Volume 2, Issue 6, June 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Elicitation and Structured Representation Modeling of Security Requirements for Authentication Vulnerabilities In Web Applications

R. Kumar
Department of Computer Science
Jamia Millia Islamia (Central University), New Delhi-110 025, India

Abstract: An understanding of the risks to which application will be exposed, can be a good starting point to design and implement secure web applications. Business oriented web applications need complex authentication policies to securely implement business processes. Threats against the confidentiality, availability and integrity of the data stored, processed and transmitted by application need to be matched against the policies, technologies and human factors that would protect them. The goal of this paper is to provide an insight into the secure development of web applications by exposing the pitfalls often encountered related to the authentication process and to security requirements that will ensure application is resilient to these attacks. In addition, a structured representation model for security requirements is also worked out for user friendliness.

Keywords: Security Requirements, Web Application Security, Security in Requirements Phase, Authentication Vulnerabilities.

I. INTRODUCTION

Attacks on web applications began appearing almost from the beginning of the World Wide Web, in the mid-1990s. Vulnerabilities in web applications have become the largest vector of enterprise security attacks. The number of Web application vulnerabilities continues to rise rapidly at a moderately steady rate of 3,000 to 4,000 disclosures per year [1]. Many organizations invested considerable resources to safeguard system security are unprotected against security attacks. Firewalls, network intrusion detection/prevention systems, anti-virus solutions, anti-spyware solutions, and Secure Sockets Layer are indeed critical security measures but alone are no longer sufficient. Security vulnerabilities, allow an attacker to access privileged data, delete critical data, and even break into the system and operate at the same priority level as the application, resulted to destroy the entire system. It is noteworthy to state that vulnerabilities are rarely used to elicit security requirement [2]. Securing the network, OS, and server but neglecting to secure the application is like building an elaborate fortress, but leaving its main gate open and unguarded. The industry's common response to securing applications has been to try to test security into the application at the end of the development process. But this approach fails to address the root cause of the problem: security, like quality, must be built into the application. Building security into an application involves designing and implementing the application in a way that reduces the risk of security attacks, then verifying that the policy is implemented and operating correctly. Essentially, security becomes a specification issue. Earlier, security used to be an afterthought, but now-a-days it is widely accepted to be an integral part of each of the phases of SDLC [3]. If the specification does not define how the application should be built to safeguard security, the application will be vulnerable to the types of sophisticated attacks that are starting to emerge now and will become increasingly prevalent in the future. The rest of the paper is organized as follows: In Section II, authentication vulnerabilities are discussed. In Section III, security requirements for the mitigation of authentication vulnerabilities are proposed. A structured representation model for security requirements is worked out in section IV. Tryout results are shown in Section V. Section VI presents Conclusion and Future work.

II. AUTHENTICATION VULNERABILITIES IN WEB APPLICATIONS

The *authentication* function enables the application to identify a request as originating from a known user, as opposed to being anonymous. Vulnerabilities are reported in well known databases/repositories [4, 5, 6, 7, 8]. Broken authentication vulnerabilities stand at third position under the OWASP top 10 application security risks-2010 [7]. Application functions related to authentication are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities [7]. Vulnerabilities related with weak authentication mechanism have been hereby collected, which may be termed as different subtypes of weak authentication.

- ***Insufficient Authentication*** permits an attacker to access sensitive content or functionality without proper authentication. When a website allows an attacker to

access the content and functionality without proper authentication, it is termed as insufficient authentication. It occurs when the software/web site doesn't prove or insufficiently prove that the claim made by an actor is correct [9]. These attackers' may attempt to steal the accounts information from others. Basically, they use the leaks or flows in the authentication or session management namely expose accounts, password, session id's to prove the false identity of users [10]. Hence, depending upon the web site resources, the web application should not be directly accessible to users without verifying the authentication.

