Title

Hardening Cookies in Web-based Systems for Better System Integrity
(*Penambahbaikan Penggunaan Cookie bagi Meningkatkan Integriti Sistem Berasaskan Web*)

Authors

Mohamad Safuan Bin Sulaiman, Mohd Dzul Aiman Bin Aslan, Saaidi Bin Ismail, Abd. Aziz Bin Mhd Ramli, Dr. Abdul Muin Bin Abdul Rahman, Siti Nurbahyah Binti Hamdan, Norlelawati Bt Hashimuddin, Sufian Norazam Bin Mohamed Aris.

Address

IT Center, Technical Support Division
Malaysian Nuclear Agency, Bangi
43000 Kajang, Selangor

Abstract

IT Center (ITC) as technical support and provider for most of web-based systems in Nuclear Malaysia has conducted a study to investigate cookie vulnerability in a system for better integrity. A part of the result has found that cookies in a web-based system in Nuclear Malaysia can be easily manipulated. The main objective of the study is to harden the vulnerability of the cookies. Two levels of security procedures have been used and enforced which consist of 1) Penetration test (Pen Test) 2) Hardening procedure. In one of the system, study has found that 121 attempts threats have been detected after the hardening enforcement from 23 March till 20 September 2012. At this stage, it can be concluded that cookie vulnerability in the system has been hardened and integrity has been assured after the enforcement. This paper describes in detail the penetration and hardening process of cookie vulnerability for better supporting web-based system in Nuclear Malaysia.

Abstrak

Pusat IT sebagai sebuah pusat yang menyediakan sokongan teknikal bagi kebanyakan sistem berasaskan web di Nuklear Malaysia, telah menjalankan kajian untuk menyiasat kelemahan bagi meningkatkan integriti pada sistem tersebut. Kajian telah mendapati bahawa, cookie dalam beberapa sistem berasaskan web di Nuklear Malaysia boleh dimanipulasikan. Objektif utama kajian ini adalah untuk memperbaiki kelemahan pada cookie tersebut. Dua peringkat prosedur keselamatan telah digunakan dan dikuatkuasakan yang melibatkan 1) Ujian Pencerobohan (Pen Test) 2) Prosedur penambahbaikan. Dalam salah satu sistem yang dikaji didapati bahawa 121 ancaman telah dapat dikesan selepas penguatkuasaan bermula dari 23 Mac sehingga 20 September 2012. Pada peringkat ini, dapatlah disimpulkan bahawa kelemahan yang terdapat di dalam sistem telah diperbaiki dan integriti telah ditingkatkan selepas penguatkuasaan. Kertas kerja ini menghuraikan secara terperinci kaedah pelaksanaan ujian pencerobohan dan proses penambahbaikan penggunaan dan keselamatan cookie untuk memberi sokongan terbaik terhadap sistem berasaskan web di Nuklear Malaysia.

Keywords: vulnerability, cookies, integrity, security, web-based system, penetration test, hardening procedure

1.0 Introduction

Integrity is one of important issues that need to be addressed very carefully in any web-based system. Data and information must be secure and well protected from being damaged, misuse or misconduct. In web-based system, vulnerability is one of the factors that can reduce integrity level of a system.

Malaysian Nuclear Agency has several web-based systems that have been developed in-house by application and system development team. The web-based systems that have been developed in-house includes 1) Electronic Punch Card System (ePC), Technical Helpdesk System (Tech Desk), Electronic Seminar Management System (eSEMs), Nuclear Malaysia International and localweb portals, Electronic Phone Book system and others. These systems used the same authentication procedure which capable of implementing single sign-on. With this feature, users can use one time sign-in with the same user name and password to access many systems.

This single sign-on technique is then become compulsory requirement for new outsourced systems since 2007. This is due to the growing number of web-based systems in agency which then leads to difficulties in managing user accounts and profiles. If it is not compulsory, it is a worry that every vendors will provide their own user's profile database for authentication and verification processes which leads to bad maintenance of user's information and it would be duplicated and contradict from one system to another when there are updates. Hence, users have to remember separate username and password for each system.

Implementation of single sign-on for web-based systems need efficient management of parameter passing, to simplify the process of every system in recognizing users that they are dealing with. Instead, users do not have to sign-in and out many times to access one systems to another. To do this, cookie's technique is used to manage user's profile for authentication and verification processes and would carry user's credential that determine user's access level of a system.

