INCORPORATING SECURITY INTO WEB APPLICATIONS - AN ASPECT ORIENTED APPROACH

Dhanya Pramod*

Abstract:

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. Web applications currently hosted are subject to risk and addressing this issue is a primary concern for organizations. Aspect oriented method to inject security concern in to web application is proposed to address this threatening risk. This work gives an outline of this self defense mechanism that should be injected in to a web application. The top vulnerabilities found in web applications are addressed and countermeasures are designed as aspects. These aspects are then injected into the hotspots of web applications. The method used to identify the hotspots is also covered in the paper. It then covers the experimental approach for detecting the vulnerabilities and testing the correctness of the self defense approach. The experiments done with various existing applications revealed so many vulnerabilities. The experiments were also done after injecting the aspect package and found that the web applications were able to defend the vulnerabilities under consideration. The performance of the application before and after injecting the aspect is measured in order to justify the feasibility of the solution.

Keywords: Web Application, Aspect oriented approach, vulnerability, self defense, cross-site scripting, Sql Injection

* Symbiosis Centre for Information Technology (Symbiosis International University), Pune – 411057, INDIA

Introduction:

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. 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.

Various approaches exists to handle web application security. Jurjens [13] defines a general approach to security concepts modeling and focus on the importance of incorporating security concerns during software development life cycle. The approach can be extended to design security notations and models for application specific counter measures of attacks. "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.

"Abstracting application level web security" by David Scott and Richard Sharp [7] suggests some ways in which the web applications can be protected. The stress in this paper is given to defend form modification, sql attacks and cross site scripting. The gateway does client side form validation and stick to authenticated data passing using Message authentication codes; and thus common attacks are prevented.

An automatic Defense 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. Yao-Wen Huang, Shih-Kun Huang, Tsung-Po Lin, Chung Hung Tsai in their paper "web application security assessment by fault injection and behavior monitoring" [30] discusses a platform for assessing web application security. As an interface between testing

techniques and Web applications, WAVES can be used to conduct a wide variety of vulnerability tests.

Stephan Kais, Engin Kirda in their paper "SecuBat-A web vulnerability scanner" [25] discusses that many web application security vulnerabilities result from generic input validation problems. They proposed a generic and modular web vulnerability scanner that analyzes web sites for exploitable SQL and XSS vulnerabilities. Using Aspect Oriented Techniques to support separation of concerns in Model Driven Development by Arnor Solberg [2] is an appreciable work that proposes a Model Driven Development framework that adopts Aspect Oriented techniques and discusses the PIM to PSM transformations.

Web Application Risks:

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. For existing web applications the security aspects can be injected to make them self protected. In the approach counter measures are designed as aspects that have two components, pointcuts and advice. Advice is the method that tackles the problem. Pointcuts matches or selects the part of the program which needs interception to inject the advice. We have identified the common pointcuts in web applications and coined advices that are to be injected at every pointcut. A pointcut selects a set of points in the execution of a program called as joinpoints.

The security concerns we are trying to incorporate are the counter measures of the following

- Authentication issues
- Authorization issues
- Cross site scripting
- Sql Injection
- Parameter tampering
- Buffer Overflow

- Session Hijacking

The countermeasure aspects are modeled separately and later on weaved with the base web application model. This enables to apply the approach to both existing and developing applications.

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

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

Aspect Injection:

The following were the attacks the researcher had taken into account for application self defense. For every attack countermeasure aspects are defined. Proof of concept implementation done using AspectJ for Java based applications. The excerpt of the advice is given in table(1). The identified pointcuts and advices of aspects are also given.

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.