Example: There are many web applications which have been designed with administrative functionality located directly off the root directory (/admin/). Although this directory is never linked from anywhere on the web site under normal conditions, but it can be accessed by using a standard web browser. The user/developer doesn't accept anyone to view this page because they assume that it is not linked. This results in not enforcing the authentication in many times. By the way, if an attacker visits this page, s/he can get the complete administrative access to the website.

- ***Brute Force*** attack automates a process of trial and error to guess a person's username, password, credit-card number or cryptographic key. In web applications some assets like identity, passwords, information, encryption keys, database lookup keys etc., are protected by a finite secret value. Brute Force attack is a method in which an attacker attempts to determine an unknown secret value to gain access to a protected asset by using an automated process usually trial-and-error [11]. Entropy of the value is smaller than perceived is the main fact about this attack. Log-in credentials brute forcing in web application is possible, as users usually select 'easy to memorize' words or phrases as passwords. Such an attack attempting to log-in to a system using a large list of words and phrases as potential passwords is often called a 'word list attack' or a 'dictionary attack'.

  Example: Shopping online with stolen credit cards usually requires information in addition to the credit card number, most often the CVV/SCS and/or expiration date. A fraudster may hold a stolen credit card number without the additional information. For example the CVV/CSC is not imprinted on the card or stored on the magnetic strip so it cannot be collected by mechanical or magnetic credit card swiping devices. In order to fill in the missing information the hacker can guess the missing information using a brute force technique, trying all possible values. Guessing CVV/CSC requires only 1000 or 10000 attempts as the number is only 3 or 4 digits, depending on the card type. Guessing an expiration date requires only several dozen attempts.
- ***Weak / insufficient Password Recovery Validation*** permits an attacker to illegally obtain, change or recover another user's password. Insufficient Password Recovery weakness occurs when a web site permits an attacker to illegally obtain, change or recover another user's password [12]. It is common for a web application to have a mechanism that provides a means for a user to gain access to their account in the event they forget their password. If a password recovery mechanism is weak, then it is more likely that it would be possible for a person other than the legitimate system user to gain access to that user's account. This weakness may be that the security question is too easy to guess or find an answer to (e.g. because it is too common). Or there might be an implementation weakness in the password recovery mechanism code that may for instance trick the system into e-mailing the new password to an e-mail account other than that of the user. There might be no throttling done on the rate of password resets so that a legitimate user can be denied service by an attacker if an attacker tries to recover their password in a rapid succession. The system may send the original password to the user rather than generating a new temporary password. In summary, password recovery functionality, if not carefully designed and implemented can often become the system's weakest link that can be misused in a way that would allow an attacker to gain unauthorized access to the system.

  Example: A web application using hints to help remind the user of their password can be attacked because the hint aids Brute Force attacks. A user may have fairly good password of "122277prince" with a corresponding password hint of "bday+fav author". An attacker can glean from this hint that the user's password is a combination of the user's birthday and the user's favorite author. This helps narrowing the dictionary Brute Force attack against the password significantly.
- ***Insufficient Anti-Automation*** The web application does not properly limit the number or frequency of interactions with an actor, such as the number of incoming requests. This can allow the actor to perform actions more frequently than expected [20]. The actor could be a human or an automated process such as a virus or bot. This could be used to cause a denial of service, compromise program logic (such as limiting humans to a single vote), or other consequences.

  Example: an authentication routine might not limit the number of times an attacker can guess a password. Or, a web site might conduct a poll but only expect humans to vote a maximum of once a day.