Even though cookies have been found to be efficient in parameter passing, but it may vulnerable if it is not well designed and protected. In Nuclear Malaysia, cookie has been applied in most of web-based systems. This is the most ideal way because cookie simplifies the authentication and verification processes, and profile information retrieval in supporting single sign-on. Therefore, IT Center, under application and system development team take proactive action by conducting a Penetration Test (Pen Test) to investigate the cookie vulnerability. The investigation has been made to one of the systems which must be maintained with high integrity due to its impact to agency. The test has been conducted in early 2012 which has found that the system's cookies vulnerable to data manipulation. Immediate and careful hardening activities have been taken seriously due to this situation.

2.0 Parameter Passing Techniques

In web-based system, several techniques can be used for parameter passing, they are 1) Cookie, 2) Forms, 3) Query string [1][2] and 4)Session [3]. Every parameter passing technique has its own advantages and disadvantages. Some functions and weaknesses of the techniques are shown in Table 2.1 below:

Techniques	Descriptions / Functions	Weakness
1. Cookie	It is also known as an **HTTP cookie**, **web cookie**, or **browser cookie**, is usually a small piece of data sent from a website and stored in a user's web browser while a user is browsing a website. When the user browses the same website in the future, the data stored in the cookie can be retrieved by the website to notify the website of the user's previous activity.[1] Cookies were designed to be a reliable mechanism for websites to remember the state of the website or activity the user had taken in the past. **Authentication cookies** are the most common method used by web servers to know whether the user is logged in or not, and which account they are logged in under. Without such a mechanism, the site would not know whether to send a page containing sensitive information, or require the user to authenticate himself by logging-in[4]	The security of an authentication cookie generally depends on the security of the issuing website and the user's web browser. If not implemented correctly, a cookie's data can be intercepted by a hacker to gain unapproved access to the user's data and possibly to the originating website.[5] [4]

2. Forms	A webform on a web page allows a user to enter data that is sent to a server for processing. Webforms resemble paper or database forms because internet users fill out the forms using checkboxes, radio buttons, or text fields. For example, Source: WIKI [5]. The form-data can be sent as URL variables (with method="get") or as HTTP post (with method="post"). **GET Method** appends the form-data to the URL in name/value pairs and useful for form submissions where a user want to bookmark the result. **POST Method** sends the form-data as an HTTP post transaction. Its Form submissions with the "post" method cannot be bookmarked and The "post" method is more robust and secure than "get", and "post" does not have size limitations Source: W3School [6]	**GET Method has** limit on how much data can be placed in a URL (varies between browsers), therefore, it is difficult to ensure that all of the form-data will be correctly transferred. Never use the **GET Method** to pass sensitive information! (password or other sensitive information will be visible in the browser's address bar) Source: W3School[6]
3. Querystring	It is the part of a Uniform Resource Locator (URL) that contains data to be passed to web applications such as CGI programs. The query string is a part of the URL which is passed to the program. Its use permits data to be passed from the HTTP client (often a web browser) to the program which generates the web page. [7] The Querystring collection is used to retrieve the variable values in the HTTP query string. The HTTP query string is specified by the values following the question mark (?). Query strings are also generated by form submission (GET Method), or by a user typing a query into the address bar of the browser. [8]	**Request.QueryString** cannot be used for sending large amounts of data (beyond 100 kb) [8]
4. Session	A Session object stores information about, or change settings for a user session. When working with an application on computer, and do some changes and then close it. This is much like a Session. The computer knows who you are. However, on the internet there is one problem: the web server does not know who you are and what you do, because the HTTP address doesn't maintain state. Active Server Page (**ASP)** (Microsoft based Web Programming Language) solves this problem by creating a unique cookie for each user. The cookie is sent to the user's computer and it contains information that identifies the user. This interface is called the Session object [9]. A **PHP** session (Open Source Based Web Programming) solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database. [10]	Different Language has different solutions. Only SMALL amounts of data stored in session variables **ASP**: It has time out. By Default it is 20 Minutes, Programmer can customize the setting according to system requirement. The main problem with sessions is WHEN they should end. But if the session is deleted too soon the user has to start all over again because the server has deleted all the information. Finding the right timeout interval can be difficult! [9] In **PHP**: session_destroy() will reset your session and you will lose all your stored session data. [10]

Table 2.1: Types of Parameter Passing Techniques

Based on Table 2.1, Cookie is the most ideal technique but it has to be well designed for better data protection. As mentioned earlier, most of the Nuclear Malaysia Internal web systems support single sign-on concept and authentication cookies have been chosen and used in this condition.