Table (1) Aspect details

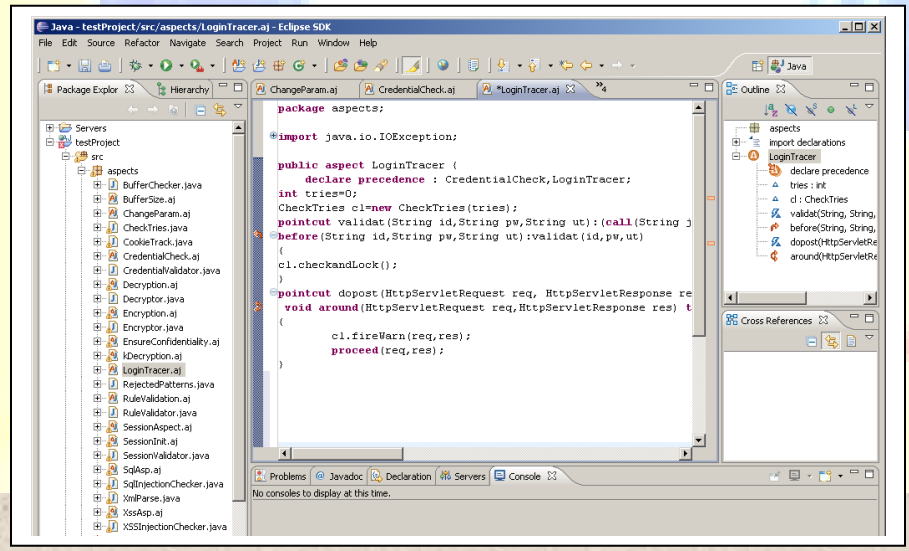

Figure (1) loginTracer Aspect

Aspect pointcut- Login validation method of application validate()

Results:

The researcher had taken various applications developed using java servlet technology for experiments. The sample applications selected were vulnerable to attacks. The popular attacks were used for testing. For every application the following categories of test cases were executed.

1. Sql injection test

Manual testing strategy was adopted.

All the input points of applications were injected with malicious patterns containing sql query modification or bypass sub queries like

Select * from user where username='dhanya' and password='dd' or '1'='1'

Username=';drop table users--

Username=' having 1=1--

Username=' or 1=1

Username=' union

Username='; drop|select

Use of select, insert, delete, where, from, show tables etc

The applications were found vulnerable.

After injecting the aspect again the tests were repeated and found that application protects itself from these attacks by cleansing the input.

2. Cross site scripting

Automation/Manual Testing strategy were adopted.

All the possible test cases as per cross site scripting cheat sheet were tested.

The applications were found vulnerable.

After injecting the aspect again the tests were repeated and found that application protects itself from these attacks by cleansing the input.

3. Authentication

Manual testing was adopted.

The candidate applications were checked for authentication module and brute force attacks. Some of the applications were found vulnerable.

After injecting the aspect login attempts were found tracked and brute force attacks were prevented.

4. Authorization

Manual testing was adopted.

The strength of credentials (length, mix of characters, match with username) has been verified and most of the applications were having weak credential checks.

5. Parameter tampering

Manual testing was adopted.

The parameters passed were checked for its value for integrity when passed between client and server. The http headers have been analyzed for ensuring this.

Some applications were found vulnerable.

The aspect was injected and found that the integrity has been ensured.

6. Session/URL manipulation

Manual testing was adopted.

Each and every request response has been verified for session hijacking by changing the url to another. Some applications were found allowing url manipulation. After injecting the aspect it was found that user was not allowed to view the restricted pages.

Each and every request response has been verified for integrity of session information (Cookies, Session variables). Some applications were found allowing manipulation. After injecting the aspect it was found that user was not allowed to change the restricted information.

7. Buffer overflow

Manual testing was adopted.

Every possible input has been verified by giving a large data.

The following table gives the results of some highly vulnerable applications.

Sr.no	Attack	Before Aspect injection	After aspect injection
1	Sql Injection	Highly vulnerable. No filtering was done to restrict sql statements through user input. (Manual testing)	Aspect filtered user input and hence fully protected (Manual testing)
2	Cross Site Scripting	Highly vulnerable 108 vulnerabilities in a single user interaction page. (Acunetix automated scan)	Aspect filtered all malicious scripts and hence full protected. (Acunetix automated scan)
3	Authentication	Was properly checked. But no tracing done for failed logins	Aspect weaved login tracer and brute force attacks were prohibited.
4	Authorization	Weak password	Aspect weaved credential checks to verify password strength
5	Parameter Tampering	No verification was done	Aspect ensured confidentiality of parameters passed.
6	Session/Url manipulation	No care taken for url modification	Aspect weaved session tracking and url verification.
7	Buffer Overflow	No input length verification	Aspect verified length of input and huge data were truncated.

