Analysis of Web based Structural Security Patterns by Employing Ten Security Principles

Rabia Riaz †, Sanam Shahla Rizvi ††, Farina Riaz †††, Nosheen Hameed †, Sana Shokat †

† Department of CS & IT, University of Azad Jammu and Kashmir, Muzaffarabad, 13100, Pakistan
†† Department of Computer Sciences, Preston University, 15, Banglore Town, Shahrah-e-Faisal, Karachi, 75350, Pakistan
††† Independent Researcher

Summary
Security is an important and reminisce issue of any software. To ignore security matters or leaving them till later stages of software development could be dangerous as it is difficult to retrofit security in an application later on. In the security critical applications, it is extremely important to avoid mistakes. Therefore, the use of security patterns is important for developing a secure system. In this paper we present how security can boost up by using ten security principles. We conducted a literature review in the field of security patterns, identified problems and proposed a pattern for user authentication function in mobile devices and carried out a comparison based research. We are using ten security design principles as matrices comparing with structure patterns. We summarize which patterns fulfill which of these ten security principles. We get these security patterns from security patterns repository.

Key words:
Web applications; security design patterns; security principles; mobile devices; user authentication

1. Introduction

Patterns approach was invented by Christopher Alexander. He defined a pattern as "a problem which occurs over and over again in our environment and that describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" [1-2]. Alexander was a buildings architect, who had created the idea of a design pattern in the course of his design work. He realized that in buildings design, there are certain well-defined components that occur repeatedly and which can be described in design terms, and reused. He defined these repeating components as design patterns, and each time the problems recurred he reused the same design pattern to provide the solution. This technique maintained accuracy and consistency in design for the common components in his buildings designs. Alexander's patterns ranged from high level to low level [3].

A security pattern is a well-understood solution to recurring information from security problem. They are patterns, in the sense originally defined by Christopher Alexander, applied to the domain of information security. Security patterns are intended to capture security expertise in the form of worked solutions to recurring problems. These patterns capture the strengths and weaknesses of different approaches in order to allow developers to make informed trade-off decisions between security and other goals. Security patterns instead try to provide constructive assistance in the form of worked solutions and the guidance to apply them properly [4-5].

A security pattern describes a particular recurring security problem that arises in security pattern specific contexts, and presents a well-proven generic solution for it. The solution consists of a set of interacting roles that can be arranged into multiple concrete design structures, as well as a process to create one particular such structure [1]. Due to various communication features security considerations are greater interest these days. Security patterns encapsulate security expertise in the form of proven solution to recurring security problems. Security patterns are used and understood by developers who are not security expert. Security patterns work like a bridge used to reduce gaps between developers and security experts [6]. A pattern normally comprises of different important parts. Table 1 describes essential parts of a pattern.

Table 1: Parts of a Pattern

Name of part	***Description***
Name	A standard and comprehensive name by which the pattern can be discussed.
Problem	A concise summary of the problem addressed by the pattern
Consequences	The impact that the pattern has on the "essential forces" at play
Solution	A description of the solution

To improve development of secure software, Viega and McGraw pointed out ten following guiding principles to achieve better security and to manage unknown attacks [7].

- Secure the weakest link
- Practice defense in depth

Manuscript received October 5, 2017
Manuscript revised October 20, 2017

- Fail securely
- Principle of least privilege
- Compartmentalize
- Keep it simple
- Promote privacy
- Hiding secrets is hard
- Be reluctant to trust
- Use your community resources

The aim of this study is to explore how security of a system can be increased by using ten security principles, and how many of security principle are satisfied by these security patterns. We highlight the objectives of this research as below:

- Identify a set of security patterns which can be used to address the issues in web-applications.
- How effectively these security patterns satisfy the security principles?
- What are the benefits and liabilities of implementing security pattern in web-application?
- Propose a user authentication pattern for Smartphone devices.
- Identify set of patterns that can be used together to increase system security.

In this paper, our work is organized in the following sequence. Section 2 gives an overview of existing work that has been done by using security patterns. Section 3 evaluates different structural security patterns from web domain that we use in a web applications when dealing with security issues with supporting security principles. In Section 4, we identify a new security pattern as user authentication for mobile devises and also compared this pattern against supporting security principles. In Section 5, we highlight ten security design principles used as matrix comparing with structure patterns. We have summarized which pattern fulfills which of these ten security principles. Finally, paper is summarized and recommendations for future are presented in Section 6.

2. Related Work

Patterns are newly introduced technique and now this technique is extensively being used to resolve general problems so that the suggested solution can be reused many times. Pattern study has been made in different areas; security is one of them [8]. The first research paper about software cryptography is [9] that later supported and enhanced towards security patterns [10-11]. Different types of templates have been used for patterns [6]. In [12] authors show how security patterns help to secure the system by using UML. A security pattern describes a particular recurring security problem that arises in specific contexts and presents a well-proven generic scheme for its solution [13]. A security pattern system is a collection of security patterns, together with guidelines for their implementation, combination and practical use in security engineering. Security has become key issue in current systems. Better security can be achieved by using security patterns in all phases of software development.

It is recommended to consider security requirement from the beginning of software development life cycle, avoid the expenses of rework and minimize security weaknesses [12]. It is not an easy job because not all the software engineers are security specialist. Therefore, research presents the guidelines on “how to apply” and “Where apply” appropriate pattern [13]. Table 2 presents the detail of related work on security patterns.

