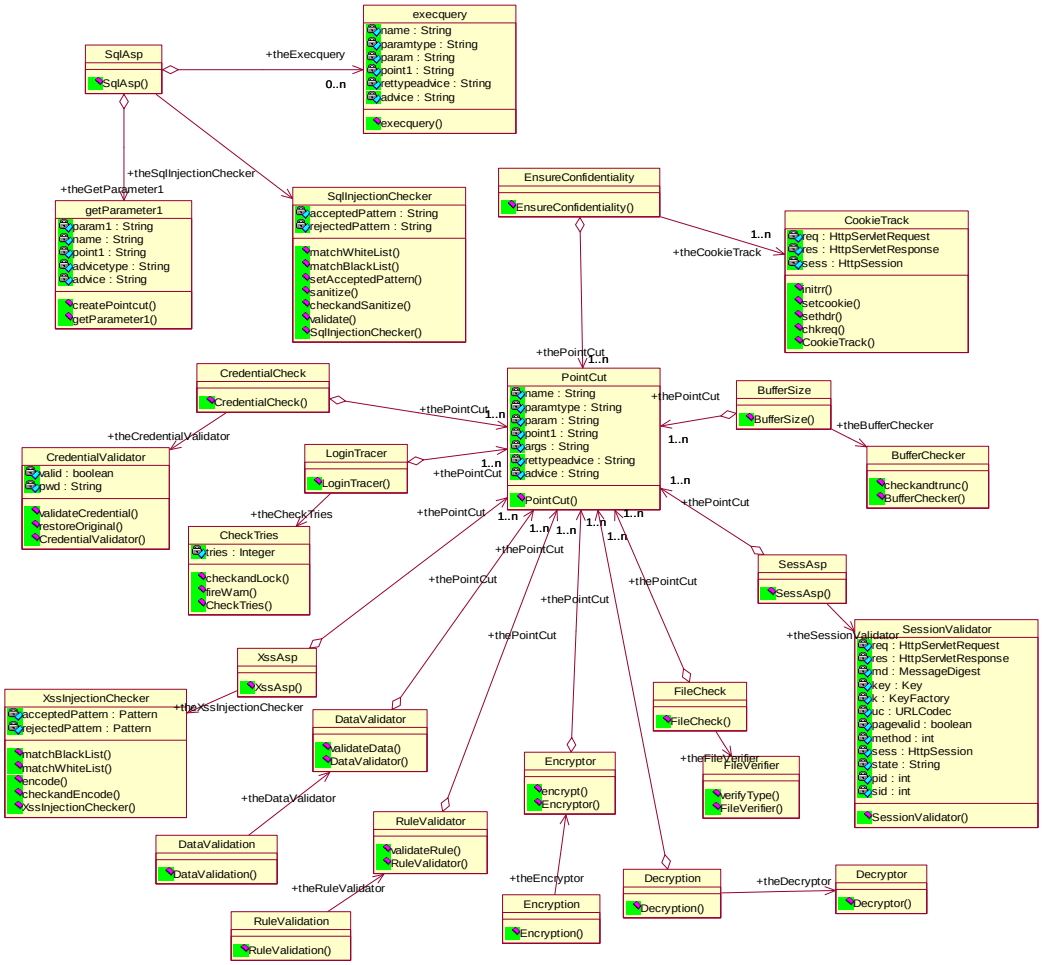


Figure (2) Java Platform Specific Model-Aspect UML model

TABLE I ASPECT DETAILS

Sr.no	Attack	Pointcut	Advice
1	Sql Injection	All pages user input methods	Search for malicious pattern using regular expression. Strip malicious pattern
2	Cross site scripting	All pages user input methods	Search for malicious pattern using regular expression. Strip malicious pattern
3	Authentication	Start of every page	Check whether authenticated. If not fire login page
		Validate method	Track failed logins by incrementing tries counter.
		Login page start	Failed login check tries>3 lock user.
4	Authorization	Start of every page	Check for valid pageid in request to see whether user is allowed to view. If not Warnpage is fixed.
5	Parameter Tampering	Links and forms	Change parameter and its values with relative values before sending to client.
		Request	Relative values converted back to actual values.
6	Session/url manipulation	Links and Forms	Insert a parameter in the query string of <a href pageid
		Request	Get Pageid from query string. If previously sent to client then only accept. Otherwise manipulated i/p Warn page is fired
7	Buffer Overflow	All pages user input methods	Check against expected size. If greater then truncate.