III. PROPOSED AUTHENTICATION SECURITY REQUIREMENTS

Security requirements may be proposed for the mitigation of authentication related security vulnerabilities, which are accepted by the majority of the well known vulnerability databases available in the literature [7, 13, 14]. An authentication is the process to determine how the application validates the identity of the user. The most serious security

risk associated with the authentication process is that of subversion. If the login mechanism fails to properly identify the user and permits access to the application, the rest of the security mechanisms become irrelevant. Authentication is also rife with information leakage, increasing the risk of unauthorized access to user data. Poor techniques in managing user registration, account lockouts and recovery lead to a number of the application layer 'denial of service attacks'. Identity Protection and Validation Infrastructure should use the combination of username, password, as authentication [15, 16]. To avoid pitfalls which may be resulted in vulnerabilities in countless web applications, password reset functionality must be implemented in a safe and secure manner by incorporating a series of best practices [17]. Web application authentication (with SSL certificate) helps consumers verify that they are on the correct Web site, and not an imposter or phishing site [18]. Risk-based authentication (Fraud Detection Service) learns how each user behaves and requires additional authentication when a fraud risk is detected [19]. Nemesis prevents authentication vulnerabilities, using authentication information of the safely authenticated users by automatically inferring the same [10]. These may be inclusive and not mutually exclusive of other recommendations. To overcome weak authentication related vulnerabilities, following security requirements are hereby proposed in Table I:

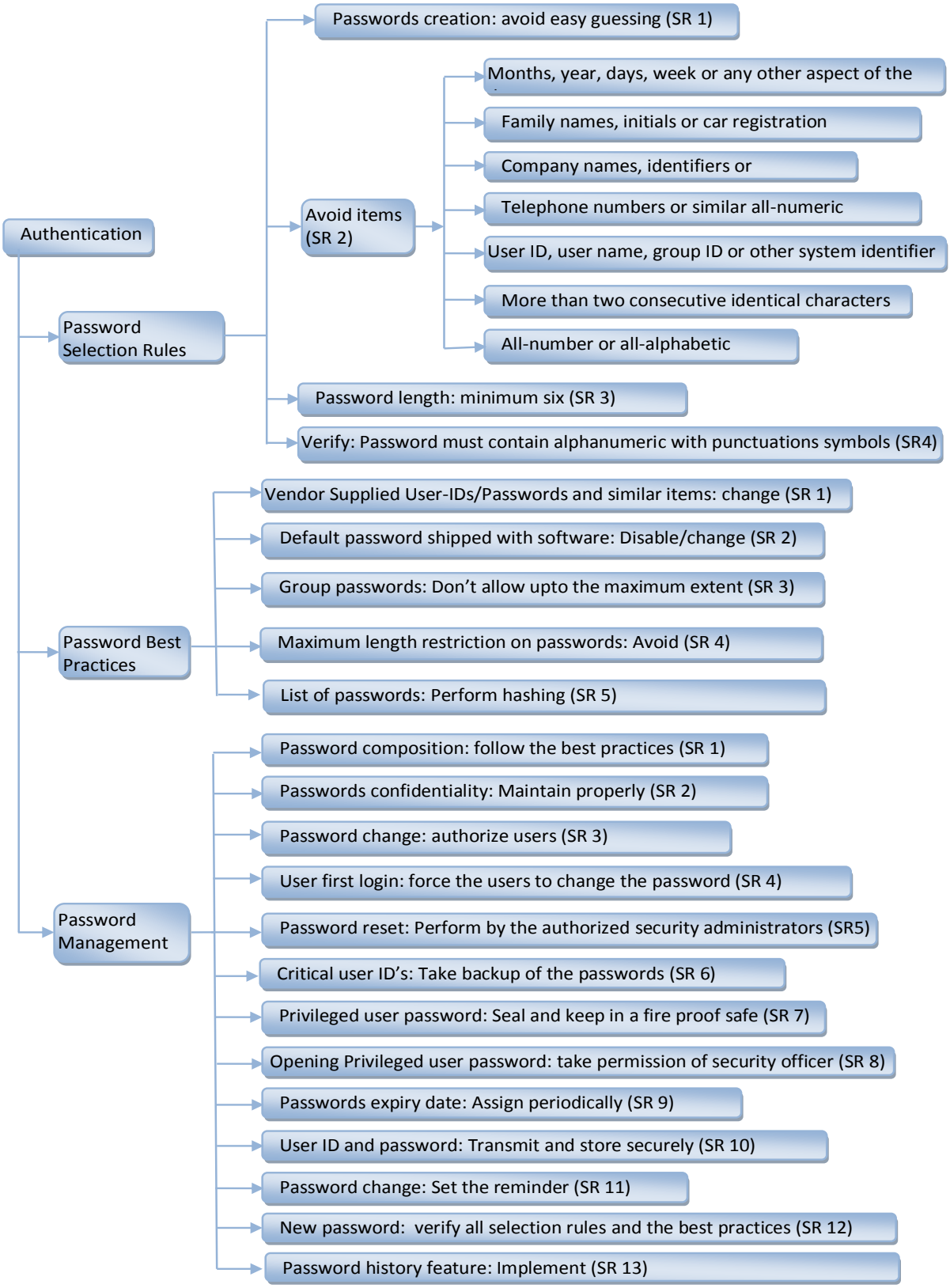
Table I: Authentication Security Requirements