3.0 Cookies Vulnerabilities

Cookies [4] have been widely used for web based system to simplify the parameter passing process. Cookie is found to be effective and easy way in passing parameter, but it has its' consequent if it is not properly protected [11]. According to Linden [12], who has conducted vulnerability case study on cookie tampering; four types of cookie risks have been identified.

They are cookie theft, cookie poisoning, cookie inaccuracies and cross-site cooking. Details explanation of the vulnerabilities as in Table 3.1.

Cookie Risks	Details
Cookie Theft	Cookies are supposed to be sent only between the Web browser and the server or servers in the same domain that set the cookie. But, if the cookie is being sent over an ordinary HTTP connection, it is visible to anyone across the network using a packet sniffer. This means cookies really cannot contain sensitive information. This is sometimes overcome by using HTTPS to encrypt the connection, but that's not the solution because it only solves one of the problems associated with having sensitive data stored in cookies. For example, cross-site scripting can be used to send cookies to servers that should not be receiving that data. Encryption does not help stop this cookie theft, which is often done with a simple snippet of HTML posted to a site that users can be tricked into clicking on, and which sends their cookie for that site to a location that the attacker specifies. Because the request is coming from the same domain intended for the cookie, there are no problems. These cookies can then be exploited by connecting to the same site using the stolen cookies, thus spoofing the original owner of the cookie [12].
Cookie Poisoning	Cookies are supposed to be sent back to the server unchanged, but attackers can modify the value of a cookie before sending them back to the server. This is typically done to carry out some sort of attack against the server that relates to some sort of data contained in the cookie. For example, if a cookie contains the price per item for something in the shopping basket, a change to this value in the cookie may cause the server to charge the user a lower price for that item. This process of modifying a cookie before it is sent back to the server is called *cookie poisoning.* Sometimes, cookie poisoning is used after cookie theft. Most Web sites only store a randomly generated unique session identifier in the cookie, and everything else is stored on the server. This pretty much eliminates the threat of cookie poisoning [12].
Cookie Inaccuracies	Even outside of deliberate cookie tampering, there are aspects to how cookies are used in some situations that cause cookies to carry inaccurate data on which to base sensitive transactions. One issue is that separate people using the same computer, browser, and user account will unavoidably share cookies. Another is that if a single user uses multiple browsers, multiple computers, or multiple user accounts, it means that the user will have multiple sets of cookies [12].
Cross-Site Cooking	Despite the fact that each site is supposed to have its own set of cookies and to not be able to alter or set cookies for any other site, there are some browser flaws that cause cross-site cooking vulnerabilities and allow malicious sites to break this rule. This is similar to cookie poisoning, but instead of the attacker attacking the site itself, the attacker is now attacking non-malicious users with vulnerable browsers. Many browsers have a flaw in how they deal with overly relaxed cookie domains. They are supposed to require a two-dot specification for all domains under the following top-level domains which includes .com, .edu, .net, .org, .gov, .mil, .int . This is supposed to prevent the setting of a cookie for a sub domain like .com. So, the actual intent is that if you wish to set a cookie on a .com site, you need to specify ".mysite.com" as the two-dot name. This breaks when you get to the international naming system for some domains like .com.au or .com.de. In some browsers, it's possible for a cookie to be set for the entire .com.au (for example) domain. Another problem is how some browsers deal with periods. There is typically no check to see whether there is anything between the periods or if there is trailing period being used to override the local domain search path. This means that a cookie can be set for ".com.," which then sends the user to http://www.mywebsite.com./ This address is not the "real" one, but how many users will care about the trailing period? Probably not very many. Even some seasoned users may not be adequately suspicious. The third issue is that attackers can force cookies on random visitors, which are then relayed to a third-party site by setting IN A record and then redirecting users to the third-party site [12].