Table (2) Placement system-test results

Sr.no	Attack	Before Aspect injection	After aspect injection
1	Sql Injection	Highly vulnerable. No filtering was done to restrict sql statements through user input. (Manual testing)	Aspect filtered user input and hence fully protected (Manual testing)
2	Cross Site Scripting	Highly vulnerable 53 vulnerabilities in a single user interaction page. (Acunetix automated scan)	Aspect filtered all malicious scripts and hence full protected. (Acunetix automated scan)
3	Authentication	Was properly checked. But no tracing done for failed logins	Aspect weaved login tracer and brute force attacks were prohibited.
4	Authorization	Weak password	Aspect weaved credential checks to verify password strength
5	Parameter Tampering	No verification was done	Aspect ensured confidentiality of parameters passed.
6	Session/Url manipulation	No care taken for url modification	Aspect weaved session tracking and url verification.
7	Buffer Overflow	No input length verification	Aspect verified length of input and huge data were truncated.

Table (3) testProject-test results

Discussion:

The main objective of the research to achieve application self defense was met. The counter measures defined in the form of aspects were found effective in protecting the web applications. The aspect oriented approach was found good in inculcating the security concerns into existing system easily. The model proposed could be used by software development team to integrate security. This approach was found to be the best in terms of the development and implementation effort of security aspects is considered. Other methods referred in the literature review either act as a separate protection tool or need inline modification of the web application code in order to integrate security in existing applications. Also there was no work in literature that takes care of these many security concerns as the researcher has done.

Performance Analysis

This research proposes a security aspect package to be injected in the application. The researcher has done a proof of the concept implementation for java servlet based web applications. This default implementation may need customization according to programmers coding style and application needs. Henceforth a performance metrics (given below) was suggested to decide the changes that need to be incorporated in the current automation of aspect counter measures. Injecting the security module into an existing application definitely increase the cpu and memory utilization. Hence the researcher has done a performance analysis (details given in next section) to find how much overhead is incurred to inject security aspect . The experiments revealed very less/negligible change in resource utilization and the method is thus effective.

Performance Metrics

Proof of the concept has been implemented in java and AspectJ.

The following table gives the details of the level of automation we have done to capture the attack pointcuts and thereafter to inject advice.

Sr.no	Attack	Automation	Customization Needed
1	Sql Injection	100 %	If new threats have

		(pointcuts,advice,malicious pattern)	cropped up. If some malicious pattern to be allowed
2	Cross Site Scripting	100 % (pointcuts,advice,malicious pattern)	"
3	Authentication	90% Pointcuts-default identified Advice- default identified	New pointcuts may crop up depending on application And credentials.
4	Authorization	80% Pointcuts-default identified Advice- default identified	New pointcuts may crop up depending on application
5	Parameter Tampering	100 % (pointcuts,advice)	
6	Session/Url manipulation	70% Pointcuts-default identified Advice-default identified	New pointcuts should be identified according to the application Advice may require modification
7	Buffer Overflow	50% Pointcuts –default identified Advice defined	Need all input data types of application to be specified to generate advice.

Table (4) Automation details

For preparing an application to be self defendable would take considerably less time for conventional applications that are built using java servlets. But for applications that use third party plugins or api customization need time which depends on the no.of new pointcuts and advice.

Automation Requirements

1. Credential Validity

Need to know the method that read new password entered by user

2. Buffer Overflow

Need to know the size of every parameter and its name

3. Login Tracer

Need to know the method that verifies credentials and servlet method that has the login form

4. Encryption/Decryption

Need to know the method used to verify login to initialize cryptos.

Need to know which parameter is to be encrypted/decrypted and methods used to

save and retrieve the data

5. RuleValidation

Need to know which all parameter has a specific business rule

6. Cookie track(Parameter manipulation)

Default pointcut works well

7. Sql Injection

Default pointcut works well

8. Xss Injection

Default pointcut works well

9. SessionAspect.

Need to know login verify page

Need to know pages that are allowed to view without login(pages before authentication)

Need to know logout page

Need to know Warning/error pages that are fired during unauthorized access

Performance overhead Analysis

The performance monitoring facility of Windows operating system is used to evaluate the application performance. Task manager reports the percentage utilization of CPU for every application running on it. The application was monitored without aspect package and performance has been noted. After injecting the aspect application performance has been noted. It was found that there is a cpu overhead which varies from 0**.7% to 1.7** depending on the size and user interaction points. Results of some experiments are given below

Project placement system
CPU utilization average(without aspect) :8.6%
CPU utilization average(with aspect) :10.3%
Overhead=1.7%