D. Aspect implementation

For the proof of the concept implementation of aspect, the researcher had used application developed using java servlet technology and AspectJ for aspect implementation.

E. AspectJ overview

Aspectj is an aspect oriented extension to java. It supports the plug and play implementation of crosscutting concerns.

An aspect consists of an association of variables, methods, pointcut definitions, inter-type declarations and advice. AspectJ pointcuts may be named or anonymous. These pointcuts may pickup the interesting points in the execution of a program ie. method or constructor invocations and executions, the handling of exceptions, field assignments and accesses etc, and exposes some of the values in the execution context of those join points.. The researcher has mainly used the target, execution and call pointcuts for implementing the security countermeasure aspect pointcuts. The advice defines pieces of aspect implementation that execute at well defined points in the execution of the program. The advice can be of three types before, after or around. The before advice will get injected and executed before the join point, after advice will get injected and executed after the join point and around advice get injected before, takes arguments from the method call/execution modify it and return the arguments.

F. Aspect details

The following were the attacks the researcher had taken into account for application self defense. The identified pointcutand advices are given below.

G. Aspect ordering and weaving

In the efforts to incorporate security concerns in model driven development the researcher modeled them as aspects. Every aspect mentioned above has its components point cut and advice. Here we identified and modeled them in connection with the joint points in the base model. Aspects were ordered while weaving to the base model.

The pages that require direct intervention of user need to be protected against injection attacks. In the mentioned search article example search article form page accepts the input from user. XssInjectionCheck aspect is weaved here to filter the user input. This will ensure that the input does not have any malicious tag. Encode aspect is weaved in case of any flaw found in the input. The DataValidation aspect validates the data entered by the user and forward to searchArticleController object. SqlInjectionCheck aspect need to be weaved both at the interface layer and at the persistence layer. So every get*(…) method invocation needs incorporation of SqlInjectionCheck aspect and ExistenceCheck aspect (this will ensure the field is present in the database table). Each and every page request will check for confidentiality. LoginTracer aspect was weaved at the login module and credentialCheck was weaved whenever credentials are modified. Encryption/Decryption aspects were weaved duringdata store and retrieval respectively.

SessionAsp was weaved for every request and response. Given the default ordering of aspects below if they appear at the same pointcut.

1. LoginTracer
2. EnsureConfidentiality
3. SessionAsp
4. XssInjectionCheck
5. SqlInjectionCheck
6. Encryption/Decryption
7. CredentialCheck
8. RuleValidation
9. BufferCheck
10. FileCheck

H. An excerpt of some aspects

The Platform specific aspect implementation and pointcuts identification are mentioned here.