Table 3.1: Types of Cookie Risks and Vulnerabilities

It is also very important to properly handle those risks in Nuclear Malaysia internal web based systems and applications. Those risks have motivated proactive action of ITC to conduct a Pen test to measure how much the risks have affected the systems.

4.0 Encryption Techniques

Cryptographic is one of the important part of the system protection and design to ensure that the cookie risks and vulnerabilities can be properly hardened. There are many cryptographic techniques in the world but cryptographic hash function is one of the most secure techniques that usually used to secure online cookies. A cryptographic hash function is a hash function, that is, an algorithm that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an (accidental or intentional) change to the data will (with very high probability) change the hash value. The data to be encoded is often called the "message," and the hash value is sometimes called the message digest or simply digests. The ideal cryptographic hash function has four main or significant properties which includes 1) Easy to compute the hash value for any given message, 2) Infeasible to generate a message that has a given hash, 3) Infeasible to modify a message without changing the hash and 4) Infeasible to find two different messages with the same hash.

Cryptographic hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as

ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information security contexts, cryptographic hash values are sometimes called (digital) fingerprints, checksums, or just hash values, even though all these terms stand for functions with rather different properties and purposes. Comparison of Cryptographic hash algorithms [13] [14] is shown in Table 4.1 below:

Algorithm	Output size (bits)	Internal state size[c 1]	Block size	Length size	Word size	Rounds	Best known attacks (complexity:rounds)[c 2]		
							Collision	Second preimage	Preimage
GOST	256	256	256	256	32	256	Yes (2105)	Yes (2192)	Yes (2192)
HAVAL	256/224/192/160/128	256	1,024	64	32	160/128/96	Yes	No	No
MD2	128	384	128	-	32	864	Yes (263.3)	No	Yes (273)
MD4	128	128	512	64	32	48	Yes (3)	Yes (264)	Yes (278.4)
MD5	128	128	512	64	32	64	Yes (220.96)	No	Yes (2123.4)
PANAMA	256	8,736	256	-	32	-	Yes	No	No
RadioGatún	Up to 608/1,216 (19 words)	58 words	3 words	-	1–64	-	With flaws (2352 or 2704)	No	No
RIPEMD	128	128	512	64	32	48	Yes (218)	No	No
RIPEMD-128/256	128/256	128/256	512	64	32	64	No	No	No
RIPEMD-160	160	160	512	64	32	80	Yes (251:48)	No	No
RIPEMD-320	320	320	512	64	32	80	No	No	No
SHA-0	160	160	512	64	32	80	Yes (233.6)	No	No
SHA-1	160	160	512	64	32	80	Yes (251)	No	No
SHA-256/224	256/224	256	512	64	32	64	Yes (228.5:24)	No	Yes (2248.4:42)
SHA-512/384	512/384	512	1,024	128	64	80	Yes (232.5:24)	No	Yes (2494.6:42)
Tiger(2)-192/160/128	192/160/128	192	512	64	64	24	Yes (262:19)	No	Yes (2184.3)
WHIRLPOOL	512	512	512	256	8	10	Yes (2120:4.5)	No	No

Table 4.1: Comparison of Cryptographic Hash Algorithms

SHA-1 produces a 160-bit message digest based on principles similar to those used by Ronald L. Rivest of MIT in the design of the MD4 and MD5 message digest algorithms, but has a more conservative design. The original specification of the algorithm was published in 1993 as the Secure Hash Standard, FIPS PUB 180 [15] [16], by US government standards agency NIST (National Institute of Standards and Technology). This version is now often referred to as *SHA-0*. It was withdrawn by National Security Agency (NSA) shortly after publication and was superseded by the revised version, published in 1995 in FIPS PUB 180-1 and commonly referred to as *SHA-1*. SHA-1 differs from SHA-0 only by a single bitwise rotation in the message schedule of its compression function; this was done, according to NSA, to correct a flaw in the original algorithm which reduced its cryptographic security.

However, NSA did not provide any further explanation or identify the flaw that was corrected. Weaknesses have subsequently been reported in both SHA-0 and SHA-1. SHA-1 appears to provide greater resistance to attacks, supporting the NSA's assertion that the change increased the security. The comparison of SHA functions [17] as in Table 4.2.

