Web Security Patterns for Analysis and Design

Takao Okubo
Fujitsu Laboratories ltd.
Kamikodanaka 4-1-1, Nakahara-ku,
Kawasaki,Japan
okubo@jp.fujitsu.com

Hidehiko Tanaka
Institute of Information Security
Tsuruyacho 2-14-1, Kanagawa-ku,
Yokohama, 221-0835 Japan
tanaka@iisec.ac.jp

ABSTRACT

Although security requirements analysis plays a very significant role in secure software development, it is difficult since it requires much security expertise and man-power. Plain and practical security requirements patterns are needed. We have presented a visualized analysis approach for eliciting security requirements by extending misuse cases, and found that some of its results can be pattern candidates. This paper proposes 8 new web security requirements patterns with our analysis approach. The proposed patterns give analysts a way to find a proper pattern for a specific security goal. They are related to security solutions, and also produce some security design possibilities. We have applied these patterns to some case studies and evaluated that they are effective for web security analysis.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Patterns*

General Terms

Security

Keywords

Analysis, design, reruirements, security pattern, web application program

1. INTRODUCTION

In this paper we present some web security requirements patterns which have been mined by applying our new security analysis approach.

Security issues have become more and more important in software development. In spite of software developers' efforts, many security incidents are reported every day. These incidents occur because security functionalities are not realized properly, or they are disregarded from the beginning.

One of the reasons why security is disregarded is the lack of security aspects in the lifecycle. In general software development, analysts only have to consider the normal behavior of programs: the intended usage by proper users. However, in order to elicit security requirements, analysts have to consider abuse or misuse of the functions from the viewpoint of adversaries. This is one of the biggest differences between security analysis and functional analysis. Currently, for security, another special analysis is done by security experts, who have the expertise of what adversaries want and where and how attacks are performed. They do the analysis from the viewpoint of the attackers and find what are the threats for the target system, this is called "threat analysis".

Unfortunately, threat analysis is difficult without security expertise. Analysts have to identify in advance potential vulnerabilities which may occur in the programming stage. There are web program-specific vulnerabilities such as cross-site scripting (XSS)[11], cross-site request forgery (CSRF)[10] and injection attacks[9]. But current security analysis methods cannot identify systematically such threats in the later stages. It requires a great deal of manpower, meanwhile the number of security experts is limited.

If there exists an efficient security requirements method, analysts can save the manpower for verifying the coverage of threat identification. If there are security requirements patterns, analysts may be able to skip some requirements steps by reusing the patterns.

We have proposed a new security requrements method which enables identification of potential threats in programming and expanding security coverage with misuse cases[7]. Although misuse cases are well visualized and efficient for understanding and verification, they lack expression of assets and they are not useful for asset-based threat analysis. Therefore we have extended misuse cases for asset-based analysis. We have applied our approach to the web domain and successfully mined 8 analysis patterns. The contexts of these patterns are clear, so analysis can find the proper patterns from use case diagrams of the target program.

The target audience of this paper are web application developers who have some knowledge about web security such as XSS, but who are not security experts. Security experts may be able to confirm the correctness of the proposed patterns. developers with little security knowledge may be able to learn that if they specify the contexts of their application, they can find the possible threats and countermeasures automatically by applying the proposed patterns.

Section 2 reviews current security patterns and their issues. In Section 3.1 our efficient security analysis approach

with the extension of misuse cases is presented. In Section 3.2 new web security patterns mined with Section 3.1's approach are presented. Section 4 examines the presented patterns with some case studies. Section 5 concludes this paper. The Appendix describes the remaining 6 patterns.

2. CURRENT SECURITY PATTERNS

In this section, current security patterns are evaluated in terms of web security. Several security patterns are presented in [6]. Some of them can be categorized as requirements patterns and the others as design patterns.