3. Evaluation of Structural Security Patterns with Supporting Security Principles

This section explains security patterns in detail and at the end of each security pattern a comparison of security patterns with ten security principles is conducted to check either those patterns are fulfilling the security principles or not.

3.1 Account Lockout

Problem

Passwords are the only approach to user authentication that has gained global user acceptance. Sometimes user chooses weak passwords that are easily guessed. Password guessing tools are very effective to find out poorly chosen passwords. Many operating systems implement login delays. It does also prove helpful to slow down the process of guessing attacks. But in the web environment an attacker is completely unknown during guessing tens of thousands of passwords per hour.

Solution

The pattern account lockout is used against password guessing attacks. In this pattern server maintains a threshold value for incorrect password attempts for each and every user. When a user inserts wrong password the count is incremented, and when user logins successfully account is cleared. When illegal attempts reach to predefine threshold value account gets locked. When account gets locked and user tries to gain access to locked account, system records that request but never processes and user is unaware of this process, that the request is rejected. In account lockout pattern account get locked automatically after some time of inactivity.

Table 2: Related work on security patterns

Citation	Article Title	Year	Explanation
Shahnawaz Alam [14]	Using security patterns in Web – Application	2013	How security patterns efficiently handle web security problems
Yoder & Baralow [8]	Architectural Patterns for enabling application Security	1997	Set of seven patterns present security model to build secure application
Rubira ,et al [9]	A pattern language for cryptographic software	1998	Describe the basic purpose of cryptography &pattern language
Cuevas, C [15]	Security Pattern for capturing encryption based access control to sensor data.	2007	Nine patterns offer four basic purposes for cryptographic software
Fernandez et al. [16]	Semantic analysis pattern	2000	Using SAPs, a methodology is developed to build the conceptual model in a systematic way.
Coggeshall [17]	Security Design Session	2001	Eight patterns present guidelines
Dan, W. [18]	Security Functional Requirements Analysis for Developing Secure Software.	2007	Description of different security patterns and comparison
Steel et al. [19]	Core security patterns, Best practices and strategies	2005	Guide to building robust end-to-end security

Supported Security Principles of Account Lockout Pattern

I. Secure the weakest link

This pattern protects account from password guessing attacks and secure weakest link. Account lockout pattern protects an account by implementing lockout mechanism and stops unauthorized access. If an attacker knows the actual password of the system then this pattern not plays a vital role, and attacker gains access to the system.

II. Practice defense in depth

Multiple invalid attempts from users to start the invalid transactions get rejected and make it more difficult by adding of best practice defense such as account lockout. Account lockout increases liability by helping and ensures that user accounts will not be compromised by using a password-guessing attack. This pattern does not reveal that account has been lockout and this also ensures that an attacker cannot know whether a password guessing attacks are simply being dropped. It also prevents the attacker from learning how many attempts are required in order to lockout an account.

III. Be reluctant to trust

Account lockout does not extend trust unnecessarily only authorized user have right to gain access to the account. The good point of this pattern is to keep informed the user about the login attempts. When a user successfully logins, it informs the user about number of failed login attempts since the last successful login. A user who mistypes his password will be able to identify that the invalid attempts were legal. But on the other hand if account is under threat user will be alert. Then user will inform the administrator about the problem. It's also important to inform user about the last successful login.

IV. Principle of least privilege

Account lockout pattern fulfills this principle by allowing only the minimum access, necessary to perform an operation. This pattern fulfills the principle of least privilege by giving access only authorized person, and by defining a predefined threshold value. It allows a predefined number of fail login attempts, and when it reaches to that threshold value, account gets locked.

It grants access to account only authorized user. This pattern mostly allows three incorrect attempts before the account is locked out and requires manual intervention by a customer service representative. Only authorized subjects may assign privileges to use an account.

V. Promote privacy

Privacy of user data and account will be increased by additional protection of the account. Reliability of individual accounts has been improved and this could indirectly improve the integrity of the site. This pattern supports the principle of least privilege and practice defense in depth. So the privacy of the account is automatically improved. Now a day, all of the on-line banking systems provide an account lockout mechanism to protect critical user data. When an attacker tries to gain access to the account by using password guessing attacks, and crosses the limit of invalid attempts the account automatically gets locked. This pattern doesn't disclose that account has been locked out and an attacker is unknown of this mechanism that account lockout is actually protecting user account.

VI. Hiding secrets is hard

Security of an application also depends on some secrets. If any application reveals these security secrets then securities can easily be exploited. This pattern fulfills this principle by not revealing to attacker that account has been locked.

3.2 Authentication Session

Problem

Now mostly web applications need some sort of session model, in which request of multiple pages is combined into an interesting skill. However, great awareness is needed to use session semantic, in trusted way [20]. Session mechanisms are enough for tracking noncritical data and apply safe transaction. But there are chances to make mistakes when applying session mechanism in the condition where accountability, integrity, and privacy are crucial.

Solution

In this pattern, server keeps the authenticated user identity and the time of the last request. With every protected page there is a standard header executes on the server, used to check the authentication associated with the session. When the user first time requests for a protected page, then server checks the authentication check and notes that no authenticated identity is present in the session information. Then the server traces the original request and forwards the user to a login page. When the server receives and verifies a user's login record (typically a username and password), it forward the user back to the originally requested page [20].

On all subsequent page requests, the server checks the authenticated identity without requiring that the user re-authenticate him or herself. There are two ways to end an authenticated session. First the user can openly invoke a logout page, which causes the store credentials to be flushed. Second if the session remains inactive for predefined time period the session will automatically end and user needs to re-authenticate for first subsequent page. This pattern stores user authenticated data on server. So the application can be very confident that user has not tempered with it.