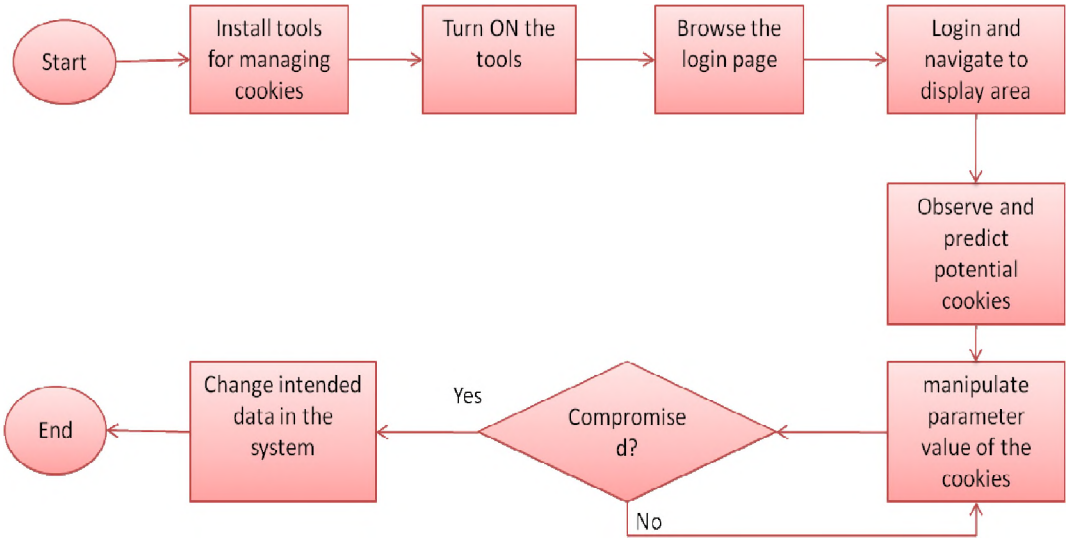
Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Operations
SHA-0								
SHA-1		160	160	512	$2^{64} - 1$	32	80	add, and, or, xor, rotate, mod
SHA-2	*SHA-256/224*	256/224	256	512	$2^{64} - 1$	32	64	add, and, or, xor, rotate, mod,
	SHA-512/384	512/384	512	1024	$2^{128} - 1$	64	80	

Table 4.2: Comparison of SHA Functions

5.0 Methodology: Penetration Test (Pen Test)

Two security procedures have been used to overcome the problem of this study namely 1) Penetration test and 2) Hardening process. Figure 5.1 shows the processes involved in penetration test which purposely to compromise the system through manipulation of its cookies.