1. Security patterns for mechanisms
   Identification and Authentication (I&A), Authorization, Access control and logging are kind of design patterns. These patterns are already well known and used. However, there are more web security vulnerabilities such as injection attacks (SQL injection[9], pass traversals), cross-site scripting(XSS[11]) and cross-site request forgery(CSRF[10]) and current patterns or frameworks do not provide complete remedy for those vulnerabilities.
   Design patterns contain another issue different from the security domain. Most design patterns do not have so much critical effect on software quality as on security quality. For example, even without applying the "Visitor pattern", it is possible to build software with the same behavior as the software with the "Visitor pattern". However if "I&A pattern" is not applied, it may be difficult to keep it secure. Therefore, security design patterns should satisfy the following requirements.
   - Appropriate patterns must be found for certain security requirements.
   - Their application should be forced.
   - Their application should be easily verified.
2. Security requirements patterns
   As requirements patterns, the steps for eliciting security requirements are presented in [6]. They are security needs identification, asset valuation, threat assessment, vulnerability assessment, risk determination, etc. Although they are necessary for eliciting requirements[2][4], they are not enough for analysts to select the right security approach. More instance patterns are required.

As a result of the analysis of current patterns, we have found that security design patterns require analysis in advance for finding the proper context applying the patterns. Security requirements patterns should be the first step.

3. PROPOSED WEB SECURITY PATTERNS

In this section, we propose some new security requirements patterns for web application programs. They are practical instance patterns, so that analysts can easily find the proper patterns for satisfying the security requirements of their programs. Moreover, the proposed requirements patterns are strongly related to the security designs (some of which are design patterns). This approach solves the issue mentioned in the previous section, that appropriate patterns must be found for certain security requirements.

3.1 A New Analysis Approach with Extended Misuse Case

The proposed patterns have been mined by applying a new security analysis approach[5]. The approach enables expanded security coverage by improving the expressiveness of Sindre and Opdahls' misuse case approach[7].

1. Asset-based extension
   - Data assets
     Although assets (especially data assets) are essential factors for threat analysis[2][4], they aren't explicitly described in the original misuse case diagram. Therefore the proposed method adds a definition of data assets as shown in Figure 1. A UML note in Figure 1 represents a data asset interchanged between an actor and a use case, and the "flow direction" stereotype represents the direction of the interchange. The "flow direction" have two types; "⟨⟨send⟩⟩" and "⟨⟨receive".

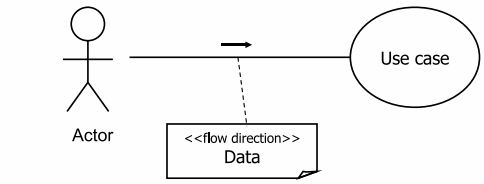
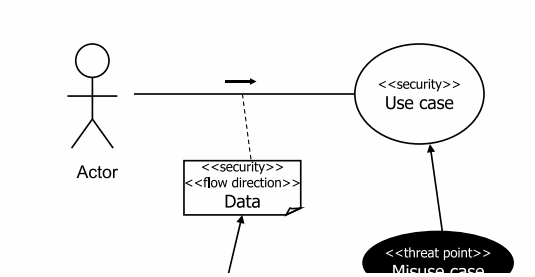


Figure 1: Asset-based extensions.

   - Security properties on assets
     Security properties of confidentiality (C), integrity (I), availability (A) and privacy(P) are added on the asset elements (data assets/ use cases) in the form of 'stereotype' of UML(see Figure 2).
   - Targets of misuse cases
     In the original misuse case diagram, targets of the misuse cases are only use cases. We have extended the original notation so that the misuse cases can point data assets in addition to use cases(see Figure 2.
2. Extension of architecture information
   The original misuse case diagram[7] has no architectural information. Our new approach[5] added a notation of stereotype "threat point" to a misuse case which represents the architecture points where the target threat may be possible. The "threat point" stereotype takes 3 types; "⟨⟨ client⟩⟩", "⟨⟨network⟩⟩" and "⟨⟨server⟩⟩". Figure 2 shows this extension. We have also added the notation representing dependency between threats. The arrow named "enables" means that the misuse case at the end point is enabled by the misuse case at the start point. This extension aims at rough assessment of the possibility of each threat. In



