Web Attacking Parameters Filtration: a Approach for Attack Signature Verification

Surendra Kumar Shukla, Romil Rawat, Cns Murthy

Abstract– Web site is the global needs for business, study and government sectors, but the usage of web application has addicted the peoples. There is no proficiency for the user at several levels such as security, context management, web applications and related information. Attacker has also been activated to steal or destroy the confidential and most secured data. In past various attacks has been notified with severe disastrous result. For stopping and detecting these attacks, various techniques and tools have been manufactured, but they are not 100% result oriented. In this paper we have presented various types of web attack and also different methods and techniques to detect and prevent them, finally we have evaluated these web attacks by different approaches.

I. INTRODUCTION

A typical "Web *Applications"* comprises three subsystems: the browser, the server, and back-end systems (see Figure 1). The browser constitutes the platform for the user interface. It renders HTML code, images, and other data visible, handles user input, and provides the runtime environment for client-side code, Helper application such as PDF reader and plug-in components. A browser process at a point in time, and often also the entire browser installation, is associated with a particular user as a personal software tool. Contrary to the impression that Figure 1 might give multiple instances may exist for each of the subsystems with complicated relationships between the instances. In particular, a browser may access multiple web servers simultaneously. The web server (Figure 1) waits for browsers to connect and answers their HTTP requests. ***[1]***

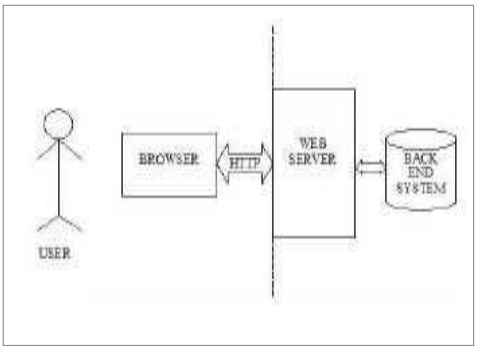

Figure 1: High-level architecture of a typical web application & inter-communication

Each request comprises a URL pointing to some resource and possibly additional parameters. The server responds either by serving a static resource, or by executing a program and returning the output of this program to the requesting browser. In the course of fulfilling a request the server, or a program executed by the server, may access back-end systems such as databases. The purpose of "Attack *Surface of a Web Application metric"* is to estimate the amount of functionality and code that a web application exposes to outside attackers. Web applications comprise a natural security boundary, indicated by the dashed line in Figure 1 users and attackers alike normally are not granted access to the web server and back-end systems other than through the HTTP interface of the web server. We further exclude from our consideration attacks that primarily target the users of an application, such as phishing attacks to get their passwords. Accessible to the outside attacker, who may or may not also be a user of the application are therefore the HTTP interface(s) of the web server(s), any server-side or back-end functionality accessible through the server, and any data displayed and code executed on the browser side, including the browser itself. ***[2]***

II. APPROACH, REQUIREMENTS AND ASSUMPTIONS

An application attack surface metric is a tool for developers and testers. Developers use it, for instance, to assess the impact of design and implementation decisions,
whereas security testers may base on the metric estimates of required testing effort or of the expected number of defects.

Manuscript Received on June, 2012

Surendra Kumar Shukla, Department of CSE, Chameli Devi School of Engineering Indore M.P. India
Romil Rawat, Department of CSE, Chameli Devi School of Engineering Indore M.P. India
Dr. CNS Murthy, Department of CSE, Chameli Devi School of Engineering Indore M.P. India

Our metric is designed with practical applicability for testers in mind, which entails a number of further requirements. We are aiming for these properties:

Attack surface approximation
Universal applicability
Grey-box restructurability
Support qualitative comparison ***[3]***
Relevant measurements

In addition to these requirements we make two assumptions. First, we limit our considerations to external attacks, excluding insider threats from our consideration. Second, we focus on attack surface elements with immediate exposure to external attackers. This means we will not attempt to distinguish applications, for instance, by their use of back end subsystems or interaction with the operating system of the server host.

III. PRAPOSED ATTACKS

A. Remote code execution

As the name suggests, this vulnerability allows an attacker to run arbitrary, system level code on the vulnerable server and retrieve any desired information contained therein. Improper coding errors lead to this vulnerability. At times, it is difficult to discover this vulnerability during penetration testing assignments but such problems are often revealed while doing a source code review.

B. XML-RPC for PHP attacks

Allow a remote attacker to execute code on a vulnerable system. An attacker with the ability to upload a crafted XML files & executed the Web application that is using the vulnerable (ML-RPC code) & exploiting register_globals (that controls the availability of "superglobal" variables).