Figure 5.1: Penetration Test Process for Cookies Manipulation

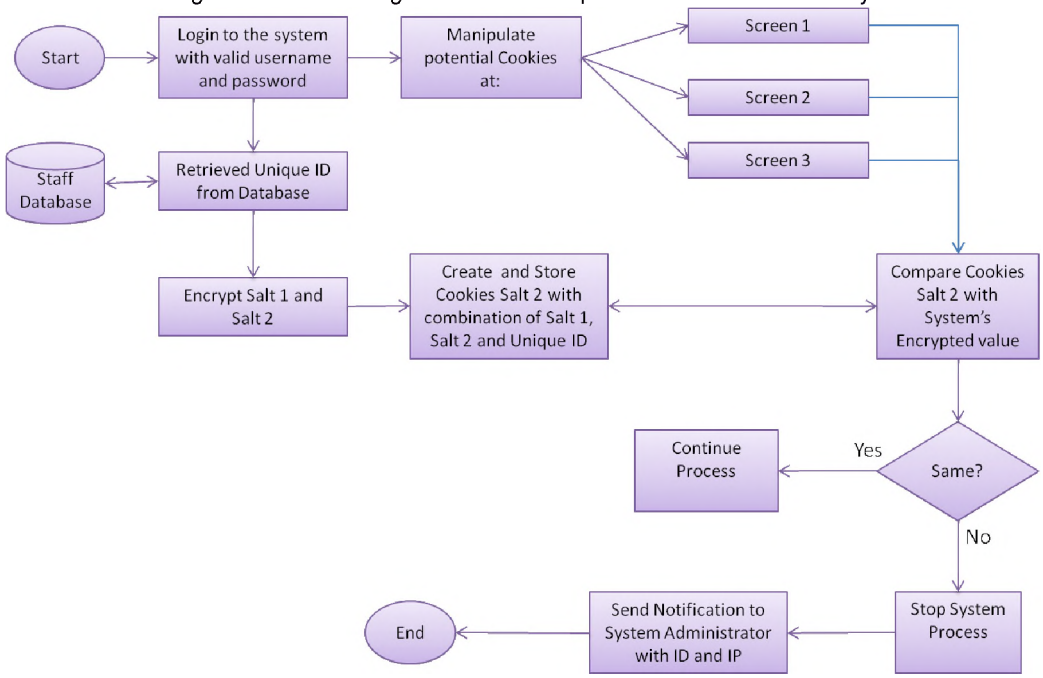


In this process, usually, with minimum information at hand, many attempts should be made until the system can be compromised. In doing this, the tester should make accurate judgement on parameter value of the potential cookies which hold the key information that could possibly used to compromise the system's credential. Once the key information has matched with the value, system's credential has been compromised and any modification at data level could be made by the tester which seems like it is a valid access.

6.0 Methodology: Hardening Procedure

It is always a serious issue once dealing with system credentials. Therefore a hardening procedure should be prepared immediately to overcome the cookies vulnerability. Figure 6.1 shows the hardening processes to protect the system from the attack. In this procedure, potential vulnerable cookies have been treated through encryption so it cannot be easily manipulated.

Figure 6.1: Hardening Procedure to Improve Cookies Vulnerability



At the very beginning stage of the user logging into the system, values of all potential cookies will be encrypted using customized SHA-1 technique. In this case, several cookies are created for administrative use to track information of the suspect that attempting to compromise the system. After the user logged into the system, any changes made to value of the authentication cookie will be validated with the encrypted value that is retrieved from the database in several modules until the user logged out. Once system found that the value of the cookie and encrypted value retrieved from database are not matched, system process will be automatically halted and warning message of the attempt will be displayed on user's screen. At the same time, system generates email message to administrator reporting the attempt with the suspect's profile and network access information. This hardening procedure has been fixed to the system since 28 March 2012 and has been tested several times. System warning message as in Figure 6.2, is displayed to the user once the system detected the attempt. Email message reporting the attempt to system administrator as in Figure 6.3.

Figure 6.2: System Warning Message Prompted to Users.

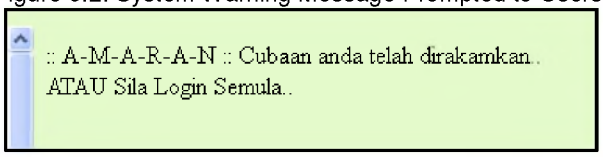
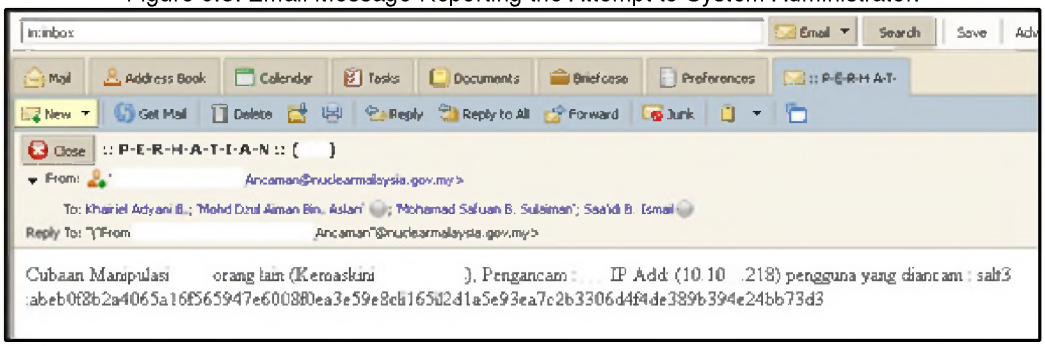