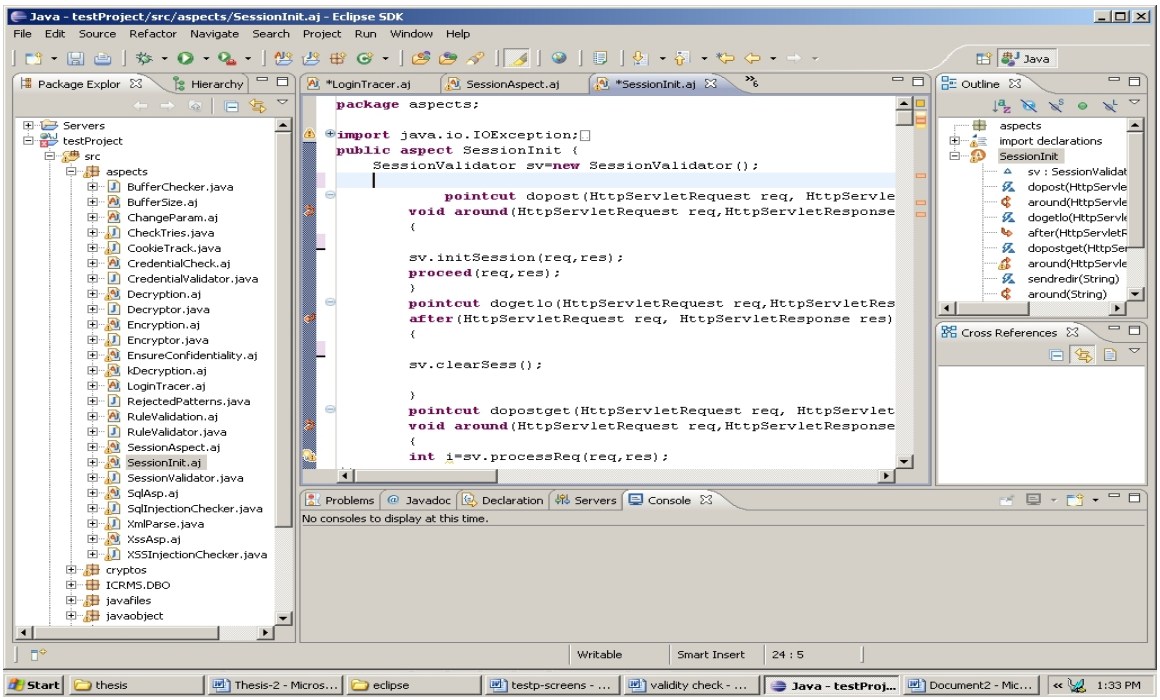


Figure (3) SessAsp aspect

Ø Aspect weaving

Hidden fields and rewritten url links are used to pass state information between pages.
The aspect is used to insert a parameter and value in url in case of url link and a hidden parameter and value in case of forms.

Ø Aspect pointcuts

à Sending response

1. Html page is formed inside the server side program.
2. Intercept the "<a href .." tag and insert additional parameter PAGEID and a generated value. If the session is new start generation of PAGEID, otherwise increment the previous value. Store it in server cache
3. Intercept the forms at <input type="submit" > and insert a hidden parameter PAGEID, before it.Store it in server cache also.

à Receiving Request

1. Before processing any get or post request
2. getURL()
3. Convert URL to string
4. Search and extract PAGEID
5. doMatch(PAGEID,Cache) //check whether the pageid is in cache . If yes valid request
6. If not valid fire error page

Ø Advice: Sessionvalidator

1. Declare and initialize variables

```
HttpServletRequest request=null;  // request object
HttpServletResponse response=null;  // response object
SessionHDIV sesshd=null;        //stores session object
  static int ipid=-1;         //stores pageid
  static int sid=1;           //stores stateid
  IState state1=null;           //stores state
  String pagenam;          //stores pagename
  cryptos.EncodingUtil eu;    // encoding object
  MessageDigest md;         //stores hash object
  cryptos.CipherHTTP c;       //stores cipher object
  cryptos.KeyFactory k;//     keyfactory object
  cryptos.Key key;          //store key
  boolean valid=true;         //stores validity
  boolean pagevalid=true;      //stores validity
  boolean formreq=false;       //stores request type
  URLCodec uc;        // url encoding object
  int method=0;             // form method type
  int sellist=0;    //select object option value
  int selectno=0;    //stores select object count
  ParamMap pm[]=new ParamMap[3];//stores select
                          object value-relative value
  String selname=null;   //stores select object name
String hf="<input type=\"hidden\" name=\"";//store hidden
field format
```

2. Initialize Session

void initSession(HttpServletRequest req,HttpServletResponse res)

a. Assign passed request and response object to instance variables request and response
b. Create new session

3. Clear Session

void clearSess()

a. Assign null to request and response object
b. Reset processed ipid=-1

4. Process Request

int processReq(HttpServletRequest req,HttpServletResponse res)

a. Assign passed values to request and response objects respectively
b. If the request url is login page/login verify page then
   Return 0
   Else
c. If querystring!=null and querystring contains "pageid" and valid =true then
   i. Decode url
   ii. //verifying parameters forn integrity
   Extract Parameters
   1. create stringtokenizer object, parse querystring using '&' as token separator
   2. retrieve while moretokens
   3. if not token.indexof("pageid") then
      retrieve parameter and value
      retrieve from session saved parameter value
      compare both values
      if same then parameters are correct
      else
         check if parameter is noneditable list
         if yes then
         retrieve saved value from map
      if cannot retrieve then parameters manipulated by user
      endif
      endif
      else
      parameters are manipulated
      restore from session
      return 0
      endif
      endif

d. else
   if request !=null //request from form
   i. get parameters and store in Enumeration paramNames
   ii. while paramNames has more elements
      1. retrieve next element
      2. retrieve its value from Session
      3. if returned value !=null then
         if pageid then
         retrieve its values
         valid=0
      4. compare value and values
         if same valid++
         repeat ii for all parameters

iii. if valid>0 pagevalid=true
else
pagevalid=false
return 0

else
invalid request
fire WarnPage
return 0;

