Defence Science Journal, Vol. 62, No. 5, September 2012, pp. 324-330, DOI: 10.14429/dsj.62.1291

Reducing Attack Surface of a Web Application by Open Web Application Security Project Compliance

Sumit Goswami[*#], Nabanita R Krishnan[#], Mukesh[#], Saurabh Swarnkar[!], and Pallavi Mahajan[$]

[#]*Directorate of Management Information System & Technologies, DRDO, New Delhi*
[!]*IAP Company Pvt Ltd, Gurgaon, India*
[$]*Beant College of Engineering and Technology, Punjab*
[*]*E-mail: sumit_13@yahoo.com*

ABSTRACT

The attack surface of a system is the amount of application area that is exposed to the adversaries. The overall vulnerability can be reduced by reducing the attack surface of a web application. In this paper, we have considered the web components of two versions of an in-house developed project management web application and the attack surface has been calculated prior and post open web application security project (OWASP) compliance based on a security audit to determine and then compare the security of this Project Management Application. OWASP is an open community to provide free tools and guidelines for application security. It was observed that the attack surface of the software reduced by 45 per cent once it was made OWASP compliant. The vulnerable surface exposed by the code even after OWASP compliance was due to the mandatory access points left in the software to ensure accessibility over a network.

Keywords: Attack surface, DRDO Intranet, project management, open web application security project, security audit, security compliance

1. INTRODUCTION

Web applications use common concepts: browsers as client, HTTP as a communication protocol between client and server, server-side and client-side runtime environments for executing code and XML for data representation. For maintaining quality assurance, the organizations use attack surface metrics to foretell vulnerabilities in the application prior to deployment[1]. The users have interest in considering the security of an application when they have to choose between the alternative applications. In this paper we address the work done by us to reduce the attack surface of web application by OWASP compliance[2]. We used 'Heumann, Keller and Turpe' approach for scoring the attack surface[3].

The user interface (UI) for client which is through the web browser is one reason that is responsible for a web application's attack surface[4]. But the same has to be retained to ensure usability. Therefore, we considered all the UI components for attack surface coverage. Thereafter, the attack surface results of two in-house versions were calculated pre- and post-OWASP compliance.

1.1 Literature Survey

The increasing use of software system has made it important to analyze the system carefully for security and robustness flaws[5]. The use of web applications by the organizations to run their application is increasing day by day. Due to this reason the attack on such systems have become the main target of attackers and hence results as the largest source of security vulnerabilities[6]. Identifying theft, phishing, malware and other computer crimes are the factors which often cost consumers and organisations and put a doubt in front of people to trust online applications[7]. Even though the proposal for large security metrics has come up but complete security of the systems is not guaranteed[8].

The attack surface for windows operating system was introduced by Michael Howard which was liberal and informal[9]. Howard, Pincus and Wing even measured the attack surfaces of seven versions of windows[10]. The attack surface of four versions of Linux was calculated by Manadhata[11], *et al.* The various applications which were smaller is size and some large enterprise systems that were implemented and coded in C and Java were measured and calculated by the attack surface method introduced by Manadhata and Wing[9]. Ha Thanh Le and Peter Kok Keong Loh suggested an application vulnerability description language (AVDL) which is a theoretical approach and is meant to realize and understand a unified data model. The AVDL is mainly based on technology which is independent, vulnerable and is used for analysis of web applications[12]. Static analysis vulnerability indicator (SAVI) is a tool based on the method given by James Walden and Maureen Doyle, which links up several static-analysis metrics and is used in ranking web applications vulnerability[8].

Various research analyses and methods had been proposed theoretically as well as on the basis of tools for attack surface calculation. Thomas Heumann, Jorg Keller and Sven Turpe introduced the notion of 'Quantifying the attack surface of a web application'[3]. The method advised and proposed by them in the paper consisted a multidimensional metric for the attack

Received 11 November 2011, revised 17 August 2012, online published 14 September 2012

surface of web applications. They also discussed the rationale and principle behind the attack surface of web application. A scalar numeric indicator for easy comparison and a descriptive and detailed vector representation for deeper analysis were included in the paper[3]. Attack surface metric is the estimation of the amount of functionality and code in a web application exposed to outside attackers[3].

In our paper we used the Thomas Heumann multidimensional metric for evaluating the attack surface calculation. The metric considered all the UI components and calculated the attack surface vector. We need deeper knowledge of an application[13] for parameter consideration; so we developed two versions of web application having same functionality and considered their parameters for metrics calculation and analyzed the results.