Minimal	*SR 1:*	*Protect identity credentials treating it as highly sensitive information and reject the credentials that have expired/no trusted/revoked by the issuing organization.*
	SR 2:	*Ensure that these credentials are not stored by the browser.*
	SR 3:	*Always employ a secure method to recover identity credentials.*
	SR 4:	*Enforce the best practices for password management specified by applicable standards.*
	SR 5:	*Assure forget to include protection against automated brute force attack namely dictionary attacks.*
	SR 6:	*Generate an alarming signal for immediate follow up, if more than one account is locked out from the same IP address in a short time (e.g., five minutes).*
	SR 7:	*Create a facility as watchdog for three consecutive incorrect password attempt and display the message 'login failed'.*
	SR 8:	*Create a session key to authenticate subsequent data transmissions in the session.*
	SR 9:	*Ensure that minimum error information is returned in the event of authentication failure.*
	SR 10:	*Adopts a policy of using least-privileged accounts.*
	SR 11:	*Ensure locking of the account after a predefined number of failed attempts.*
	SR 12:	*Ensure to implement account locking mechanism in every module of the application requiring the user to authenticate.*
Clients/organizations	*SR 1:*	*Establish multitier authentication mechanisms, preferably two tier, to resist session hijacking and other common internet attacks for highly sensitive information.*
	SR 2:	*Ascertain strong one-tier authentication for medium and low level sensitive information.*
	SR 3:	*Ensure to implement a matrix for user rights structure.*
Administrators	*SR 1:*	*Authenticate the identity of the administrator by using credentials before gaining access to her account.*
	SR 2:	*Establish a process/technology to verify that a secure computer is being used by the administrators.*
	SR 3:	*Verify that the proposed authentication mechanism being followed in the organization.*
	SR 4:	*Take appropriate action in case in any violation.*
Users	*SR 1:*	*Sign an agreement for setting users responsibilities.*