5. Redirecting Page
String sendRedir(String uri)
a. If request url=login page/verify page then
ipid++
b. If session !=null then
Set session attribute "pageid"+ipid to ipid
Prepare parameter pageid
Find hashed value of parameter pageid, ie.hash(ipid)
c. Save in session attribute
"shash"+ipid the hash(ipid)
d. Attach pageid parameter to request url
e. Valid=true
f. Save url in session attribute "url"+ipid
g. Return url
6. Start Session
void initSess()
a. Get session object
b. If ipid=-1 then start session
c. Ipid=generateinitialpageid //randomly generated
d. Set attribute "pageid"+ipid to ipid
7. Check and process tags
String checkout(String st)
a. If st contains "<a href=.." then
i. Ipid++
ii. If session !=null then
iii. Set session attribute "pageid"+ipid to ipid
iv. Extract query string and store in rurl
v. Set session attribute "url"+ipid to rurl
vi. Prepare pageid parameter
1. if ? is present in st then
attach pageid string "&pageid"+ipid+"="+ipid to query string of rurl
2. else attach "?pageid"+ipid+"="+ipid as query string to rurl
vii. Hash parameter values and encode url
1. read query string
2. tokenize parameters
3. replace parameter value with hval(ie.hash(parameter value))in st
4. set session attribute param+"hash" to hval
5. set session attribute "shash"+ipid to hval
6. url=querystring
7. eurl=encoded url
8. replace url with eurl in st

b. if st contains "<select id " then
i. sellist=1
ii. extract id
iii. selname=id
iv. create parammap object

c. if st contains "<option " then
i. extract value
ii. put sellist and value in paramMap object(sellist,val)
iii. replace val in st with sellist
iv. increment sellist

d. if st contains "</select" then
i. sellist=0
ii. set session attribute selname+"map" to parammap object
iii. selectno++//increment list counter

e. if st contains "form method=" then
i. if form method=get then method=1
ii. else method=0
iii. if request!=null then
1. if session !=null then
2. ipid++
3. set session attribute "pageid"+ipid to ipid
4. rurl=extracted action url
5. set attribute "url"+ipid to rurl
6. find hash value of ipid ie hval
7. set session attribute "shash"+ipid to hval
8. prepare parameter "pageid"+ipid=hval

9. concatenate st with prepared pageid parameter

f. Check validity
i. If st contains "<html " then
ii. If request !=null then //retrieve parameters
1. get parameters from request
2. get session attribute values for every parameter
3. compare if requested and retrieved values match
4. if matches then pagevalid=true else pagevalid=false

iii.//retrive parameters from form method=get if request !=null
1. qs=get query string
2. if qs !=null then
3. qs1=decoded qs
4. retrieve parameters from qs1
5. retrieve pageid
6. get session attribute "shash"+ipid
7. compare requested and retrieved values
8. if same then
retrieve session attribute url
9. compare rurl and url
10. if same valid=true
else valid =false
if rurl !=logn page/ verify page then
fire Warnpage
else
valid=false

a. How to apply aspect to existing applications

Aspect can be applied to existing applications. Aspect pointcuts are identified according to the technology used for developing applications. Proof of the concept implementation has been done according to compile time weaving based technology java servlets.