Supported Security Principles of Authentication Session Pattern

I. Promote privacy

Many web banking and e-commerce applications depend on this pattern. Any site that require user authentication and does not want to store information on client can use this pattern. Because it is dangerous to re-enter credit card numbers on every page, the authenticated session pattern stores the user's authenticated identity on the server so the application can be more secure and confidential. This pattern maintains session data on server as part of session object. To associate session object with client the application server assign unique random identifier to each session object. This pattern enhances accountability by providing a secure approach to repeated authentication.

II. Compartmentalize

This pattern compartmentalizes the session security policy within a single component, so that changes to the policy like affecting usability, accountability, and performance does not impact the client or other parts of the application.

III. Practice defense in depth

The authentication session store user's authenticated information on the server that makes it more reliable and confident. No one can temper with it. Application server grants unique, random identifier to each session object. This session identifier is then given to the client, and the client presents it on each subsequent page request. Session identifier must be hard to guess. These are provided by the application server and cannot be modified by custom applications.

IV. Principle of least privilege

This pattern grants access after providing valid session identifier. When a user submits an invalid session identifier, the application server does not process that application. Large numbers of invalid session identifiers are a clear indicator that something is going wrong with the application. The client requests a protected page from the server, passing the session identifier, then session mechanism call up protected page. Authentication checkpoint checks that authenticated identity field, if it is empty then it returns to login screen for correct information otherwise grants requested page.

V. Be reluctant to trust

This pattern does not extant the trust and increases the integrity of the data by using secure approach and applying repeated authentication. Data stores on the server not on the client, which is very secure for confidential transactions. For example, this pattern is used by different online banking systems due to its strong security.

VI. Hiding secrets is hard

This pattern assigns unique random identifier which is hard to guess and it protects user session. When a user submits an invalid identifier, the application server should notify the application about that event.

3.3 Client Input Filters

Problem

In web environment client and server are necessary. Sometimes client executes on hardware which is un-trusted and web developer has no control on that. Developers believe that application will work and behave as they programmed. But often attackers temper with client that causes applications to behave in unsecure manner e.g. java scripts functions are used by many sites for data validation. But attackers easily copy the source page, change them and then execute new client code. Some sites use different security measures on client side like check password and use account lockout but some time attackers cross these restrictions.

Solution

The solution of this problem is to re-compute all the data on the server provided by the client. Checks used for the data validity, should be reperformed on the server. It is very important that sensitive data should be kept in encrypted form. During re-computing server should take great care. In case of any sign of bogus data transferring, it simply should reject such type of requests. The pattern client input filter is capable of changing request before sending to the required object. In this pattern if any data is not according to the context, it should just got rejected the request. In this pattern central logging mechanism is used to account all the filtering actions and if it detects any suspicious request it rejects and reports.

Supported Security Principles of Client Input Filter

I. Practice defense in depth

Data store on the client always have risks of data tempering, an attacker can modify the client code so the application behaves in un-trusted way. Basic purpose of the client input filter pattern is, it takes all the transaction from client suspected and filters it on server. Different types of data validity checks are performed on the client repeated on the server before the processing of data. This pattern is able to filter all the data before transferring to server or intended object. If any data shows the sign of tempering and not easily got fixed, the client filter drops the request. Central logging mechanism is used for reporting all filtering events.

II. Promote privacy

This pattern increases the privacy of data deal by a web site, and increases the reliability. Client input filters protect the application from data tampering performed on un-trusted clients. Sensitive data that must be stored on the client should be kept in an encrypted, tamper-proof form.

III. Be reluctant to trust

This pattern does not extent trust unnecessary, by rechecking all the data again on the server. Mostly long URLs are dropped. Client filters should be able to modify requests before delivering them to the intended object. If the data cannot easily be fixed, the client filter should reject or simply drop the request. All filtering events should be reported to the central logging mechanism.

3.4 Encrypted Storage

Problem

Large number of user sensitive information is stored on the web applications like social security numbers, credit card numbers, passwords etc. To protect such critical information every security effort can be used. In past, due to the loss of sensitive information it was very difficult for companies to recover their publicity back. So it is recommended that don't store sensitive data. But sometimes it is necessary to store data.

Solution

This pattern encrypts the sensitive user data before transferring it to disk. Before using, it has to decrypt in memory. So if an attacker steals the data, he is not able to get access to critical data. In this pattern single key is created by the application server and uses to encrypt and decrypt user information. Under this solution, the application server maintains a single key that is used to encrypt and decrypt all critical user data. The most important thing is that key should be saved in secure way, and it should be changed timely.

Supported Security Principles of Encrypted Storage

I. Practice defense in depth

Combination with other security techniques, this pattern practices defense in depth. This pattern supports this principle by providing security in layered form. This pattern encrypts the data before transferring to any disk. Data decrypt in the memory when needed. User confidential data is protected by using single key. This key is used to encrypt and decrypt the data, and the key is changed time to time. When the key is changed the old key makes it more authenticated and confidential. Different types of COTS (commercial off-the-shelf) devices can be used to protect the key

II. Fail securely

Pattern supports the principle of fail securely. There are huge cases of hackers stealing records containing sensitive user's data. The encrypted storage pattern provides a second line of defense against the theft of data on system servers. Although server data is typically protected by a firewall and other server defenses, there are numerous publicized examples of hackers stealing databases containing sensitive user information. The encrypted storage pattern ensures that even if it is stolen, the most sensitive data will remain safe because data is in encrypted.