1.2 Project Management Application

Management of various projects in an organization is done with the help of a project management application. Data repository of projects (PDR) is an in-house developed web application running on DRDO Intranet. It is having 25 online forms for keeping the details about the projects. User is authenticated with LDAP running on the Mail Server.

Version 1 of PDR was developed using JSP and ORACLE 10g. It was running on Intranet using TOMCAT web server 4.1 on RedHat Linux 4.5. After deployment of application on the intranet, third party security audit was done by authorized Indian computer emergency response team (CERT-In) security auditor. The summary of the audit report was that the application was not safe for use due to the following reasons:

- Inputs and outputs are not properly designed and lack of good GUI.
- Data integrity does not exist and erroneous data is allowed
- Common and dangerous web vulnerabilities such as cross site scripting and SQL injection exist in the application.
- Pro-active, time bound controls using available best practices should be implemented.

The same process of third party security audit was repeated with version 2 of PDR. Version 2 was developed using JSP, Servlet and Oracle 10g. It was also running on Intranet using TOMCAT web server 7 on Red Hat Linux 4.5. The version 2 followed MVC architecture and was developed according to the guideline of OWASP top 10 security principal 2007. In version 2 we had rectified the vulnerabilities issue. Version 2 was audited and certified by the CERT-In security auditor.

2. ATTACK SURFACE AND SYSTEM RESOURCES

The term attack surface refers to the amount of code, functionality and interfaces of a system exposed to attackers. Attack surface can be calculated in terms of system resources. The system resources are generally the data items, channels and operating environment[5]. The following is the usage of attack surface metrics:

- The programmers use the attack surface metric to improve the quality of code.
- Testers use this metric to estimate the extent to which testing has to be done.
- Users use this metric to compare different applications
- Organizations use this metric to make proper investment on the better application[14].

2.1 Working Model of a Web Application

Web application is accessed on the internet or an intranet. Web application development is the execution of software on a platform independent browser[15] and in developing a web application certain risks are involved like security and software bugs[16]. The main purpose of the attack surface metric is to estimate the area of web application that is exposed to the adversaries. Generally the users and attackers can access the web application through the HTTP interface(s) of the web server(s)[17]. The HTTP interface can be defined as one of the attack surface of the web application. However, there are other risks also, like cookies etc. but the symptom does not always lead to the vulnerabilities in an application[18].

PDR follows the model view controller (MVC) architecture. The goal of this architecture is to separate the application data and the business data from the presentation data to the user. When web application runs on the browser then http request is sent to the particular server where web application is hosted. TCP/IP is responsible for locating and connecting to the server where the application is hosted. The accessibility model of the project management web application is represented in Fig. 1. The web server software, which in case of PDR application is Apache Tomcat 7.01 continuously, runs on the server machine. A working MVC model of PDR version 2 in the execution environment is shown in Fig. 2.

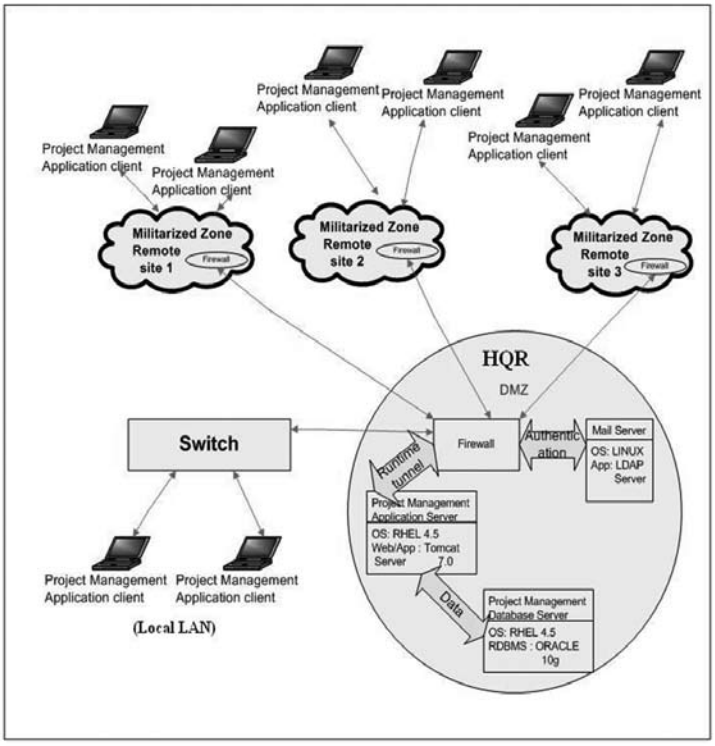

Figure 1. Accessibility model of the project management web application.