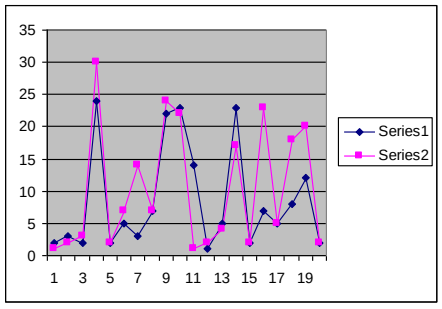

Figure (2) CPU utilization chart-placement system

Project jspSample
CPU utilization average(without aspect) :8.38%
CPU utilization average(with aspect) :9.79%
Overhead=1.41%

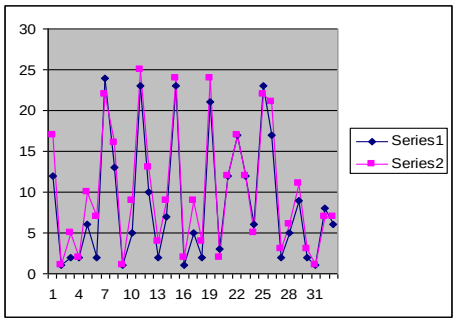

Figure (3) CPU utilization chart-jspSample

Project:testProject
CPU utilization average(without aspect) :3.11%
CPU utilization average(with aspect) :4.23%
Overhead=1.18%

Project:testpr
CPU utilization average(without aspect) :2.8%
CPU utilization average(with aspect) :3.66%
Overhead=0.86%

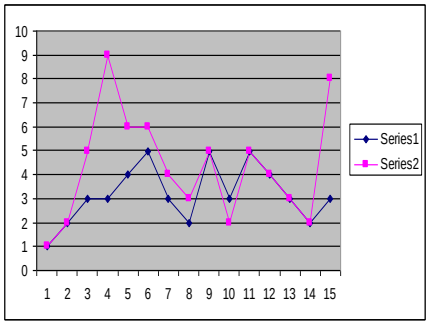

Figure (4) CPU utilization chart-testProject

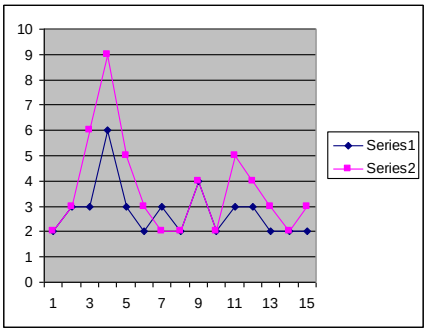

Figure (5) CPU utilization chart-testpr

The application was also monitored without aspect package and memory utilization has been noted. After injecting the aspect application memory utilization has been noted. It was found that there is little memory overhead Results of some experiments are given below

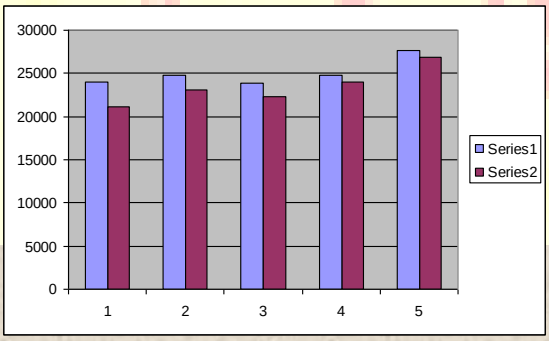

Figure (6) Memory utilization chart-testpr placement system

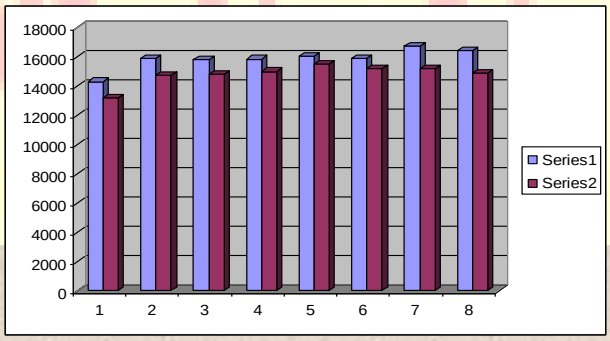

Figure (7) Memory utilization chart-

Conclusion:

In this era of collaborative computing and knowledge management, proliferation of threats is an issue of major concern. This work depicts the efforts to integrate security in to web applications without manually changing of code. The top vulnerabilities reported at code level such as cross site scripting, sql injection, parameter tampering, authentication, authorization, url manipulation, session hijacking and buffer overflow have been addressed in the work.