III. Promote privacy

This pattern increases privacy by ensuring that user data cannot be decrypted, even if it has been stolen. Different sites use encryption to protect the most confidential data to be stored on the server.

IV. Be reluctant to trust

This pattern minimizes the trust and enhances the security. In this approach user data is protected by using single key. Server also protects a single key that is used to decrypt and encrypt user's sensitive data and the key is changed time to time. When the key is changed it must require the old key which makes it more authenticated and confidential. Key is also saving in secure way.

V. Hiding secrets is hard

The encrypted storage pattern encrypts the most critical data before transferring. Each user data is protected by using single key. Application server maintains a single key to encrypt and decrypt all critical user data.

3.5 Minefield

Problem

We make lot of efforts to protect our system but this possibility always exists that any server which is remotely accessible has a chance of attack. If an attacker knows the function of the system, it is easy for attacker to go through system rather than to carry out the long procedure. Use of COTs devices is expensive [21].

Solution

It can be used to incompatible the attacker with tools. It alerts the administrator about the existence of an attacker. We can rename common, critical commands on the server and replace alerts, when an attacker uses that command it awakes the operator about the presence of attacker.

Supported Security Principles of Mine Field Pattern

I. Secure weakest link

To prevent attacking is impossible but using some techniques it is possible to protect the system to some extent. An attacker is always looking for parts that are easy to break, and tries to enter through these weak links. This pattern can be implemented to secure the weakest link. Pattern allows the modifications that have different effects. It breaks the compatibilities with existing attack tools. It alerts the operator about the presence of an attacker and then countermeasures can be adopted and stops further processing. We can rename different commands such a way, when an attacker tries to enter or type the command it shuts down the system or alert the operator about the presence of the attacker.

II. Defense in depth

If security is provided in form of security layers, it better protects the system. This pattern provides security in layered form. This pattern commences customization that detects an attacker and counters this advantage. By adding different booby traps, it informs operator about the presence of an attacker. Different deception toolkits can be used like to change the organization of a file system, to rename all the administrative commands used on server and to rename them with tools that alert the administrator about the suspicious activity initiating by an attacker. Collectively use of these techniques provides higher assurance that an attacker will not be able to attack the server.

III. Fail securely

This pattern can be implemented in fail securely mannered. It helps to detect if an attacker tries to access the system. The system uses different techniques and traps to make uncomfortable the attacker with applications to aim with that he stops attacking.

IV. Least privilege

When we rename the files and change the location of the files then only the authorized person knows about these modifications and only authorized person have access to those files. So privileges are given only to authorized person by following the principle of least privilege.

V. Be reluctant to trust

This pattern fulfills this principle by adding different traps that don't allow easy access to the system, and not trusting on everybody. This pattern totally neglects the trust on suspicious behavior.

3.6 Network Address Blacklist

Problem

Weaknesses in the software had raised the amount of web theft. Now operating systems are strong enough that the hacker community shifts to the other side, now their targets are application layer weaknesses. Unknown access is achievable in web environment so the chance of attacking exists there, and it's not easy to stop them. When preventions fail to defend against hacking, there should be some other ways to defend the site. Lockout pattern helps to defend the individual account but not helpful to protect the web site.

Solution

Network address blacklist keeps track of network addresses that show improper behavior or suspicious activities, and if they originate any request it normally dropped. In some cases application developers have no rights to blacklist system, in this case human operator works as a blacklist. In the manual way when server alarms, the operator selects the manual safety measures like fire walls.

Supported Security Principles of Network Address Blacklist

I. Promote privacy

This pattern definitely improves the privacy. The privacy depends on how efficiently we implement this pattern and what sort of security measures we use. If operator has complained of misusing then it contacts to ISP. If ISP fails to end this activity then operator has the right to block this complaint for further activity through firewall.

II. Practice defense in depth

This pattern blocks the IP address that causes some danger or trying to illegal access. This pattern uses different intrusion detection systems that help to enhance the security.

III. Compartmentalize

This pattern isolates the suspicious activity and blocks that activity. Any IP address that is involved in suspicious activity should be blocked. This feature protects the system.

IV. Be reluctant to trust

A network address blacklist mechanism maintains a list of network addresses that have exhibited in any inappropriate behavior. When a request is received from a blacklisted address, it simply is dropped on the floor.

3.7 Password Propagation

Problem

The problem addressed by this pattern is that mostly web applications create single database account for all users. Hardcoded password is assigned to database from application server. And if that account is compromised then data of every user is at risk. Mostly databases are not able to hold up huge amount of user account [22].

Solution

This pattern divides the application into two parts, front-end and back-end. Front-end is used to communicate with user and presenting data, back-end is used for transaction process and as well as for maintaining single user account. In this mechanism front-end does not have the rite of direct entry to the back-end. User authentication is revalidating on back-end. If the front-end has hijacked the attacker have no right of entry or change to user data because back-end also requires password, and if password is not valid back-end rejects that request.

Supported Security Principles of Password Propagation

I. Secure the weakest link

Mostly web applications use single database account to save and supervise all user data. If this site is hijacked, the attacker might have complete right of entry to every user data. This pattern requires that any user's authentication information must be verified by the database before granting access to that user's data.

This pattern secures the weakest link by defining duel verification process. In this pattern every application is broken into two parts a front-end and back-end. The front-end is used for communicating with user and shows data in a formatted way. The purpose of back-end is executing the transaction process and supervises individual user account data. So by using this security pattern we can secure the weakest link.