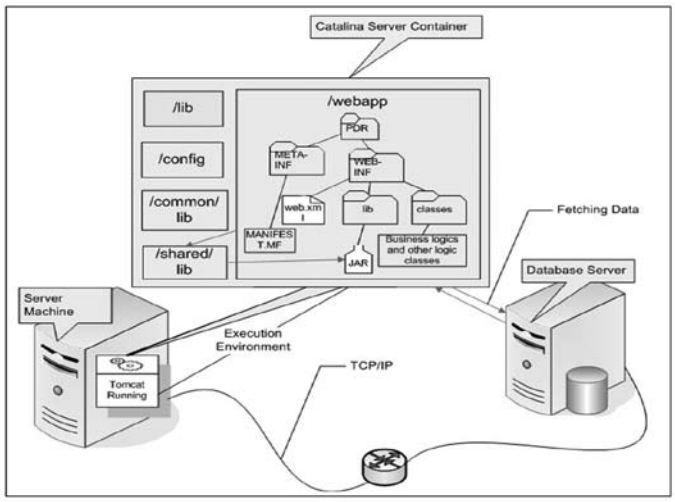

Figure 2. Working model of web application.

2.2 Securing Web Application

When an application is reviewed, analysed and audited, a variety of problems can be unearthed that effect the application. On arranging these problems in some order, it will be easy to tackle them. This order of problems can be reffered as vulnerability catagories[13]. Some of the vulnerability catagories are session management, exception management, input validation, confidentiality and integrity[12].

Apart from these vulnerabilty catagories, there are authentication, authorisation, etc. The security is maintained in the web based application by taking into account the following techniques:

- Reduce the attack surface of the application or remove the unused protocols, functionalities, etc.
- Use least privilege: The security of a system can be increased by running processes having least privileges or access right and hence the capability of the attacker to attack the application can be reduced.
- The developer should not trust the input of the user as the user is considered primary weapon of the attack.[13]
- The applications follow MVC model and other like frameworks which abstract the idea of 'pages' on the server, in the controller, from what is physically presented to the end-user's browser in the 'view'[4]. So Framework is preferable appraoch to develop a web application.
- Various researchers proposed frameworks to avoid vulnerabilities. Teng Lvl and Ping Yan' proposed a framework based on Access Control Policy Description Language and Security Policy Description Language to provide web security[19]. The XML based solution is also one approach to provide security.

2.3 Attack Surface Metric

Attack surface is measured in terms of the resources of an application that are exposed to the adversaries. More the resources are exposed to the user or attacker, more will be the attack surface of an application and hence more will be the insecurity. All the resources are not considered to have equal effect on the security of an application. The entry points and exit points of an application are also considered to be the part of attack surface. The entry point is the point through which data can be entered in the system and the exit point is the point through which data can be retrieved from the application[20]. The application's channels are also considered the basis for the attacks since the attacker can connect to the application through application's channels. There are also another basis for attacks on an application i.e. the attackers can use the persistent data for attack an application. This persistent data can be referred as untrusted data items[21]. The attack surface of the application can be reduced by reducing the amount of running code, reducing application access by users/attackers at entry points, and privilege to limit damage potential[22].

3. HANDLING OF VULNERABILITY AS PER OWASP 2007 GUIDELINES

Open web application security project (OWASP)[2] is to help developers, designers and organizations get an insight to most common web application security vulnerabilities. To improve the software based on the audit report, the entire web application for PDR was redesigned as per OWASP-2007 guidelines[2]. The solution implemented by us to prevent these vulnerabilities is as mentioned:

(a) Cross Site Scripting: We had implemented a code for prevention of XSS vulnerability[23] and functions in java script to sanitize the unwanted input and output to execute. Standard input validation mechanism has been used to validate all input data for length, type, syntax by designing functions at common JavaScript file with regular expression. Invalid input like external script, blacklisted keywords etc. are rejected by using tag lib directors (TLD). One of the ways to prevent XSS is disabling Javascript but this approach is not recommended.