Following are the steps to weave aspect in to existing application at source code level.

i. Install eclipse IDE and server for eg: tomcat
ii. Plug-in for AspectJ should be installed to enable aspect compilation
iii. Install spring frame work and apache commons-code
iv. Open web application project in IDE Eclipse

v. Convert the web application project to aspectJ project
vi. Copy the aspect package into the project folder
vii. Build the project

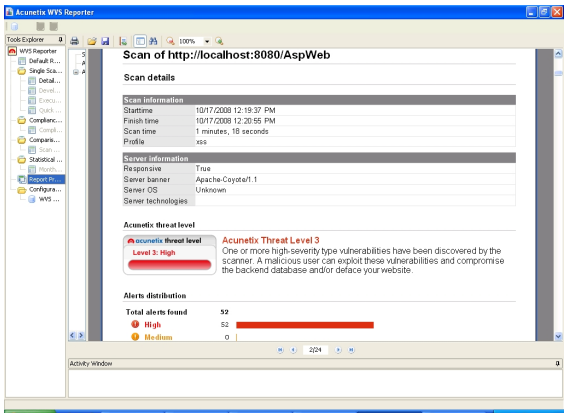
The above steps will weave the default framework to the application. The default framework has the commonly used pointcuts and advice implementation.
Customization can be done manually to identify new pointcuts.

III. RESULTS

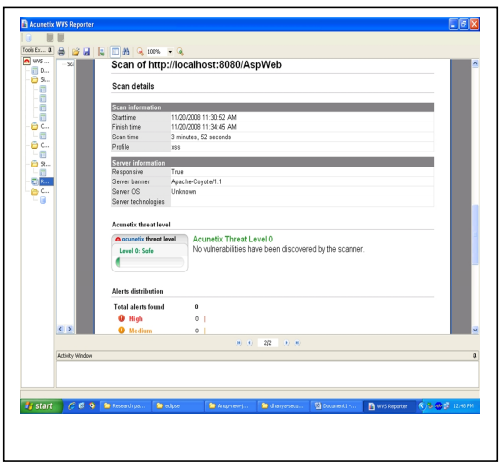
We have used manual and tool based experiments to do the vulnerability checks before and after the injection of advice. Acunetix vulnerability scanner results are shown in the figure below

A. Screen Shots

AspWeb is an application developed in java servlets. It was susceptible to cross site scripting vulnerability as the report in figure (4) shows. After applying the XSSInjection Check aspect it was found self defendable as shown in figure(5)

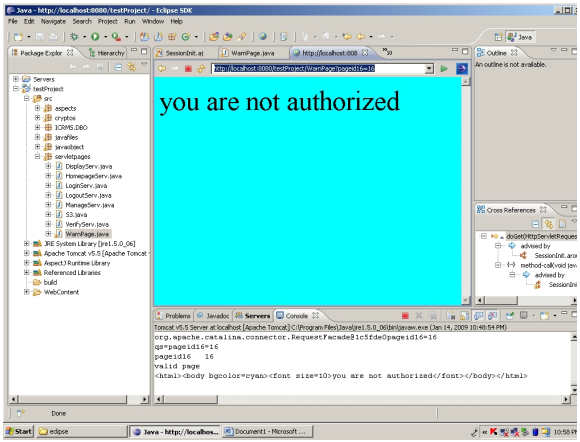


Figure(4) Test Result of AspWeb application (before)



Figure(5) Test Result of AspWeb application (after applying XSSInjectionCheck aspect)

When url of a page that is not passed to the client is requested user is not allowed to view that page and redirected toWarnPage



Figure(6) Test Result of jspSample application (after applying session management aspect)

IV. DISCUSSION

We have discussed some of the work done in the related area and mentioned how our work is differentiated or contributed to these approaches, and how these work are related to our approach.

Lidia [17] introduces an approach that can be used to weave multiple aspects in to the executable UML model. We do weave multiple aspects of the security domain and thus more specific.

An overview of AspectJ by Gregor Kiczales [9], Erick Hilsdale discusses the capabilities of AspectJ. AspectJ is an extension of Java which facilitates aspect Oriented features. AspectJ provides language support for defining aspects and its properties. The join point model gives a clear picture of how crosscutting concern is weaved with non-aspect code. The paper discusses commonly use pointcut designators and joinpoint. The tool support is provided by JBuilder4. This paper only covers important points in the initial draft of AspectJ. But the concept of AOP has been integrated well in to AspectJ. We use AspectJ to implement our proof of concept.

The members of the Web Application Security Consortium [28] have created a project to develop and promote industry standard terminology for describing the issues. Application developers, security professionals, software vendors, and compliance auditors will have the ability to access a consistent language for web security related issues. We have addressed the standard problems that are discussed by the consortium.