	SR 2:	*Emphasize the risk involved with the use of insecure computers, periodically among the users.*
	SR 3:	*Enforce users to apply the best practices for strong authentication practically.*
	SR 4:	*Users' ids must not be shared until not necessary.*
Password Selection Rules	*SR 1:*	*Encourage the users to create the passwords to avoid easy guessing.*
	SR 2:	*Avoid use of any familiar item like date, family names, company names, telephone number, user ID, user name, group ID or other system identifier for password*
	SR 3:	*Ensure that minimum length of password is eight.*
	SR 4:	*Ascertain that the password is alphanumeric with punctuations symbols.*
Password Best Practices	*SR 1:*	*Always change Vendor Supplied User-IDs/Passwords, encryption keys, and other access codes included with vendor-supplied systems.*
	SR 2:	*Disable/change default password shipped with software.*
	SR 3:	*Don't allow group passwords upto the maximum extent.*
	SR 4:	*Avoid enforcing a maximum length restriction on passwords.*
	SR 5:	*Any list of passwords should be hashed (one-way hash) to avoid the chance of reading authentication information by an attacker.*
Password Management	*SR 1:*	*Ensure that password composition is proper as per the best practices.*
	SR 2:	*Maintain proper confidentiality of passwords.*
	SR 3:	*Ascertain that capabilities are provided to change the passwords by users.*
	SR 4:	*Configure the system to force the users to change the password immediately after the first login.*
	SR 5:	*Ensure that resetting of user password is performed by the security administrators after verification of identity.*
	SR 6:	*Set up a process for backup of passwords for all the critical user ID's.*
	SR 7:	*Seal all the privileged user passwords and keep in a fire proof safe.*
	SR 8:	*Open password envelops only with the written permission of security officer but don't forget to change the password immediately and keep in a new sealed envelope.*
	SR 9:	*Set an expiry date of passwords, with already decided periodicity.*
	SR 10:	*Transmit and store each user ID and password in a secure manner.*
	SR 11:	*Set the reminder to change the password at least before 3days of expiry.*
	SR 12:	*Verify that for the new password, all the selection rules, best practices are followed.*
	SR 13:	*Ensure to implement a password history feature.*

IV. STRUCTURED REPRESENTATION MODEL OF SECURITY REQUIREMENTS

Security requirements presented in the aforementioned tables have been placed in the order of implementation for the mitigation of vulnerabilities belonging to authentication vulnerabilities family. A structured representation model, using the diagrammatic representation is worked out for user friendliness, is described in order as follows:



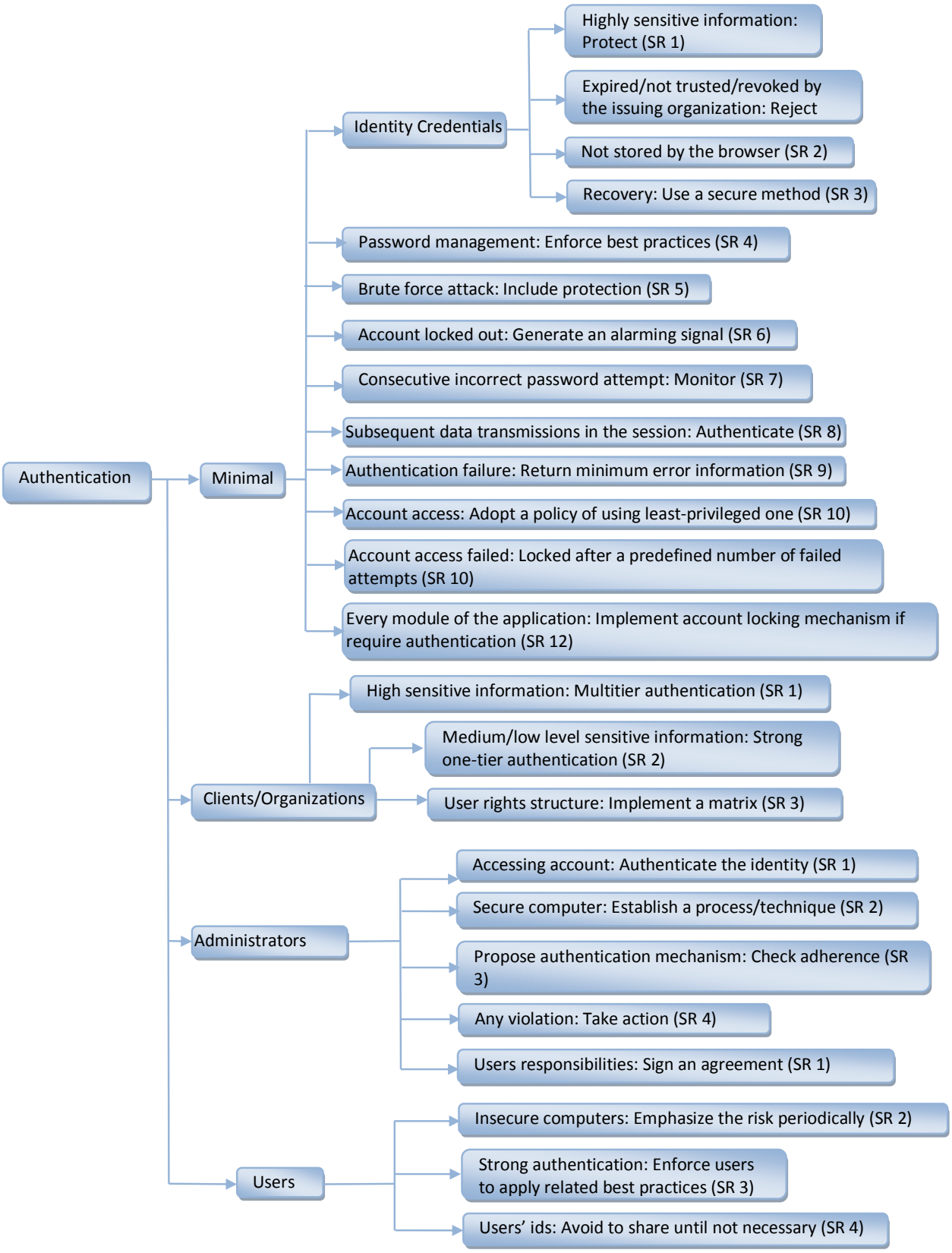


Fig. 1: Structured Representation Model of Authentication SRs

V. TRYOUT RESULTS

The proposed security requirements have been validated by using SRS of one live project named Personal Investment Management System (PIMS). This web application is intended to help the user to keep account of his/her money invested in institutions such as Banks and Share Market. Following are the features in the scope of the web application:

- Managing investment of a single user, which would include maintaining bookkeeping information about entities like Portfolio, Security, and Transaction.
- Computation of Net-Worth and Rate of Investment (ROI) of the Investor.
- Giving alerts to the user, if he requests for one.
- Downloading the current prices of shares from the web.
- User authentication.
- Features for actual purchasing and selling of securities. That is, actually buying and selling of shares/securities is done outside PIMS.
- Tax computations for gains/losses.
- Any market related prediction.

Security for overall application is highly needed.

After the study of SRS (complete and ready to submit for designing), researcher found that some of the security requirements have already been taken care of. But there are a number of remaining security requirements, to be incorporated before proceeding to the design phase, as given in the following Table II:

Table II: Proposed Security Requirements (SRs) and Compliance Status

Proposed Security Requirements (SRs)	**Compliance Status (Y/N)**	
	Y	**N**
Minimal: Incorporate SR1 to SR12.	SR7, SR11	SR1 to SR6, SR8, SR9, SR10, SR12
Clients/organizations: Incorporate SR1 to SR3.	SR2	SR1, SR3
Administrators: Incorporate SR1 to SR4.	SR1 to SR2	SR3, SR4
Users: Incorporate SR1 to SR4.	SR1	SR2, SR3, SR4
Password Selection Rules: Incorporate SR1 to SR4.	Y	
Password Best Practices: Incorporate SR1 to SR5.	SR1, SR2, SR3	SR4, SR5
Password Management: Incorporate SR1 to SR13.	SR1, SR2, SR4, SR9, SR10	SR3, SR5 to SR8, SR11, SR12, SR13

To evade the authentication vulnerabilities, wherever security requirements were not in place, the same may be adapted. By incorporating all these SRs, SRS of the proposed application may be strengthened with reference to security.

VI. CONCLUSION AND FUTURE WORK

Keeping in view the daily increase of web application vulnerabilities, developing secure applications has become a necessity. Developing security requirements along with requirement phase of SDLC and following them in later phases may help to produce a more secure application. This paper proposes security requirements for authentication process based on the vulnerability analysis related to the same. A structured representation model, using the diagrammatic representation is also worked out for user friendliness. It appears to be an evolving process as new vulnerabilities and their corresponding security requirements shall be identified. Therefore, extension/modification and proposal of new security requirements may also be done. This work may provide guidance to the industry as well as academia for developing more secure web applications.