(b) SQL Injection Flaws: Connection and query execution has been established using class. So direct reference to execute database query on server has been disabled. User supplied field is strongly-typed or checked for type constraints and also user input is filtered for escape characters. Query will be executed on the basis of credential assigned to user after input validation.

(c) Insecure Direct Object Reference: The application has been developed using object oriented MVC model approach with Tag Lib Directors (TLD) or frameworks, so it eliminates Insecure Direct Object References. No webpage can be accessed directly from the browser and it checks for session validation.

(d) Cross Site Request Forgery: One of the popular defense against CSRF attacks is the use of a secret token with each request[2, 24, 25]. In version 2 of PDR we had implemented a secret token or random unique key, which is generated with each request. Another way to prevent CSRF is the use of CSRF Guard[25].

(e) Information Leakage and Improper Error Handling: Proper error/exceptions handling has been implemented. Error messages are customized so that error should not display content of server or web application.

(f) Broken Authentication and Session Management: We had implemented a secure mechanism for session management and avoiding cookies. Session expires after idle time of 30

minutes or once the logout option is clicked in the application[26].

(g) Failure to Restrict URL Access: No webpage can be accessed directly from the browser as it checks for session integrity and user credential before showing the web contents.

4. QUANTIFICATION OF THE ATTACK SURFACE

Attack surface vector AS represents the attack surface. According to the Eucledian norm, the attack surface indicator ASI is given by $ASI = |AS|$. Boolean values are the raw measurements that show the presence or absence of a feature given by value 1 or 0, enumerations show multiple-choice measurements, or non-negative integer values as the result of counting.

The infinite count is mapped to the finite range value 0:10. The raw value 0 is mapped with 0, raw value 1-2 is mapped with 1, 3-5 is mapped with 2, 6-9 mapped with 3, 10-14 mapped with 4, 15-20 mapped with 5, 21-27 mapped with 6, 28-35 mapped with 7, 36-44 mapped with 8, 45-54 is mapped with 9, 55-∞ is mapped with 10. The attack surface *AS* of a web application is defined as

AS=ddist;dyn;(security);(input);(active);cookie;role;rights

i.e. the components of attack surface vector are: degree of distribution *ddist*, page creation method *dyn*, security mechanisms *(security)*, input vectors *(input)*, active content *(active)*, cookies *cookie*, user roles and access rights. Round brackets indicate groups of components.

Maximum value of attack surface vector is given by:

AS_{max}*=34;1;(1;10;10;10;10;10;10;10);(1;1;1;4;1;8);(5;7;8;6;10);40;10;10*

The various components of the attack surface metric have been shown as parameter family in Table 1. These are the components that affect the security of a web application and hence are used to calculate the attack surface metric of the web application. These components have been taken from OWASP. The range of these components has been defined in the Table 1. The various components are described as:

Degree of distribution *(ddist)* determines the spanning of the application over multiple domains. More the value of *ddist*, more are the chances of attacks.

- Sub domains ($sdom_{wa}$), domains (dom_{wa}) and foreign domains (dom_{ext}) are required to calculate the value of ddist, which is given by: $ddist = 1/2.sdom + dom + 2.dom_{ext}-1$.
- For max value we consider *sdom* = 10 (we consider raw value 55-∞ for *sdom* is 10)