C. Cross-site Timing Attacks

With direct attacks, it is only possible to see the ‘public’ side of the web. If one could make requests as another user, using that user’s preferences and login credentials, it would be possible to find out information that is visible to that user alone. Since these preferences and credentials are typically sent automatically in a cookie, we merely need to time these cookie-enabled requests. Web browsers have taken many steps to prevent one web site from learning anything about requests made by the user’s browser to other sites. This broad class of attacks, known as cross-site, has been known and studied for some time, but remains a large source of problems on the web. Despite the presence of different preventative measures in modern web browsers, we can nevertheless still time Cross-site content. Browser timing techniques Given that JavaScript is the most common form of dynamic content on the web, it will come as no surprise that it forms the basis for the most reliable method of timing cross-site content. JavaScript itself is typically prohibited from learning anything about the content of any data that is not hosted on the same domain as the page containing the script; this is a direct application of the same-origin principle. ***[4]***

D. SQL Injection

SQL injection is a very old approach but it's still popular among attackers. This technique allows an attacker to retrieve crucial information from a Web server's database. Depending on the application's security measures, the impact of this attack can vary from basic information disclosure to remote code execution and total system compromise. It is obvious that these error messages help an attacker to get a hold of the information which they are looking for (such as the database name, table name, user names, password hashes etc). Thus displaying customized error messages may be a good workaround for this problem, however, there is another attack technique known as “Blind *SQL Injection”* where the attacker is still able to perform a SQL injection even when the application does not reveal any database server error message containing useful information for the attacker.

E. Cross-Site Scripting Attack

The success of this attack requires the victim to execute a malicious URL which may be crafted in such a manner to appear to be legitimate at first look. When visiting such a crafted URL, an attacker can effectively execute something malicious in the victim's browser. Some malicious JavaScript, On a search engine that returns 'n' matches found for your '$_search' keyword. Within discussion forums that allow script tags, which can lead to a permanent XSS bug. On login pages that return an error message for an incorrect login along with the login entered. ***[5]***

F. User name enumeration

It is a type of attack where the back-end validation script tells the attacker if the supplied user name is correct or not. Exploiting this vulnerability helps the attacker to experiment with different user names and determine valid ones with the help of these different error messages. User name enumeration can help an attacker who attempts to use some trivial usernames with easily guessable passwords, such as test/test, admin/admin, guest/guest, and so on. These accounts are often created by developers for testing purposes, and many times the accounts are never disabled or the developer forgets to change the password. During pen testing assignments,the authors of this paper have found such accounts are not only common and have easily guessable passwords, but at times they also contain sensitive information like valid credit card numbers,
passport numbers, and so on. Needless to say, these could be crucial details for social engineering attacks.

G. PDF Attacks

It was actually started with designing of PDF back doors. The targeted version was version Adobe Reader 8. We have already seen URI handling flaws encompassing command execution in standalone PDF documents. The URI flaw worked fine. The Adobe took stringent steps to patch the stated flaws in an effective manner. The PDF provides a functionality which is always interactive. We are mainly concerned with the attacks that require user interaction. This attack is composed of designing a malicious PDF form with no standard inputs. The POST call in HTTP object is used to dispatch the JavaScript code directly to the domain which hosts that PDF file. No Security check is performed on In line JavaScript that uses protocol handler. [6]

H. Session fixation

As mentioned above, web session security is mainly focused on preventing the attacker from obtaining either intercepting, predicting or brute-forcing a session ID issued by the web server (also called "target *server"* in this paper) to the user's browser. This approach, however, ignores one possibility: namely the possibility of the attacker *"issuing"* a session ID to the user's browser, thereby forcing the browser into using a chosen session. We'll call this class of attacks "session *fixation"* attacks, because the user's session ID has been fixed in advance instead of having been generated randomly at login time. In a session fixation attack, the attacker fixes the user's session ID before the user even logs into the target server, thereby eliminating the need to obtain the user's session ID afterwards. Attack process generally, session fixation attack is a three-step process-

Session setup: First, the attacker either sets up a so- called "trap ***session"*** on the target server and obtains that session's ID, or selects a – usually arbitrary – session ID to be used in the attack. In some cases, the established trap session needs to be maintained (kept alive) by repeatedly sending requests referencing it to avoid idle session timeout.

Session fixation: Next, the attacker needs to introduce her session ID to the user's browser, thereby fixing his session.