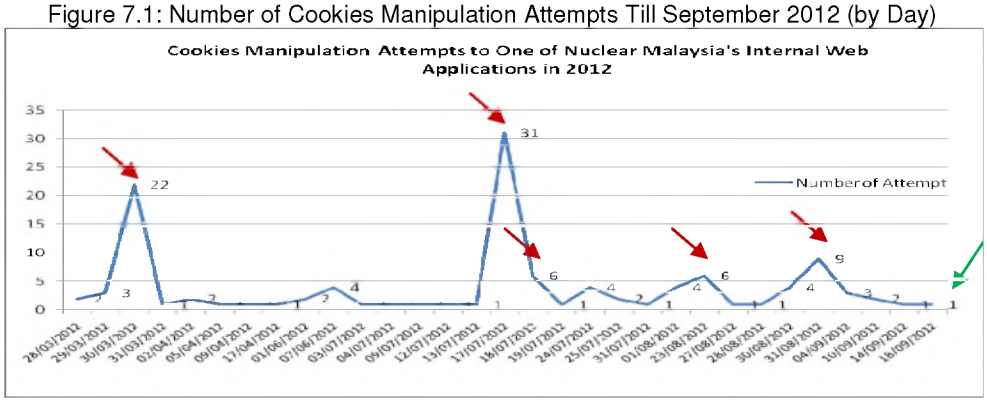
Figure 6.3: Email Message Reporting the Attempt to System Administrator.



7.0 Result and Discussion

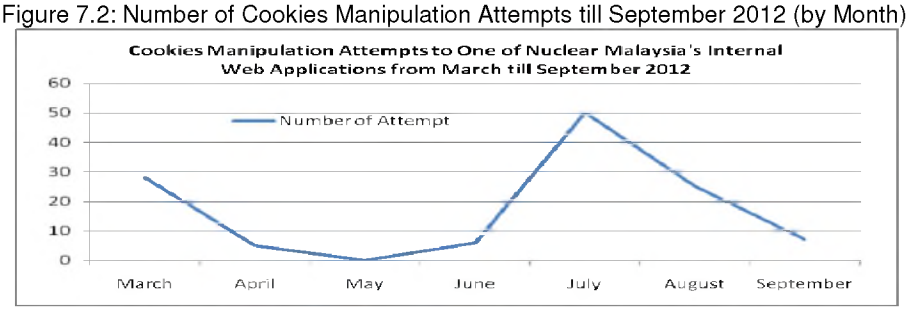
Data has been collected through email notification message received by the administrator during the attempt from March till September 2012. Data has been analyzed and the graph has been plotted using Microsoft Excel. Within this period, two improvements have been made on the system especially on the encryption levels and techniques to harden the cookie vulnerability. Line Graph in Figure 7.1 (by Day), shows that the number of cookies manipulation attempts to one of Nuclear Malaysia's Internal web Applications in 2012. In Figure 7.1 (by Day), shows that the number of cookie's manipulation attempts. Most attempts were in March (22 attempts) and July (31 attempts). Both months have the obvious number of attempts because the testing for the improvements have been made in both months. Other months remain attempts less than 10.

Figure 7.1: Number of Cookies Manipulation Attempts Till September 2012 (by Day)



After the hardening process to improve the cookie vulnerability fixed into the system in July 2012, the numbers of attempt have significantly reduced over time. This can be seen in Figure 7.2 (by Month), it has dramatically reduced from July to September.

Figure 7.2: Number of Cookies Manipulation Attempts till September 2012 (by Month)



However, in Figure 7.2 (by Day), number of attempts were increased in the middle and towards the end of every month. These can be seen from the dates which have red arrows pointing to the data in the graph. It is a big question on how the attempts could be occurred. Additional observations have been conducted by application and system development team and it was finally answered that why the phenomena happened in that situation. The attempts usually happen in the middle and toward the end of every month due to the system usage that need users to update their activities during the period of time. It is found that, some users use the same computer to access different user profiles. It means that, same browser hold at least two data for the same cookies variables. Once the system detected the latest cookies variables differed from the one that has been recorded earlier, the system considered it as an attempt, the system halted the remaining processes and request the user to relog-in, to reset the value of all related cookies. This situation could happen if the users did not maintain historical data in the browser properly which older cookie information is still stored in the system which may confuse the system during verification process.