F **re**
n

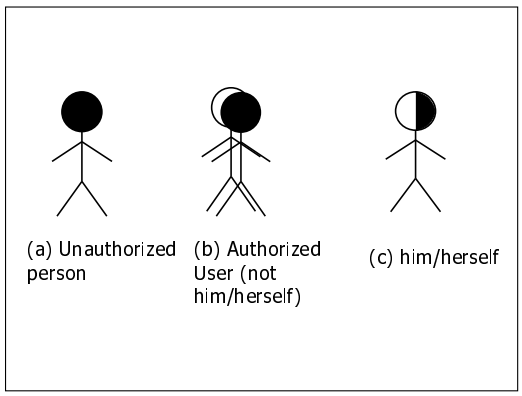


Figure 3: Fine classification of mis-actors.

the analysis stage, fine grained threat assessments like attack tree[4] are difficult to apply because details of the system architecture have not yet been determined. However, in order to elicit security requirements,$_1$security analysis needs (even rough) threat assessments.

3. Adding types of mis-actors
   The original misuse case diagram[7] contains only one kind of mis-actor. But security measures must differ depending on the types of mis-actors:

   (a) unauthorized person

   (b) other authorized user (not him / herself)

   (c) actor him / herself

   User identification and authentication (I&A) is effective as a measure against abuse of use cases for type (a) of mis-actor, but not for type (c). For type (c), another security measure is required. As just described, there is a risk that some of the measures related to a single mis-actor may be missed. Therefore, our new method[5] defines three types of mis-actor as shown in Figure 3.

The proposed approach is also used for describing the patterns graphically.

3.2 Web Security requirements patterns with the New Approach

The proposed security patterns have been mined through the application of our approach to web programs. We have found that some typical threats and solutions are identified in certain patterns of the use cases and assets. We have formalized them as web security requirements patterns.

- "Abuse" pattern
- "Confidential data being sent" pattern
- "Integral data being sent" pattern
- "Shared data being received" pattern
- "Proivate data being received" pattern
- "Integral data being received" pattern
- "ID/Password authentication" pattern
- "Session management" pattern

Each pattern involves the context, problem patterns and solution patterns. The context navigates the right context (use case and asset types) that the pattern should be applied. Problem sections help identifying potential threats in programs. Solution sections can be used to select proper countermeasures against certain threats.

In the following sections, two patterns will be described in detail. The pattern description uses the same pattern characteristics formats defined in [6]. Refer to the appendix for details of the rest of the patterns.

3.2.1 "Abuse" pattern

Context This pattern should be applied if it is undesirable that the target use case is "abused". "Abuse" means an unintended use violating availability of the target use case. It represents the following threats or attacks.

- Unauthorized use: use by a person who is not allowed to use the use case.
- Unintended use: use for unintended purposes such as fraud.
- Attacks such as Denial of service (DoS).

Security property "availability" is required for use cases of the target program. This pattern does not care whether there is a data asset or not.

Table 3.2.1 and Figure 4 shows the context. Analysts can find the applicability of this pattern by verifying whether the usecase diagram of their program contains the matching pattern of Figure 4.

Characteristic	condition	security	data flow
Use case type	any	availability	n/a
Data asset	any	any	any

Table 1: "Abuse" pattern: context.

Problem See Figure 5, 6 and 7. Consequently, there exist the following threats in the program.

1. Unintended use of the use case (including abuse and Denial of Service (DoS) attack) by an unauthorized user and an authorized user.
2. Spoofing of the use case function (such as a server application) by an unauthorized user,
3. Repudiation by an authorized user (him/herself).
4. Spoofing of users which enables unintended use and repudiation.
5.

In Figure 5 there are "enabled" notations between somi misuse cases. It means that some misuse cases may become possible due to other misuse cases. It is similar to the association of threat trees and fault trees.

Solution See Figure 8, 9 and 10. Solutions are the following:

1. I&A is useful for preventing unintended use of the use case by an unauthorized user.
2. Server authentication such as SSL/TLS and providing server comfirmation methods (such as not to hide the address-bar of web browsers) are useful for preventing or detecting spoofing of the use case function.
3. I&A is useful for preventing user spoofing.
4. Logging is useful for detecting abuse and repudiation.

Figure 4: "Abuse" pattern: context.

Consequences By applying the problem pattern, analysts can easily find all the typical web threats in the general use cases. They can also use it to confirm their result of the threat analysis. By applying the solution pattern, they can identify design of the security functions as countermeasures against the threats. Some of them can be related to security design patterns (see "See also").