This pattern also appears in web front-ends to existing banking systems. Most banking systems already use PINs to authenticate users. But PINs are not strong enough to withstand automate guessing attacks. Therefore, a web enabled banking application uses a password to authenticate the user to the application, and requires the user to provide a PIN in order to authenticate the user to the traditional banking system

II. Practice defense in depth

This pattern fulfills this principle by adding the security in different layers. Pattern enforces the user to authenticate his/her self before getting access to user's data. When front-end verifies a user's password stored on server as part of the session data, it starts a transaction by passing the user information to the back-end. When back-end verifies the user's password then it grants access to the user data. It's not possible for front-end to have direct access to user's data but it has to go through the back-end.

III. Fail securely

Password propagation pattern can be implemented as fail securely manner if the front-end is under attack or compromised. An attacker cannot be able to access and change the user account because every request also is verified on the back-end for further transaction. If the front-end is totally under attack only the passwords of the users are at dangers that are currently logged in.

IV. Principle of least privilege and Compartmentalize

This pattern assigns privileges only to the authorized persons. Only a user verified both by front-end and back-end gets granted access to user account. Access is restricted only to authenticated person. This pattern is an example of partitioned application. In this pattern back-end stands for trusted proxy.

V. Promote privacy

This pattern promotes privacy by implementing principle of least privilege and practice defense in depth. Privacy is improved because if the web server has been compromised it will not grant access to an uninformed user to user account. A well-structured back-end helps a lot to protect whole web server.

VI. Be reluctant to trust

This pattern fulfills this principle by granting access only to authorized person. Verification process held at two levels, front-end and at back-end. If a user authenticates by the front-end and not by the back-end cannot get access to user database.

3.8 Secure Assertion

Problem

Any server that is accessible from the internet is more vulnerable to be attacked. No matter how much efforts are expended in protecting it, the possibility exists that an attacker will penetrate the system. The most difficult attacks to detect are those that exploit vulnerabilities in the application itself. These attacks are generally not visible to intrusion detection systems because they are unique to the application and not widely known attack on standard COTs component. Application level attacks cannot be defended at the system level.

The firewall, the operating systems, the intrusion detection system, and even the application server may view all such traffic as legitimate application requests. For example, if a banking application fails to adequately validate electronic funds transfer requests, an attacker might use the system to cause a victim's bank account to become overdrawn. To the system, these requests look to be legitimate.

Solution

The secure assertion pattern transparently re-links all the application level sanity checks with a mechanism that integrates into the system wide intrusion detection or event reporting system. Any failed assertions encountered by the application are automatically reported to the system wide monitoring console as a possible security relevant event. The secure assertion pattern transparently reports the events that developers already detect and recover from.

Supported Security Principles of Secure Assertion

I. Secure the weakest link

This pattern is helpful to secure the weakest link. By sprinkling lots of sanity checks in the application may keep attentive the administrator about tempering of application. The secure assertion pattern also offers developers an interface for reporting detected problems discovered and recovered from. For example, a function that scans user input and replaces illegal and dangerous characters should report any such replacements via the provided reporting interface. Account lockout strongly helps to secure weakest link with the combination of this pattern.

II. Defense in depth

Secure assertion pattern transparently re-links all the application level sanity checks with a mechanism that integrates into the system wide intrusion detection or event reporting system. Any failed assertions encountered by the application are automatically reported to the system wide monitoring console as a possible security relevant event. Pattern provides developers with a reporting framework that allows system administrators to be aware of potentially security relevant events occurring within the application.

III. Promote privacy

This pattern improves accountability indirectly by preventing exploits that circumvent authentication. Pattern improves integrity by removing opportunities for compromise of the application and helps to ensure exploitation of remaining weaknesses to not go undetected.

IV. Be reluctant to trust

Pattern informs the system administrator about any sign of tempering with application that can be evidence for the presence of an attacker.

V. Hiding secret is hard

This pattern is used to check programmer assumptions about the environment and proper program behavior. These are application specific sanity checks sprinkled throughout the system. An attacker also remains unaware about this technique.

3.9 Server Sandbox

Problem

Web applications have huge number of unknown users and can be accessed from anywhere. Website that deals with user input has chance of hacking, and behaves un-trusted way. For example, many web servers contain logic errors that can be exploited to allow private files to be served over the internet. Other servers contain undiscovered buffer overflow errors that can allow client provided malicious code to be executed on the server [9].

Solution

Server sandbox pattern strictly limits the privileges that web application components possess at run time. This is most often accomplished by creating a user account that is to be used only by the server. Operating system access control mechanisms are then used to limit the privileges of that account on needed to execute [22]. This approach accommodates systems that require administrative privileges to start the application, but does not need those privileges during normal operations.

Supported Security Principles of Server Sandbox

I. Secure the weakest link

Server sandbox removes any global privilege that is not essential and replaces with specific user and group privilege. A compromised web server allows an external hacker to gain access to all global resources. Eliminating the global privileges ensure that the hacker have no access to useful utilities and operating system features.

II. Defense in depth

It applies security in layered form around vulnerable components that limit the application's ability to access resources and operating system APIs. By using operating system network filtering it prevents the server from initiating connections to other machines.

III. Least privilege

Server sandbox pattern strictly limits the privileges that web application components possess at run time. This approach accommodates systems that require administrative privileges to start the application but does not need those privileges during normal operations. The most common example of this is a UNIX application server. The application can start with additional privileges but once those privileges are no longer needed it executes a privilege drop to revoke it into the less privileged operating mode.