- Page creation method (*dyn*) distinguishes whether the pages are dynamically created on server side or not. The value of *dyn* is 1 if it uses server-side technologies, otherwise 0.
- Security mechanism (security), if present reduces the value of attack surface. Transport layer security and input validation are considered here. Security mechanism comprises:
  - crypt ∈ {0,1} which shows the presence of TLS;
  - cryptomix ∈ {0,10} which shows the mixing of the contents accessed over the TLS with the contents accessed over HTTP;
  - validate ∈ {0,10}, the value 10 indicates that input validation is not present or has been broken[2];
  - buffer errors ∈ {0,10}, the value is 10, if the developer wants to put the data in the buffer above its threshold amount, otherwise 0;
  - cross-site request forgery (CSRF) ∈ {0,10}, the value is 10 if there is a CSS attack, otherwise 0;
  - cross-site scripting (XSS) ∈ {0,10}, the value is 10 when XSS is present, otherwise 0;
  - sql injection ∈ {0,10},the value is 10, when a security vulnerability is exploit in database layer of an application;
  - direct object reference ∈ {0,10},the value is 10 if the internal implementation object is exposed to the user, otherwise 0.[8]
- Input vectors (input) increase the complexity of an application.
  - The presence of URL parameters (urlparam), HTML forms (forms), hidden form fields (hidden) and HTTP authentication mechanisms auth is indicated by 1.
  - File uploads is indicated by files ∈ {0,8} and search ∈ {0,2,4} indicates the presence of search function. If no site search is present then the value is 0, if locally implemented mechanism is present then the value is 2 and if internet search engine is used then the value is 8.
- Active content (active) has the following set:
  - js ∈ {0,5} for JavaScript,
  - js_{ext} ∈ {0,7} if javascript is loaded from a different site,
  - sss ∈ {0,8} if server-side scripting is used,
  - Ajax ∈ {0,6} if AJAX is used,
  - java ∈ {0,8} if java applets are used,
  - RIA own ∈ {0,10} if flash is used.
- Cookies (cookie) are a compound parameter. These include:
  - c ∈ {0,10} represents presence of cookies and
  - c_{ext} ∈ {0,10} represents the number of foreign cookies. From these we can calculate the value of cookie which is given by: cookie= $c+3.c_{ext.}$
- Access control is also a compound parameter: Role and rights.
  - Role ∈ {0,5,10} and represents the user status: unauthenticated (0), authenticated (5) or root (10);
  - Right ∈ {0,5,10}: none (0), limited (5) or root (10)[3].

Keeping in view these parameters, all the values of our web applications i.e. PDR version 1 and version 2 has been quantified which has been shown into Table 1.

The values assigned in Table 1 shows the maximum risk that can occur in the two project management applications. Here the maximum possibility of risk or insecurity has been taken that can occur in the web applications without considering the precautions or safety measures taken in the web application.

The minimum and maximum attack surface vector are given by :

Attack surface vector ASI

Min 0;0;(0;0;0);(0;0;0;0;0;0;0);(0;0;0;0;0;0);0;0;0 0

Table 1. Attack Surface parameters of project management application

Parameter family	Short Name	Parameters	Range	Version 1 value	Version 2 Value
Degree of distribution	Ddist	Subdomains ($sdom_{wa}$)	0,10	0	0
		Domains (dom_{wa})	0,10	1	1
		Foreign domains (dom_{ext})	0,10	0	0
Dynamic creation	Dyn	Dynamic creation	{0,1}	1	1
Security features	Security	TLS (*crypt*)	{0,1}	1	0
		Partial TLS (*crypto-mix*)	{0,10}	1	0
		Validate (*validate*)	{0,10}	1	0
		Buffer Error	{0,10}	10	0
		Cross site request forgery (CSRF)	{0,10}	10	0
		Cross site scripting	{0,10}	10	0
		SQL injection	{0,10}	10	0
		Direct object reference	{0,10}	10	0
Input vectors	Input	URL parameters (*urlparam*)	{0,1}	1	1
		Forms (*forms*)	{0,1}	1	1
		Hidden fields (*hidden*)	{0,1}	1	1
		Authentication methods (*auth*)	{0,1}	1	1
		Search *((search)*	{0,2,4}	0	0
		File upload (*files*)	{0,8}	8	8
Active content	Active	Client-side scripting own (*js*)	{0,5}	5	5
		Client-side scripting foreign (js_{ext})	{0,7}	0	0
		Server side scripting (*sss*)	{0,8}	8	8
		Ajax (*ajax*)	{0,6}	0	0
		Java (*java*)	{0,8}	0	0
		RIA own (*flash*)	{0,10}	0	0
Cookies	Cookies	Own cookies (c_{wa})	0,10	0	0
		Foreign cookies (c_{ext})	0,10	0	0
Access control	Role rights	Role	{0,5,10}	5	5
		Privileges	{0,5,10}	0	5

Max 34;1;(1;10;10;10;10;10;10;10;10);(1;1;1;4;1;8);(5;7;8;6;10);40;10;10 63.37

The parameters of the web application Project Management Application *Version 1* is:

0;1;(1;1;1;10;10;10;10;10;10);(1;1;1;1;0;8);(5;0;8;0;0;0);0;5;5

And its ASI (attack surface indicator) is given by:

$\sqrt{1^2+0^2+1^2+10^2+10^2+10^2+10^2+10^2}+(1^2+1^2+1^2+1^2+0^2+8^2)+(5^2+0^2+8^2+0^2+0^2+0^2)+0^2+5^2+5^2$

= 26.192

So the insecurity in the web application is 41.33 per cent.