Session entrance: Finally, the attacker has to wait until the user logs in to the target server using the previously fixed session ID and then enter the user's session. [7]

I. Unicode Attacks

The biggest cornerstone of making these characters from different languages possible relies on the utilization of Unicode. The Universal Character Set (UCS) is a repertoire using Unicode for all characters we may use. The most popular version of UCS uses a 16 bit number to represent a character code. There are a lot of visually similar characters coexisting in the UCS. The possibility of using similar characters to generate fake domain name using national alphabets is firstly reported in [8] and named "Homograph *Attack".* This is a general concept. Our efforts focus on the survey of UCS which is the
most populated character set for the internationalization of Web information, thus we would like to give it a more specific name, "Unicode *Attack".* The "Unicode *Attack"* is more than just faking domain name. [9]

Phishing Attack: malicious people (phishers) could use visually similar characters to mimic a real Internationalized Resource Identifier (IRI) [10, 11]. Another possibility is that an original Web page could be mimicked by similar characters such that certain existing Anti-Fishing systems [12] would fail to catch this kind of attack because they must find sensitive word(s) in emails before actual comparison.

Web Identity Faking/Attack: malicious people (pretenders) may use similar user names to pretend another user's identity. Many Web based systems (Website, Email, Instant Message, Blog, Wiki, etc.) utilize text string to represent the user names. There will be no problem if only ASCII characters are permitted to use as a user name.

Spamming Attack: malicious people (spammers) may create numerous spams while keeping the appearance of the original email.

IV. DEFENSES/ COUNTERMEASURES AGAINST ATTACKS

A. Counterattacks for User name enumeration

Display consistent error messages to prevent disclosure of valid usernames. Make sure if trivial accounts have been created for testing purposes that their passwords are either not trivial or these accounts are absolutely removed after testing is over - and before the application is put online. Edit the source code so that the input is properly verified.

B. For Cross-site timing attacks

Counting the number of items in a user's shopping cart from the database and then selectively decides which ones to display will be vulnerable to leaking the total number of records. One could look for such coding patterns to detect basic timing attacks and correct them, but this is likely to be very error-prone. One defense is to ensure that the web server always takes a constant amount of time to process a request. Blaze ***[13]*** proposed an operating system level mechanism for doing so. A similar system could be built for web servers. However, simply ensuring that total request time is constant is insufficient. If the server is using chunked encoding, inter-chunk timings could reveal sensitive information,
even though the total response time is constant. For chunked encoding it is critical that all inter-chunk times are constant. We implemented this specific defense as an Apache module called mod time pad.

C. Countermeasures SQL Injection

Avoid connecting to the database as a super user or as the database owner. Always use customized database users with the bare minimum required privileges required to perform the assigned task.

D. For PDF attacks

Never allow the PDF file to be opened in the browser itself.
In designing custom application with different user roles appropriate check should be applied on the type of file being uploaded. The content-disposition parameter should be applied in a right manner because it explains the behavior of content whether it should be used inline or an attachment. The RFC 2183 clearly explains about the security problems. It is advisable the content should be treated as attached and not inline in the body part. Avoid inline opening of PDF documents. The filters should be applied appropriately so that content is not rendered as dynamic. Use of flash to display the PDF document is a good solution to prevent these types of attacks. ***[6]***

E. Restricting the session fixation

Preventing logins to a chosen session & Preventing the attacker from obtaining a valid session ID. Binding the session ID to the browser's network address (as seen by the server) Binding the session ID to the user's SSL client certificate - very important and often overlooked issue in highly critical applications: *each* server-side script must first check whether the proposed session was actually established using the supplied certificate. Session destruction, either due to logging out or timeout, must take place on the server (deleting session), not just on the browser (deleting the session cookie).The user must have an option to log out – thereby destroying not just his current session, but also any previous sessions that may still exist (in order to prevent the attacker from using an old session the user forgot to log out from).[7]

F. Countermeasures Remote code execution

More recent PHP versions have register_globals set to off by default; however some users will change the default setting for applications that require it. This register can be set to "on" or "off" either in a php.ini file or in a .htaccess file. The variable should be properly initialized if this register is set to "on." Administrators who are unsure should question application developers who insist on using register_globals. It is an absolute must to sanitize all user input before processing it. As far as possible, avoid using shell commands. However, if they are required, ensure that only filtered data is used to construct the string to be executed and make sure to escape the output.