Aspect oriented approach to integrate the security module facilitates

i. Easy development of web applications(programmer need not worry about security) and reduce complexity in design.
ii. Integration of only security aspects that are essential
iii. Security to those who can afford little performance issues.
iv. Separation of security concerns
v. Integration of security aspect to existing as well as developing applications

Experiments with the developed software were done manually and using well known web vulnerability scanner Acunetix to prove it to be self defendable. The researcher conducted the vulnerability testing of sample applications which are susceptible to vulnerabilities. A detailed analysis of the results was done. Then the aspect package was weaved at the source code level (compile time) and conducted the tests again. The results were observed and verified that the previous attacks were prevented and the application has become self defendable to attacks.

Limitations of this approach is that since new web application vulnerabilities and attacks can crop up any point of time in future addressing them will be a hurdle. The current research scope is authentication, authorization, cross site scripting, sql injection and parameter tampering. To give the proof for the concept and framework generation the language and tools need support for aspect oriented methodology.

References:

- Ahsan Habib, Mohamed M. Hefeeda, and Bharat K. Bhargava, "Detecting Service Violations and DoS Attacks", National Science Foundation,pp1-13
- Arnor Solberg , "Using Aspect Oriented Techniques to support separation of concerns in Model Driven Development", IEEE 29th Annual international computer software and applications conference(COMPSAC'05)
- Application security-An essential part of your risk management program-IBM whitepaper 2005,pp1-2
- Bobbitt M.Bulletproof web security. Network security magazine. Techtarget storage media may 2002,pp 1-10
- David Basin, Jurgen Doser, "Model driven security for process oriented systems. SACMAT03 ACM conference Cotno, Italy
- David Larochelle and David Evans. "Statically Detecting Likely Buffer Overflow Vulnerabilities" In 2001 USENIX Security Symposium, Washington, D. C., August 2001
- David Scott, Sharp R Abstracting Application level web security. 11th international conference on WWW.
- 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.
- Gregor Kiczales, Erick Hilsdale, An overview of AspectJ
- 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
- Joshi J,W. Ghafoor, A Stafford.E. "Security models for web based applications" Communications of ACM Feb 2001

- Jin-Cherng Lin and Jan-Min Chen. "An automatic Defence Mechanism for malicious injection attack."
- 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.
- J Jurjens UMLSec: Extending UML for secure systems development
- Kevin Heineman, SPI Dynamics: "Complete web Application security :Phase I building web Application security into your development process. SPI Dynamics whitepaper 2002
- Leslie Lamport. Password authentication with insecure communication. Communications of the ACM,Nov 1981
- Lidia Fuentes, Pablo Sanchez , Designing and Weaving Aspect-Oriented Executable UML models, Journal Of Object Technology August 2007.
- 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.
- N.Koch, Classification of model transformation techniques used in UML based web engineering, IET Softw,2007,pp 98-111
- Omar Ismail and Masahi Etoh, "A proposal and implementation of Automatic detection/collection system for Cross site scripting vulnerability" .
- P.Fraternali, P.C Lanzi Exploiting conceptual modeling for web application quality evaluation. 13th International conference on World wide Web May 2004
- 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)
- P. Fraternali, N. Moreno and A. Vallecillo. WebML modielling in UML, IET Software, 2007 pp 67-80
- Secure Software Development by Example. IEEE security & privacy, July 2005

- Stephan Kais, Engin Kirda "SecuBat-A web vulnerability scanner". 15th international conference World Wide Web May 2006,pp248-253
- Steven M Bellovin, Michael Merrit. "Encrypted key exchange:Password based protocols secure against dictionary attacks". IEEE symposium on security and privacy 1992
- Symantec whitepaper, Internet security threat report trends for july 05-06
- Web Application security consortium.Threat classification
- William G.J. Halfond, Alessandro Orso, WASP: Protecting Web applications Using Positive Tainting And Syntax-Aware Evaluation. IEEE Transactions On Software Engineering, Vol. 34, No. 1, January/February 2008, pp65-81
- Xiang Fu, Xin Lu, Boris Peltsverger, Shijun Chen , "A Static Analysis Framework For Detecting SQL Injection Vulnerabilities", 31st Annual International Computer Software and Applications Conference(COMPSAC 2007)
- 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
- 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