The parameters of the web application Project Management Application *Version 2* is:

0;1;(0;0;0;0;0;0;0;0;0;0;0);(1;1;1;1;0;8);(5;0;8;0;0;0);0;5;5

And its ASI (attack surface indicator) is given by:

$\sqrt{0^2+1^2+(0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2)+(1^2+1^2+1^2+1^2+0^2+8^2)+(5^2+0^2+8^2+0^2+0^2+0^2)+0^2+5^2+5^2}$

= 14.42

So the insecurity in the web application is 22.76 per cent.

5. RESULTS AND ANALYSIS

From the attack surface metric calculated in this paper, the security of different web applications, whether built on same technology or different technology can be compared. The value of the attack surface metric of project management application has reduced down from 41.33 per cent calculated for version 1 to a value of 22.76 per cent in version 2 due to incorporation of the security features in version 2.

It can be observed that the value of attack surface metric of a web application can be reduced but cannot become 0 per cent. As a web based application cannot run in isolation without inputs and outputs from or to users, agents and other applications or network connectivity, it is prone to attacks through these exposed surfaces. An application generally has some degree of cohesion and coupling exposing the gaps. Beside, ease of use calls for reduction in security and enhanced security reduces the ease of use. As the application has to be finally used by a person or a software, accessibilty has to be provided to the application, leading to an optimisation between accesibility and security. Even a 'black body' application is exposed to attacks from the data that it receives.

6. CONCLUSIONS

In this paper authors analysed the attack surface metrics and importance of OWASP compliance in web application by developing two web applications of similar funtionality, one followed the OWASP compliance while the other did not. More the attack surface of the application, more is the effort the tester has to put on testing and if the attack surface of the application is less, then the tester has to put less effort on testing. Similarly, the developer has to improve its quality of code, if the attack surface of the application is more. In

continuation of this research, the following is being attempted for future:

- Comparison between different web applications and examining their results.
- Calculation and analysis of attack surface using various proposed metrics.
- Finding an easier way for evaluating attack surface of any web application without having deeper web application development knowledge.