G. Binding Sensitive Information to ConfirmationTokens

The first solution is based on *confirmation tokens.* In principle, the concept of a confirmation token is similar to a transaction number (i.e., TANs) commonly used in online banking. TANs are randomly generated numbers that are sent to customers as hard copy letters via regular (snail) mail. Each time a customer would like to confirm a transaction, she selects a TAN entry from her hard copy list and enters it into the web application. Each TAN entry can be used only once. The idea is that an attacker cannot perform transactions just by knowing a customer's user login name and password.

"We propose to bind the information that the user wants to send to our confirmation token". In other words, we propose to use confirmation tokens that (partially) depend on the user data. Note that when using confirmation tokens, our focus is not the protection of the confidentiality, but the integrity of this sensitive information. [14]

H. Using CAPTCHAs for Secure Input

Graphical input is used by some banks and other institutions to prevent eavesdropping of passwords or PINs. Instead of using the keyboard to enter sensitive information, an image of a keypad is displayed, and the user enters data by clicking on the corresponding places in the image. Unfortunately, these schemes are typically very simple. The basic idea of the second solution is to extend graphical input with CAPTCHAs [15]. A CAPTCHA, which stands for Completely Automated Public Turing test to tell Computers and Humans Apart, is a type of challenge-response test that is used in computing to determine whether or not the user is human. Hence, a CAPTCHA test needs to be solvable by humans, but not solvable (or very difficult to solve) for computer applications. CAPTCHAs are widely employed for protecting online services against automated malicious programs or scripts.

V. ATTACK SURFACE PARAMETERS

Degree of Distribution summarizes how an application spans multiple domains. Cross-domain issues are a common source of vulnerability. Distribution makes such issues more likely as it requires the application to work around the same-origin policy enforced by web browsers to separate resources of different origins.

Page Creation Method the parameter represents the binary distinction between applications that dynamically create pages on the server side and those that do not. Server-side application code is a source of vulnerability.

Security Mechanisms the security group of components of the attack surface vector represents the use of common security mechanisms. We consider two mechanisms, Transport Layer Security (TLS) and input validation

Input Vectors The HTTP interface between client and server supports a number of input vectors. The more vectors an application uses, the more complex it is.

Active Content increases the attack surface on the client side, requiring the user to have plug-ins or helper applications.

Cookies constitute another input channel into the application; they are often used to implement user authentication and session management; cookies can be used to track users; and leave easy-to-observe traces in the browser.

Access Control The parameter role reflects the user status and can assume one of three values: unauthenticated, authenticated or root and linked to an identity, certain access rights are associated with his identity. ***[1]***

VI. EXPERIMENTAL EVALUATION

We validate our attack surface metric by applying it to several web sites. This demonstrates how to use the metric, whether its application is practical in real-world scenarios, and whether the results are in line with our expectations with used attacks & their possible solutions. And we compare it to the theoretical extremes.

A. Sample Web Applications

We chose set of evaluation samples to cover a range of different web sites and applications (PHP) but with subsets of multiple similar applications. Simple websites primarily are serving pages with information. We use *gaos.org* (an open source software advocacy group and built using a wiki engine) and *testlab.sit.fraunhofer.de* (research group of one of the authors.) as samples. Several web-based document management systems developed independently by different companies using different programming languages are Live-link *9.2, BSCW 4.4.5* and *cFolders 4.0.0.* All three are multi-user applications allowing their users to store and share files.

B. Results/ Quantitative Comparison

It depicts the attack surfaces measured for the sample applications. The indicator values that mentioned in section 5 exhibit several interesting properties. This results show same analysis as approximation of ***[1]..***

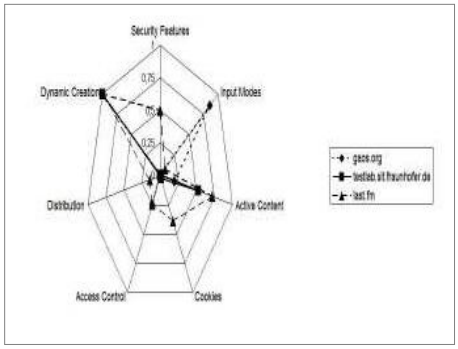

Figure 2: Qualitative comparison of the attack surfaces of web sites real-world sites or applications

First, they cover only about one third of the theoretical range. This does not come unexpected: the lower and the upper end of the range represent extreme cases that (Figure 2) is unlikely to reach