See Also The following patterns are related to this pattern. For example, unintended uses and DoS attacks threaten use cases required availability.

- I&A[6]
- Logging[6]
- Secure networks (SSL)[6]

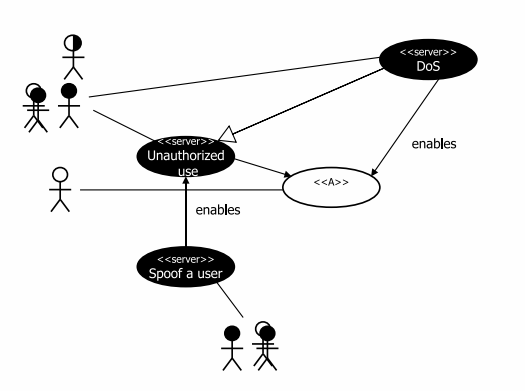


Figure 5: "Abuse" pattern: problem (1).

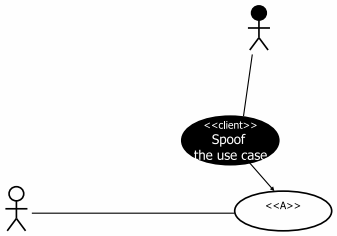


Figure 6: "Abuse" pattern: problem (2).

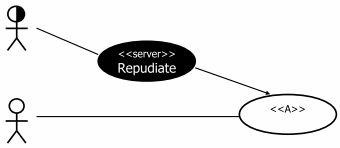


Figure 7: "Abuse" pattern: problem (3).

3.2.2 "Confidential data being sent" pattern

Context This pattern can be applied when there are some "confidential" data assets sent from actors to use cases. Table 3.2.2 and Figure 11 shows the context.

Problem From Figure 12, we can find that, there exists the following threats in the program.

1. Disclosure data being sent on the client enabled by vewing caches.

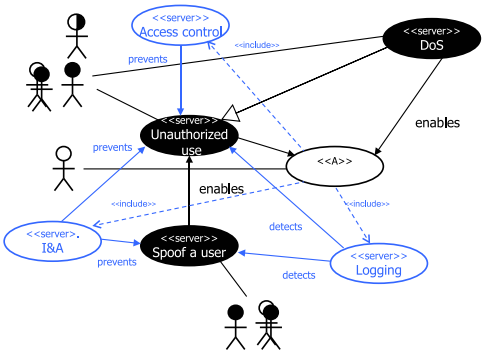


Figure 8: "Abuse" pattern: solution (1).

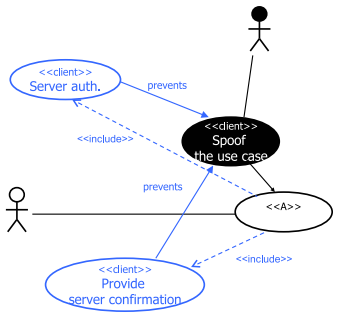


Figure 9: "Abuse" pattern: solution (2).

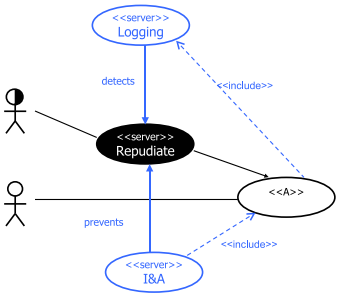


Figure 10: "Abuse" pattern: solution (3).

Characteristic	condition	security	data flow
Use case type	any	any	n/a
Data asset	required	confidentiality	send

Table 2: "Confidential data being sent" pattern: context.

2. Disclosure data being sent on the client enabled by shoulder hacking.
3. Disclosure of the sending information on the network enabled by man in the middle (MITM) attacks.
4. Disclosure of the sending information on the server enabled by injection attacks[9].
5. Disclosure of the sending information on the server enabled by XSS[11] attacks.
6. Disclosure of the sending information on the server enabled by inappropriate access control.

Solution See Figure 13. Solutions are as the following.