REFERENCES

1. Fenton, Norman E. & Neil, Martin. A critique of software defect prediction models. *IEEE Trans. Softw. Eng.* 1999, **25**(5), 675-89.
2. OWASP 2007, The ten most critical web application security vulnerabilities 2007 update, https://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf (accessed on 16 August 2012)
3. Heumann, Thomas; Türpe, Sven & Keller, Jörg. Quantifying the attack surface of a web application. In Proceedings of Sicherheit'2010, July 2011, LNI, 170, pp.305-316,
4. Measuring web application security coverage. http://fanaticmedia.com/infosecurity/archive/April 11/Measuring WebAppSec Coverage final.htm (Accessed on April 2011)
5. scitz, Justin & Niem, Joey. Analyzing attack surface code coverage. 2007, SANS Institute , http://www.sans.org/reading_room/whitepapers/application/analyzing-attack-surface-code-coverage_1996, (Accessed on 16 August 2012)
6. Yonghee, Shin; Andrew, Meneely; Laurie ,Williams and Jason, A. Osborne. 2011. Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities. IEEE Trans. Softw. Eng. 2011, **37**(6), 772-87.
7. Walden, J. & Doyle, M. SAVI: Static-analysis vulnerability indicator. IEEE Security Privacy, 2012, 10(3), 32-39.
8. Manadhata, P.K. & Wing, J.M. An Attack Surface Metric. IEEE Trans. Software Eng., 2011, **37**(3), 371-86.
9. Howard, M. Fending off future attacks by reducing attack surface. http://msdn.microsoft.com/library/default.asp? url=/library/en-us/dncode/html/secure02132003.asp, 2003. (Accessed on 16 August 2012)
10. Howard, M.; Pincus, J. & Wing, J. Measuring relative attack surfaces. In Proceedings of Workshop on Advanced Developments in Software and Systems Security, 2003
11. Manadhata, Pratyusa K., Tan Kymie M.C., Maxion, Roy A. & Wing Jeannette M. An approach to measuring a system's attack surface. Aug-2007, http://reports-archive.adm.cs.cmu.edu/anon/2007/CMU-CS-07-146.pdf (Accessed on August 2012)
12. Ha Thanh, Le & Loh, P.K.K. Evaluating AVDL descriptions for web application vulnerability analysis. In IEEE International Conference on Intelligence and Security Informatics, ISI 2008. 17-20 June 2008. pp.279-281.
13. Web application security fundamentals. http://msdn.microsoft.com/en-us/library/ff648636.aspx (Accessed on September 2011).
14. Lee, Vincent C.S. & Shao, Linyi. Estimating potential IT security losses: An alternative quantitative approach. IEEE Security Privacy, 2011, **4**(6), 44-52.
15. Web application development, www.icreonglobal.com/web-application-development.html, (Accessed on July 2011).
16. Manadhata, P. & Wing, J. An attack surface metric, in First Workshop on Security Metrics, Vancouver, BC, August 2011.
17. About internet application and web application server, http://livedocs.adobe.com/ coldfusion8/htmldocs/help.html?content=introducing-cf-2.html. (Accessed in September 2011).
18. Attacking web applications at the source. http://networksecurity.org.ua/0596007949/networkst-chp-6-sect-1.html. (Accessed on July 2011).
19. Lv, Teng & Yan, Ping. A web security solution based on XML technology. In International Conference on Communication Technology, 2006. ICCT '06, 27-30 Nov. 2006, pp.1-4.
20. Manadhata, P.K.; Karabulut, Y. & Wing, J.M. Report: Measuring the attack surfaces of enterprise software. In ESSoS 09: Proceedings of the 1st International Symposium on Engineering Secure Software and Systems (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 91–100, September 2011.
21. Manadhata , Pratyusa; Wing , Jeannette; Flynn, Mark & McQueen, Miles. Measuring the attack surfaces of two FTP daemons. In Proceedings of the 2nd ACM workshop on Quality of protection, 2006, Virginia, USA, October 2011.
22. Mitigate Security risks by minimizing the code you expose to untrusted users. msdn.microsoft.com/en-us/magazine/cc163882.aspx, August 2011.
23. Shar, Lwin Khin & Tan, Hee Beng Kuan. Defending against cross-site scripting attacks. Computer, 2012, **45**(3), 55-62.
24. Siddiqui, M.S. & Verma, D. Cross site request forgery: A common web application weakness. In IEEE 3rd International Conference on Communication Software and Networks (ICCSN), 2011 May 2011, pp. 538-43.
25. Boyan, Chen; Zavarsky, P.; Ruhl, R. & Lindskog, D. A study of the effectiveness of CSRF guard. In IEEE 3rd international conference on social computing (socialcom), 9-11 Oct. 2011. pp.1269-272.
26. Munakata, S. & Hiji, M. A session management method to improve web applications usability on mobile network. In IEEE Region 10 Conference, TENCON 2006. 14-17 Nov. 2006. pp.1-4.

Contributors

Mr Sumit Goswami obtained his MTech (Comp. Sci. & Engg.) from IIT Kharagpur. Presently working as Scientist 'E' at DRDO, New Delhi. His areas of interest include network centric operations, mobile ad hoc and sensor networks, web-hosting security, text mining and machine learning. He has published 53 papers/chapters in various journals, books, data competitions and conferences.

Ms Nabanita Radhakrishnan obtained her BTech (Elect. & Comm. Engg.) from Guindy Engineering College, Chennai and M Tech (Electrical Engg.) from IIT Madras. Presently working as Director, Management Information System and Technologies (MIST) at DRDO Hqrs, New Delhi. In this capacity she has conceptualized and commissioned an upgraded DRDO Intranet with a multi-tier security infrastructure and a number of software applications. She is a Member of Aeronautical Society of India and Instrument Society of India.

Mr Mukesh obtained his MCA from IGNOU, Delhi and MSc (Computer Science) from MDU Rohtak. Presently working as Senior Technical Assistant at DRDO HQr. His research area include: Software development, website designing and hosting, Linux, Windows, MySQL, JAVA, JSP, ORACLE, Crystal Report, Visual Basic, and PHP.

Mr Saurabh Swarnkar obtained his BE (Comp.Sci. & Engg.) from Institute of Information Technology and Management, Gwalior, and PGDAC from CDAC- Advance Computing Training School, Bengaluru. Presently working as a Programmer in IAP Company Ltd, Gurgaon. His research area include: Developing web application, web designing and hosting.

Ms Pallavi Mahajan is pursuing BTech (Computer Science and Engineering) from Beant College of Engineering and Technology, Gurdaspur, Punjab. She is presently doing 6 months internship from DRDO, New Delhi.