Web security vulnerabilities continually impact the risk of a web site. When any web security vulnerability is identified, performing the attack requires using at least one of several application attack techniques. These techniques are commonly referred to as the class of attack (the way security vulnerability is taken advantage of). Many of these types of attack have recognizable names such as Buffer Overflows, SQL Injection, and Cross-site Scripting. As a baseline, the class of attack is the method the Web Security Threat Classification will use to explain and organize the threats to a web site. The Web Security Threat Classification is a cooperative effort to clarify and organize the threats to the

security of a web site. The Goals are

i. Identify all known web application security classes of attack.
ii. Agree on naming for each class of attack.
iii. Develop a structured manner to organize the classes of attack.
iv. Develop documentation that provides generic descriptions of each class of attack.

In addition to Web Application Security Consortium there exists lot many organizations and individuals who work towards security attack prevention. To name a few SPI Dynamics, Symantec, Security Forge, IBM, Microsoft, IEEE, ACM etc.

SPI Dynamics has a series of whitepapers[15] describing various steps in building secure web application by incorporating it into the development phases, various attacks and its aftereffects, web application security assessment etc.

"Comparison of available tools for buffer overflow" by John Wilander and Mariam Kamkar[10] discusses the buffer overflow concern. This paper gives a good illustration of the tools available for buffer overflow prevention. It contains an exhaustive analysis of buffer overflow attack scenarios and categorized them. The paper does not talk about java/.net based applications buffer overflow issues and is focused on c/c++ based applications. Our work presents the buffer overflow concern in java based applications.

Jurjens [13] defines a general approach to security concepts modeling but we focus on modeling web application specific countermeasures of attacks.

The work "A proposal and implementation of Automatic detection/collection system for Cross site scripting vulnerability" by Omar Ismail and Masahi Etoh [20] discusses the cross site scripting vulnerability. Their work is based on the client side XSS detection and manipulation of request and response. Our work uses aspect oriented server based detection and manipulation of user input.

"Abstracting application level web security" by David Scott and Richard Sharp [7]. This paper also suggests some ways in which the web applications can be protected. They have illustrated the difficulties that are inherent in adding security to these applications. They emphasize on security that is beyond the restrictions of technologies, server and database which is used for the development and deployment. They have presented a structuring technique that incorporates security policies. A specialized language called security policy description language was proposed to program an application level firewall(security gateway). The program written in SPDL is compiled and executed on firewall. Firewall dynamically analyzes this http request and response using SPDL to ensure security. The stress in this paper is given to defend form modification, sql attacks and cross site scripting. They gateway does client side form validation and stick to authenticated data passing using Message authentication codes; and thus common attacks are prevented.

An automatic Defence Mechanism for malicious injection attack by Jin-Cherng Lin and Jan-Min Chen [12]presents an application level security gateway to filter malicious input that lead to scripting attacks. This method is targeted at binary form of software while our approach is based on source code level compile time aspect weaving. This approach gives error page in case any malicious code is found. Our approach encode/strip the malicious code and execution continues.

V. .CONCLUSION

"Prevention is better than cure". Same is the case for web application. Since internet has given way for easy accessibility web applications have become not only popular but a necessity. The loopholes in web applications are tremendous that it should be addressed in all phases of web application life cycle. Web application security issues are very critical and exist in spite of the availability of best coding practices. Here we defined how the countermeasures of attacks can be modeled separately and later on weaved to the base model. Conversion of these aspects in to platform specific one is to be done inorder to achieve the effectiveness of the approach. Our proof of concept is implemented for java servlet based application where AspectJ is used as the plugin for defining aspects.. Security incorporation responsibilities drifted from developers to analyst. This security aspect models are platform independent one. This paper contribute to the software development community in reducing the complexity of incorporation of security concerns during the development. It also enables the existing applications to be self defendable by weaving these aspects to itself.

ACKNOWLEDGMENT

We sincerely thank all those who have reviewed our paper.