1. Disabling caches is useful for preventing disclosure data on the client enabled by vewing caches.
2. Hiding the data is useful for disclosure data on the client enabled by shoulder hacking.
3. Encryption of communication (such as SSL/TLS) is useful for preventing disclosure on the network enabled by MITM attacks.
4. Countermeasures against injection attacks (such as use of parameter binding mechanism) are useful for preventing disclosure on the server.
5. Countermeasures against XSS (such as sanitizing output) are useful for disclosure on the server.
6. Correct access control are useful for preventing disclosure on the server.

Consequences Injection attacks such as SQL injection, OS command injection are performed with injecting malicious input including some part of SQL/ OS commands. Injection attacks enable tampering and disclosure of information. Therefore if there is a data asset being sent (this context), developers need to implememnt countermeasures preventing injection attacks.

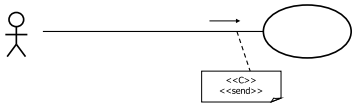


Figure 11: "Confidential data being sent" pattern: context.

See Also The following patterns are related to this pattern.

- Secure networks (SSL)[6]

4. CASE STUDIES

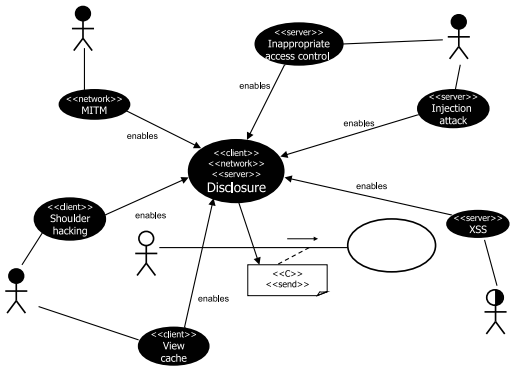


Figure 12: "Confidential data being sent" pattern: problem.

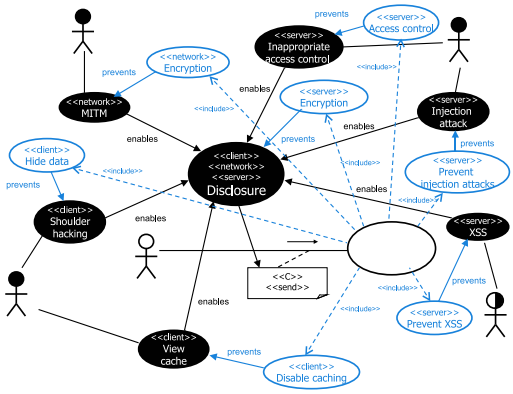


Figure 13: "Confidential data being sent" pattern: solution.

4.1 Web Bookstore

We had some case studies when we apply our patterns to several sample application programs. This paper presents one of these applications, which is a web bookstore such as Amazon.com. It has three use cases:

1. Search books
   Customers search books they want to buy. They may want to keep secret the search keywords and search results.

2. View/edit personal information
   Customers are required to input their personal information such as name, address and telephone number. They are allowed to view and edit their own information on the web, but not other customers' information.

3. Order books
   Customers can order books. The order list sent from customers to the server must not be seen or modified by any other persons.

Before applying our pattern, the extended use case to which data assets and security properties are added must be drawn. Figure 14 presents the extended use case diagram. After drawing the diagram, each part of the diagram is compared to the contexts of our patterns. If it matches one of the

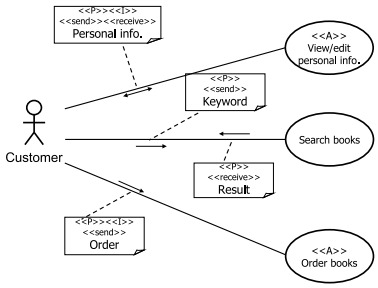


Figure 14: The extended use case diagram of a web bookstore.

contexts, the pattern of the context can be applied. In Figure 14, at the "View/edit user information" use case part, "Abuse" pattern (Section 3.2.1, "Leaks of data being sent" pattern (Section 3.2.2), "Tampering data being sent" pattern (Section A.1), "Disclosure of private data" pattern (Section A.3) and "Tampering receiving data" pattern (Section A.4) can be applied. At the "Search books" part, "Leaks of data being sent" pattern(Section 3.2.2) and "Disclosure of private data" pattern (Section A.3) can be applied. At the "Order books" part, "Abuse" pattern (Section 3.2.1), "Leaks of data being sent" pattern (Section 3.2.2) and "Tampering data being sent" pattern (Section A.1) can be applied. The result of the first analysis iteration is presented in Table 4.1.