REFERENCES

[1] IBM Security Solutions. (2010, August). IBM X-Force® 2010 Mid-Year Trend and Risk Report.

[2] Golnaz Elahi, Eric Yu, and Nicola Zannone. "A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities," Journal of *Requir. Eng.* 15, 1 (March 2010), 41-62. DOI=10.1007/s00766-009-0090-z http://dx.doi.org/10.1007/s00766-009-0090-z

[3] Steve Lipner, "The Trustworthy Computing Security Development Lifecycle," proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), p.2-13, December 06-10

[4] IBM Security Solutions, IBM X-Force® 2010 Mid-Year Trend and Risk Report, August 2010

[5] Nancy R. Mead, Gary Mcgraw, "A Portal for Software Security", Published By The IEEE Computer Society, IEEE Security & Privacy, 2005.

[6] WASC threat classification version 2.00. (2010). Retrieved January 10, 2010, from web application security consortium. http://www.webappsec.org

[7] Open Web Application Security Project. (2010) The ten most critical Web application Security vulnerabilities. Retrieved March 10, 2010, from http://umn.dl.sourceforge.net/sourceforge/owasp/OWASPTopTen2010.pdf, 2010.

[8] CWE/SANS Top 25 Most Dangerous Programming errors and Common Weakness Enumeration (CWE). Retrieved April 16, 2010, from http://cwe.mitre.org/top25

[9] Security focus (2008). BilboBlog admin/index.php Authentication Bypass Vulnerability. Retrieved December 2, 2009, from http://www.securityfocus.com/bid/30225.

[10] Dalton, M., Zeldovich, N., and Kozyrakis, C. (2009). "Nemesis: Preventing authentication & access control vulnerabilities in web applications," In Proceedings of the *18th Annual USENIX Security Symposium*.

[11] Endler, D. (2001) Brute-force exploitation of web application session IDs. iDEFENSE, 2001. Retrieved March 02, 2008, from http://www.cgisecurity.com/lib/SessionIDs.pdf

[12] Insufficient Password Recovery. Retrieved January 05, 2008, from http://projects.webappsec.org/Insufficient-Password-Recovery

[13] National Vulnerability Database. Retrieved February 21, 2009, from http://nvd.nist.gov/

[14] Common Weakness Enumeration. Retrieved March 21, 2010, from http://cwe.mitre.org/

[15] VeriSign,VeriSignIdentityProtection(VIP). (2010). Network, Retrieved Feb 14, 2010, From <http://www.verisign.com/authentication/consumerauthentication/shared-authentication-network/index.html>

[16] Best Practices for Your "Forgot Password" Feature. Retrieved July 28, 2010, from http://www.fishnetsecurity.com/sites/com.fishnetsecurity/downloads/Forgot_Password_Best_Practices_v2.0.pdf

[17] Shpritz, A., D. (2010, July). Securing Self-Service Password Reset Functionality In Web Applications Retrieved November 24, 2010, from http://www.securabit.com/wp-content/uploads/2010/08/self-service-password-reset_v5-1.pdf

[18] VeriSign, (2010), Secure Socket Layer (SSL): How It Works,VeriSign, Retrieved Feb 14, 2010, from <http://www.verisign.com/ssl/ssl-information-center/how-sslsecurity-works/ index.html>

[19] Bo, Li. & Congwei, X. "E-commerce Security Risk Analysis and Management Strategies of Commercial Banks." *International Forum on Information Technology and Applications*, (1). (2009) 423-424.

[20] Pramod, D., "A Study of Various Approaches to Assess and Provide Web based Application Security", International Journal of Innovation, Management and Technology, 2(1), (2011), 58-62.