IV. Compartmentalization

Complex application may require some unsafe privileges throughout its execution time. In that case, the application is partitioned so that only a minimal component has the

dangerous privileges. The other component should run using restricted user account. Components that communicate directly with clients should have the bare minimum privileges. Components with dangerous privileges should be buffered from client requests by other components.

V. Promote privacy

This pattern greatly enhances privacy by preventing component vulnerabilities from causing the entire server to be compromised.

VI. Be reluctant to trust

A server sandbox removes any global privilege that is not essential and replaces them with specific user and group privileges.

4. Proposed Pattern: User Authentication for Mobile Devices

A recently published online Smartphone's user authentication protocol (OSAP) [23] can be used as a solution pattern by any mobile based client server application to solve their problems.

Problem

User authentication on mobile devices is difficult. People want their computing devices to keep the confidentiality of their private information. If the pin is sent in open form, users face many problems like session hijacking, man-in-the-middle attack, spoofing, dictionary attacks etc.

Solution

People want their computing devices to keep the privacy of their information. For this purpose, these devices must ask for proof of the user's right before giving access. This method is called authentication [24]. Once the given proof is confirmed the machine will act on the user's behalf.

Proposed user authentication for mobile devices provides better authentication of online Smartphone's users. This pattern uses server to store user information instead of device. Pin is sent in encrypted form by using secure technique. This technique decreases the drawbacks of existing user authentication schemes developed for applications like internet banking, E-commerce, and instant messaging. Pattern provides online registration and pin is generated on server. Following attacks have been removed by using this pattern.

- Man in the middle attack
- Phishing attacks
- Dictionary attacks
- Brute force
- Shoulder force
- Guessing attacks

How to Authenticate Users?

User authentication pattern consists of two element, client side and server side. Client side is used for entering data by user. Server side is responsible for generating the pin and authenticating the legitimate users. In this pattern, user's detail like contact number, IMEI number, and pass code is used to register the client at server. PIN is generated at server side and LSB technique is used to hide the PIN inside cover image. Server sends back the image to user then user decrypts the image to get the pin. Whole process is summarized in Figure 1. Trade-offs is explained in Table 3.

Security Principles Supported by User Authentication for Mobile Devices Pattern

I. Secure the weakest link

Pattern successfully fulfills this principle. User would not be authenticated successfully if IMEI of device is changed and user has provided incorrect PIN. This pattern provides authentication and resistance against attacks like phishing, shoulder surfing, brute force, content injection, dictionary, and guessing attack. Successful authentication ensures that LSB has safely transferred the pin from server to user, and user has not changed his /her device. It has successfully authenticated the device and user.

II. Practice defense in depth

User authentication pattern defenses in depth by securing user pin by applying LSB technique. If someone steals user pin and tries to access the account, the server will reject that access. This pattern provides authentication of user as well as device. If device is changed it will not authenticate even the original user because it joins the user account with that exact device.

III. Least privilege

This pattern limits unauthorized access and provides better authentication in online applications using mobile phones. Our scheme is efficient and user friendly. User asks to enter his data once in the registration phase. LSB technique is used to cover the pin. Therefore, illegal user cannot decrypt it. Only legitimate user will be able to decrypt and confirm back.

IV. Fail securely

If an attacker obtains the stego image, he will have to X-OR his IMEI with pin for authentication. This X-ORed pin is then sent to server for confirmation. Server authenticates the pin by X-ORing this pin with IMEI that it has saved earlier from original user. Since attacker IMEI will be different from original user, a different pin will be generated. This will not match at server side thus attacker will not get authenticated, as a result the attacker will not be able to steal authentication details.

V. Keep it simple.

This pattern is simple in use. It provides ease for non-technical users. User will decrypt and confirm back by just pressing the confirm button.

VI. Promote privacy
This pattern highly promotes privacy and improves the efficiency. User information will not leak by using this pattern.
VII. Hiding secrets is hard
In this pattern, user information is encapsulated by using highly secure technique.
VIII. Be reluctant to trust
This principle is foundation of this pattern. It only authorizes legitimate user. Any user with wrong pin or wrong IMEI number will simply be rejected.

Table 3: Trade-offs of user authentication for mobile devices pattern

Accountability	User authentication pattern increases accountability by ensuring user accounts not to be compromised using a password guessing attack.
Confidentiality	Confidentiality of user data is increased by using user authentication pattern.
Integrity	Integrity of individual account is enhanced to improve the integrity of the site.
Performance	It improves the performance of the account and application.
Cost	Cost has reduced.