REFERENCES

[1] Ahsan Habib, Mohamed M. Hefeeda, and Bharat K. Bhargava, "Detecting Service Violations and DoS Attacks", National Science Foundation,pp1-13
[2] A. Schauerhuber, M. Wimmer, E. Kapsammer, W. Schwinger and W. Retschitzegger, Bridging WebML to model-driven engineering: from document type definitions to meta object facility, IET Softw. 2007.
[3] Application security-An essential part of your risk management program-IBM whitepaper 2005,pp1-2
[4] Bobbitt M.Bulletproof web security. Network security magazine. Techtarget storage media may 2002,pp 1-10
[5] "cross site scripting info ". http://httpd.apache.org/info/security-Apache
[6] David Larochelle and David Evans. "Statically Detecting Likely Buffer Overflow Vulnerabilities" In 2001 USENIX Security Symposium, Washington, D. C., August 2001
[7] David Scott, Sharp R Abstracting Application level web security. 11th international conference on WWW.
[8] Filippo Ricca, Massimiliano Di Penta, Marco Torchiano,, Paolo Tonella, Mariano Ceccato , The Role of Experience and Ability in Comprehension Tasks supported by UML Stereotypes , Softwae Engineering, 2007 , ICSE 2007 May 29 th International Conference, Pages 375-384.
[9] Gregor Kiczales, Erick Hilsdale, An overview of AspectJ
[10] John Wilander,Marian Kamkar.A comparison of publicly available tools for dynamic buffer overflows prevention. N/w and distributed System security symposium conference proceedings 2003, pp3-10
[11] Joshi J,W. Ghafoor, A Stafford.E. "Security models for web based applications" Communications of ACM Feb 2001
[12] Jin-Cherng Lin and Jan-Min Chen. "An automatic Defence Mechanism for malicious injection attack."
[13] J.Juerjens. UMLsec: Extending UML for Secure Systems Development. In Proc. Of 5th Int. Conf. on the Unified Modeling Language, Lect. Notes in Comp. Sci. 2460 pages 412-425, Springer, 2002.
[14] J Jurjens UMLSec: Extending UML for secure systems development

[15] Kevin Heineman, SPI Dynamics: "Complete web Application security :Phase I building web Application security into your development process. SPI Dynamics whitepaper 2002
[16] Leslie Lamport. Password authentication with insecure communication. Communications of the ACM,Nov 1981
[17] Lidia Fuentes, Pablo Sanchez , Designing and Weaving Aspect-Oriented Executable UML models, Journal Of Object Technology August 2007.
[18] Nora Koch and Andreas Kraus , The Expressive Power of UML-based Web Engineering, 2nd Int. Workshop on Web-oriented Software Technology(IWWOST02), Malaga, Spain, June 2002.
[19] N.Koch, Classification of model transformation techniques used in UML based web engineering, IET Softw,2007,pp 98-111
[20] Omar Ismail and Masahi Etoh, "A proposal and implementation of Automatic detection/collection system for Cross site scripting vulnerability" .
[21] P.Fraternali, P.C Lanzi Exploiting conceptual modeling for web application quality evaluation. 13th International conference on World wide Web May 2004
[22] Peter F. Linington and Pulitha Liyanagama. Incorporating Security Behavior into Business Models using a Model Driven Approach. 11th IEEE International Enterprise Distributed Object Computing Conference(2007)
[23] Pramod Dhanya. "Modeling security aspects in model driven web application development", International Conference in Advancement in Computing, ACM ,Chikli, India 2008
[24] Secure Software Development by Example. IEEE security & privacy, July 2005
[25] Stephan Kais, Engin Kirda "SecuBat-A web vulnerability scanner". 15th international conference World Wide Web May 2006,pp248-253
[26] Steven M Bellovin, Michael Merrit. "Encrypted key exchange:Password based protocols secure against dictionary attacks". IEEE symposium on security and privacy 1992
[27] Symantec whitepaper, Internet security threat report trends for july 05-06
[28] Web Application security consortium.Threat classification
[29] Web Application security whitepaper-Acunetix Nov 2005
[30] Yao-wen Huang, Shih-kun Huang, Tsung-Po Lin, Chung-Hung Tsai Web application security assessment by fault injection and behaviour monitoring.12th International World Wide Web Conference-ACM Press,pp 149-156
[31] Yao-wen Huang, Fang Yu. Securing web application code by static analysis and runtime protection. 13th International conference on World wide web May 2004-ACM press,pp41-48