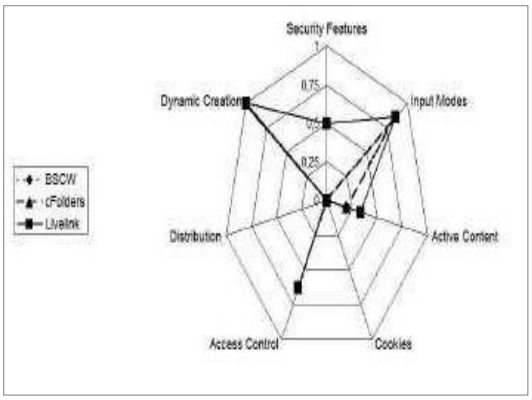

Figure 3: Qualitative comparison of the attack surfaces of document management applications

Second, and more surprisingly, the two apparently simple web sites reach indicator values close to those of the document management applications considered (Figure 3). This illustrates what the attack surface metric really measures. The two web sites have a high indicator value because they use similar technologies and concepts as web applications. With few exceptions our metric does not consider differences in complexity or size, but rather the mere presence or absence of features.

VII. CONCLUSION & FUTURE ROUTE

In this paper we've demonstrated web application attacks, their countermeasures and their criticality. If there is a consistent message among each of these attacks, the key to mitigate these attacks is to sanitize user's input before processing it. Through this paper we tried to show how the attackers use the approaches together to reach to sites with vulnerable products and then hack and deface them.

The attack surface metric defined in this paper is a first attempt to measure vulnerability expectations for the class of web applications. A first evaluation with existing applications indicates a good relation of our proposal with expectations. In future work, we would like to compare our approach to other metrics ***[15].*** Understand the impact of correlations. Not all parameters used in the metric are independent. For instance, each foreign cookie implies involvement of a foreign web site, but not vice versa. Further work in these directions may ultimately lead to some kind of security metric construction kit providing building blocks

REFERENCES

1. Thomas Heumann, J¨org Keller, Sven T¨urpe; "Quantifying the Attack Surface of a Web Application". OWASP Top 10 – 2010. available online.
2. http://www.owasp.org/index.php/,Category: OWASP_ Top_Ten_Project, 2010.
3. Peter Mell, Karen Scarfone, and Sasha Romanosky. "A Complete Guide to the Common Vulnerability Scoring System", http://www.first.org/cvss/cvss-guide.pdf, June 2007."
4. Andrew Bortz, Dan Boneh, Palash Nandy. "Exposing Private Information by Timing Web Applications". WWW 2007, May 8–12, 2007, Banff, Alberta, Canada. ACM 9781595936547/07/0005.
5. SecurityFocus, "Microsoft IIS IDC Extension Cross Site Scripting Vulnerability "http://online.securityfocus. com/bid/5900/info/
6. Aditya K Sood, SecNiche Security. "PDF Silent HTTP Form Repurposing Attacks Web Penetration Testing" Mitja Kolšek, ACROS Security. "Session Fixation Vulnerability in Web-based Applications" Version 1.0 revision 1. December 2002 (Revised February 2007 – The Acknowledgments section).
7. Gabrilovich E. and Gontmakher A., "The Homograph Attack, Communications of the ACM" 45(2), pp.128,2002
8. Anthony Y. Fu Wan Zhang Xiaotie Deng Liu Wenyin. "Safeguard against Unicode Attacks: Generation and
9. Applications of UC-SimList" Duerst M., Suignard M., "RFC 3987: Internationalized Resource Identifiers (IRIs)", The Internet Society, 2005.
10. Fu A. Y., Deng X., Liu W., "A Potential IRI based Phishing Strategy", WISE2005, LNCS Vol. 3806,pp.618-619, 2005
11. Liu W., Deng X., Huang G, Fu Y., "An Anti-Phishing Strategy based on Visual Similarity Assessment", IEEE Internet Computing (2), pp. 58-65, Mar/Apr. 2006.
12. Matt Blaze. Simple UNIX time quantization package. Previously available on the web.Martin Szydlowski, Christopher Kruegel, Engin Kirda."Secure Input for Web Applications".
13. Jarrett Rosenberg. "Some Misconceptions About Lines of Code". In Fourth International Software Metrics Symposium (METRICS'97), volume 0, pages 137–142, Los Alamitos, CA, USA, November 1997. IEEE Computer Society.
14. Carnegie Mellon University. The CAPTCHA Project. http://www.captcha.net.
15. Gabrilovich E. and Gontmakher A., "The Homograph Attack, Communications of the ACM" 45(2), pp.128,2002
16. http://www.owasp.org/index.php/,Category: OWASP_ Top_Ten_