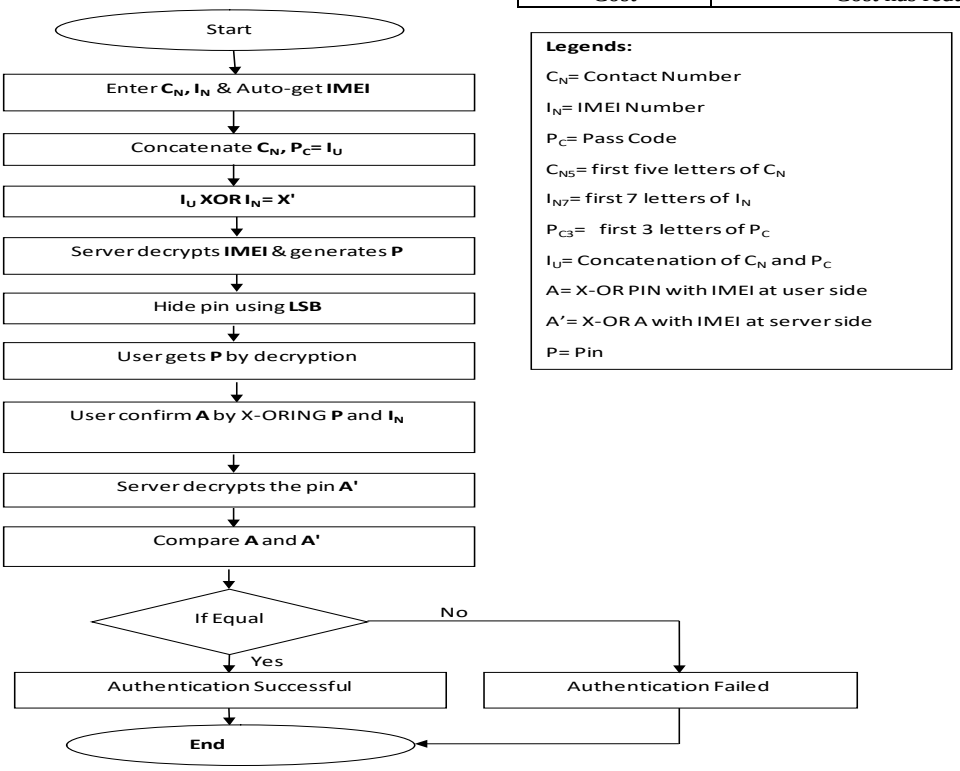

Figure 1: OSAP Process

5. Security Pattern Comparison

Table 4 provides comprehensive results of this study. It shows evaluation of ten security principles against web based security patterns. This comparison helps us in deciding that which pattern we should follow to fulfill our security requirements while designing a system. The results of this research encourage the use of pattern based approach to develop secure business processes and web applications. At the same time, it helps management and designers in selection of the most suitable patterns according to their circumstances and requirements. The enforcement of security in the software development life cycle of the application reduces the high cost and efforts associated with implementing security at a later stage. This study clearly emphasizes the effectiveness of these patterns in handling the security issues. It is specifically useful in deciding what security patterns to apply by keeping the security attacks in the mind. As the attack rate is increasing at an exponential rate in web applications due to the advancement of technology.
Sometimes one pattern cannot take care of all the security requirements and we need a group of patterns to meet all the security requirements in our system. Some patterns use others as base to improve security. So system designers can use other patterns to support and improve security

measures. Based on the conducted survey and evaluation of patterns against security principals, we have grouped various patterns which can be combined to achieve high level of security. Table 5 shows the related patterns, the collective use of these patterns help to achieve the ten security principles in any system.

Table 4: Security patterns vs Security principles

Patterns	Secure weakest link	Defense in depth	Fail securely	Least privilege	Compartmentize	Keep it simple	Promote privacy	Be reluctant to trust	Hiding secret is hard	Community resources
Account lockout	Yes	Yes	No	Yes	No	No	Yes	Yes	Yes	No
Authenticated session	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes	No
Client input filter	No	Yes	No	No	No	No	Yes	Yes	No	No
Encrypted storage	No	Yes	Yes	No	No	No	Yes	Yes	Yes	No
Mine field	Yes	Yes	Yes	Yes	No	No	No	Yes	No	No
Network address blacklist	No	Yes	No	No	Yes	No	Yes	Yes	No	No
Password propagation	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No
Secure assertion	Yes	Yes	No	No	No	No	Yes	Yes	Yes	No
Server sand box	Yes	Yes	No	Yes	Yes	No	Yes	Yes	No	No
User authentication for mobile devices	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No

Table 5: Related patterns

Patterns	Related Patterns
Authenticated session	Network Address Blacklist and Password Propagation
Account lockout	Authenticated Session, User authentication for mobile devices, Network Address Blacklist
Encrypted storage	Client Input Filters
Minefield	Network Address Blacklist
Network address blacklist	Account Lockout, Client Input Filters and Minefield, User authentication for mobile devices
Password propagation	Account Lockout and Authenticated Session
Secure assertion	Server Sandbox and Network Address Blacklist
Server sandbox	Minefield a related pattern
Client input filter	Network Address Blacklist
User authentication for mobile devices	Account Lockout, Encrypted Storage, Password Propagation

6. Conclusion and Future Work

Patterns are a promising proposal towards security. These are useful to build and evaluate systems. Security patterns help us consider non-functional security requirements at the beginning of the design. In the security critical applications, it is extremely important to avoid mistakes. Therefore, the use of security patterns is important for developing a secure system.

This study highlights template of pattern, security patterns, and ten security principles and also presents set of patterns that are used in different web applications to handle security issues. We have discussed structural security patterns in detail and compared them with ten security principles. Combinations of patterns are extensible because of the possibility of replacing a pattern with another concrete realization of the same pattern. They are reusable because of the possibility of replacing several of the used patterns to fit the requirements of a new application.

There are many patterns with different purposes. Reason why developers must combine many patterns may include establishing a high degree of security within the system. But they have the problem of choosing which pattern must be used and deciding which pattern will better adapt to the system security requirements. We also propose a security pattern for mobile device.

In future, this work would be extended by studying the different security architectures existing in the systems design, together with defining a method to specify flexible security architectures that can easily adapted to systems with very different security requirements as well as guarantee security using security patterns.

References

[1] Alexander, C.; Ishikawa, S.; Silverstein, M. The Timeless Way of Building. International Journal of Computer Science and Network Security, 1979, 2(4), 123-220.