In the view of system's passing parameter technique, selection on authentication cookie for single sign-on implementation is considered as acceptable technique in which other techniques have some shortcomings and security issues. Although cookie is not 100% secure from threats, something could be done to make it more secure by adapting some encryption techniques. Customized SHA-1 encryption has been used in this work and it is found usable and secure in present environment.

In addition, hardening process using encrypted data with validation technique could secure the system from those mentioned cookie vulnerabilities and risks. It is also important to conduct a pen test once there is cookie technology update with new penetration objective to improve and ensure system security and integrity.

8.0 Conclusion and Future Works

In conclusion, this study has met its objective by conducting a Pen Test and hardening the system from cookie manipulation as resulted from the test. Hardening process has involved secure and usable techniques which include customized SHA-1 encryption and system verification to improve cookie vulnerability for system security and integrity. Number of attempts have significantly reduced which indicated that the system is secure and reliable. In future works, more secure environments will be further explored in which HTTPS and VPN platforms would be the area to be discovered due higher demand from users in Nuclear Malaysia to access internal web applications globally.

9.0 Acknowledgements

The author would like to express special thanks to higher management of Nuclear Malaysia for providing platform and better supports to conduct this kind of research. Special thanks to Nuclear Malaysia Staff especially RIMC and Secretariat of R&D Seminar for providing us opportunity to share our findings and results in our works. Not to forget, thanks to IT Center staff forever lasting supports and last but not least, greatest thanks to my wife and my cute little son for their kind concern and support to deliver such contribution.

10.0 References

1. Lee,T.B., Fielding, R. and Masinter, L. (2005), "Syntax Components", RFC 3986, (section 3).
2. Lee,T.B., Fielding, R. and Masinter, L. (2005), "Generic Syntax", RFC 3986, (section 3.4).
3. Charles (2012), "Why is passing the session id as url parameter insecure?", retrieved date: 21 September 2012. from http://security.stackexchange.com/questions/14093/why-is-passing-the-session-id-as-url-parameter-insecure.
4. Cookie (2012), "HTTP cookie", retrieved date: 10 September 2012, from http://en.wikipedia.org/wiki/HTTP_cookie
5. Form (2012), "Form Web", retrieved date: 20 September 2012, from http://en.wikipedia.org/wiki/Form_%28web%29
6. Form_Method (2012), "HTML <form> method Attribute", retrieved date: 21 September 2012, from http://www.w3schools.com/tags/att_form_method.asp
7. QueryString (2012), "Query string", retrieved date: 21 September 2012, from http://en.wikipedia.org/wiki/Query_string
8. ASPQueryString (2012), "ASP QueryString Collection", retrieved date: 18 September 2012, from http://www.w3schools.com/asp/coll_querystring.asp
9. ASP_Session (2012), "ASP Session Object", retrieved date: 18 September 2012, from http://www.w3schools.com/asp/asp_sessions.asp

10. PHP_Sessions (2012), "PHP Sessions", retrieved date: 21 September 2012, from http://www.w3schools.com/php/php_sessions.asp

11. Barth, A., Berkeley, U.C. (2011), "Overview", HTTP State Management Mechanism, RFC 6265,Section 3, retrieved date: 22 September 2012, from http://tools.ietf.org/html/rfc6265#section-3

12. Linden, M. A. V. D. (2012), "Vulnerability Case Study: Cookie Tampering", Information System Security, Auerbach Publications © Copyright 2009, retrieved date: 21 September 2012, from http://www.infosectoday.com/Articles/Cookie_Tampering.htm

13. Hash_function (2012), "Cryptographic hash function", retrieved date: 19 September 2012, from http://en.wikipedia.org/wiki/Cryptographic_hash_function

14. MD (2005), "Merkle–Damgård construction", retrieved date: 20 September 2012, from http://en.wikipedia.org/wiki/Merkle%E2%80%93Damg%C3%A5rd_construction

15. HashStandard1 (1995), "*Secure Hash Standard"*, FIPS PUB 180-1, retrieved date: 24 September 2012, from http://www.itl.nist.gov/fipspubs/fip180-1.htm

16. HashStandard2 (2002), "*Secure Hash Standard"*, FIPS PUB 180-2, retrieved date: 24 September 2012, from http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

17. SHA-1(2012), "Comparison of SHA functions", retrieved date: 24 September 2012, from http://en.wikipedia.org/wiki/SHA-1#cite_note-4