[2] Tešanović, A. What is a pattern. In Dr.ing. course DT8100 (prev 78901/45942/DIF8901) Object-oriented Systems. IDA Department of Computer and Information Science, Linköping, Sweden, 2005.

[3] Blackley, B.; Heath, C. Technical Guide, Security Design Patterns. International Journal of Research in Engineering and Technology software, 2004, 2(3), 25-55.
[4] Aleksandra, T. What is a pattern?. Journal of Research in Computer and Information Science, 2002, 55-134.
[5] Heyman, T.; Yskout, K.; Scandariato, R. An Analysis of the Security Patterns Landscape, Proc. Of 29th International Conference on Software Engineering Workshops, 2007, 25-65.
[6] Kienzle, D.; Elder, M.; Tyree, D.; Edwards, J. Security Patterns Template and Tutorial. 2002. Available online: http://www.securitypatterns.com/documents.html.
[7] Viega, J.; McGraw, G. Building Secure Software. In How to Avoid Security Problems the Right Way, Addison-Wesley Professional Computing Series, 2011.
[8] Yoder, J.; Barcalow, J. Architectural Patterns for Enabling Application Security. Proc. of the Pattern Languages of Program Design, 2000, 301-336.
[9] Braga, A.; Rubira, C.; Dahab, R. Tropyc: A pattern language for cryptographic software, 1998.
[10] Fernandez, E.B.; Yuan, X.Y. Securing analysis patterns. Proc. of the 45th ACM South East Conference, 2007, 36-57.
[11] Eduardo, B.F.; Rouyi, P. A pattern language for security models. Proc. of 8th Conference on Pattern Languages of Programs, 2001, 234-368.
[12] Wassermann, R.; Cheng, B.H. Security patterns. International Journal of Computer Science, 2003, 8(6), 23-125.
[13] Schmidt, D.C.; Stal, M.; Rohnert, H.; Buschmann, F. Pattern-oriented Software Architecture, In Patterns for Concurrent and Networked Objects. Wiley & Sons, 2000, 4(2), 57-156.
[14] Alam, S. Using security patterns in Web Application, GRIN Verlag, 2013, USA .
[15] Cuevas, A. et. al. Security Pattern for Capturing Encryption based Access Control to Sensor Data. Proc. International Conference on Emerging Security Information, Systems and Technologies, 2008, 62-67.
[16] Fernandez, E. B.; Yuan, X.Y. Semantic Analysis Patterns. Proc. of the 19th International Conference on Conceptual Modeling, 2000, 183-195.
[17] Coggeshall, J. Session Authentication. 2001. Available online: http://www.zend.com/zend/spotlight/sessionauth7may.php.
[18] Dan, W. Security Functional Requirements Analysis for Developing Secure Software. Dissertation, USC. May 2007.
[19] Steel, C.; Nagappan, R.; Lai, R. Core Security Patterns. Best Practices and Strategies for J2EE. Pearson Education, 2005, India.
[20] Morrison, P.; Fernandez, E.B. The Credentials Pattern. Proc. of Pattern Languages of Programs Conference, 2006, 345-360.
[21] Hallstrom, J. O.; Soundarajan, N.; Tyler, B. Monitoring Design Pattern Contracts. Proc. of the FSE-12 Workshop on Specification and Verification of Component-Based Systems, 2004, 87-94.
[22] Hafiz, M. A. Collection of Privacy Design Patterns, Proc. of ACM PLoP., 2006, 1–13.
[23] Riaz, R. Rizvi, S.S. Mushtaq, E. Shokat, S. OSAP: Online Smartphone's User Authentication Protocol. International Journal of Computer Science and Network Security, 17(3), 2017, 7-12.
[24] Riaz, R. Chung, T.S. Rizvi, S.S. Yaqub, N. BAS: The Bi-phase Authentication Scheme for Wireless Sensor Networks. International Journal of Security and Communication Networks, 2017.

Rabia Riaz is working as assistant professor in university of Azad Jammu and Kashmir. She is currently chairperson of department of Software Engineering. She holds a PhD in electrical engineering from AJOU University, South Korea. Her research interests include, wireless network, sensor networks, data security, encryption and authentication mechanism.

Sanam Shahla Rizvi received the B.C.S. degree in Computer Science from Shah Abdul Latif University, Khairpur Pakistan, in 2003, and the M.C.S. degree in Computer Science from KASBIT University, Karachi Pakistan, in 2004, and M.S. degree in Computer Science from Mohammad Ali Jinnah University, Karachi Pakistan, in 2006, and Ph.D. degree in Information and Communication Engineering from Ajou University, Suwon, South Korea, in 2010. She is currently working as associate professor at Department of Computer Sciences at Preston University, Karachi, Pakistan. Her research interests include flash memory storages, data management, database systems, indexing structures, and wireless sensor networks.

Farina Riaz received the BSCS degree in Computer Science from Quaid-e-Azam University, Islamabad Pakistan, in 2007, and the M.S. degree in Software Engineering from National University of Science and Technology, Islamabad Pakistan, in 2010. She has been working as adjunct lecturer in Manipal University, Dubai Campus for more than two years. Her research interests include wireless networks, sensor networks, data security, and encryption and authentication mechanism.

Sana Shokat received her MS (Software Engineering) degree from Bahria University Islamabad. She is currently candidate of P.h.D degree at University of Azad Jammu & Kashmir, Muzaffarabad, Pakistan.