In this case it requires a second iteration. In the first iteration another asset "password" has been added to the system, so we have to consider threats to passwords. If analysts choose ID/password authentication and password management, "ID/password" pattern (Section A.5) and "Session management" pattern (Section A.6) may be applied. Therefore, threats and solutions on Table 4.1 are added to the result of the first iteration.

4.2 Discussion

Efficiency of patterns Usual security analysis requires the following steps[6][2][4].

1. Use case definition
2. Asset identificaton
3. Asset valuation
4. Threat identification
5. Threat assessment &risks
6. Security solution elicitation
7. Security solution determination

With the presented patterns, some of the steps 4 to 6 can be skipped by matching the pattern context to the given use cases and assets. Analysts can obtain typical web threats and solutions from patterns. However, the threats in the presented patterns are only typical ones. Analysts may have to check if there are others.

Threat	asset	mis-actor	location	solution	solution type
Abuse	view/edit personal info. order books	unauthorized person	server	I&A	prevent
Abuse	view/edit personal info. order books	unauthorized person	server	logging	detect
Leaks	personal info. Search keyword order	other person	client	countermeasure against XSS	prevent
Leaks	personal info. search keyword order	other person	network	encryption	prevent
Leaks	personal info. search keyword order	other person	server	countermeasure against injection	prevent
Disclosure	personal info. search result	other person	client	disabling caching	prevent
Disclosure	personal info. search result	other person	client	I&A	prevent
Disclosure	personal info. search result	other person	network	encryption	prevent
Disclosure	personal info. search result	other person	server	access control	prevent
Disclosure	personal info. search result	other person	server	countermeasure against injection	prevent
Tampering	personal info. order	other person	client	I&A	prevent
Tampering	personal info. order	other person	network	encryption	prevent
Tampering	personal info. order	other person	server	countermeasure against injection	prevent
Tampering	personal info. order	other person	server	countermeasure against XSS	prevent
Tampering	personal info. order	other person	server	countermeasure against CSRF	prevent
Tampering	personal info. order	other person	server	logging	detect

Table 3: The analysis result of the first iteration.

Threat	asset	mis-actor	location	solution	solution type
Disclosure	password	unauthorized person	client	hiding password	prevent
Disclosure	session token	unauthorized person	client	countermeasure against XSS	prevent
Disclosure	session token password	unauthorized person	client	cookie management	prevent
Disclosure	session token	unauthorized person	network	SSL	detect
Brute force attack	password	unauthorized person	server	strong password spec.	prevent
Dictionary attack	password	unauthorized person	server	strong password spec.	prevent
Session fixation	session token	unauthorized person	server	countermeasure against session fixation	prevent
Session hijack	session token	unauthorized person	server	countermeasure against session hijack	prevent

Table 4: The analysis result of the second iteration.

Sufficient condition The case study result of the "web bookstore" points that the presented threat identification for the patterns is good sufficiency. It enables threats referred in web threat classifications such as [8] to be identified when applying the patterns.

Necessary condition The presented patterns, by themselves, cannot give the reasoning about what kind of threats are required for the given program. The reasoning has to rely on "security expertise" because the patterns are based on the result of human analysis, although it is visualized for easy understanding. To prove the necessary condition, we need more empirical evaluation for our analysis method.

5. CONCLUSIONS

This paper proposed some new requirements patterns for web security. The proposed patterns solves one of the security-related problem: how to find a proper pattern for a certain security goal. Each context of the pattern gives analysts the information whether it should be applied to their pro-

grams or not. Contexts, problems, and solutions are represented visually which helps the understanding of analysts and stakeholders.

The proposed patterns also provide security design possibilities. They appear in the "Solution" characteristics of patterns. Some of the security design candidates, such as I&A are related to the current security design patterns. which are presented in "See also".

Our case study shows that if the use case and asset analysis is done, it is easy for analysts to find and apply our patterns. As a result analysts obtain sufficient threats and suggestion of their solutions.

Our approach is based on the misuse case approach. It does not decompose the activity sequences because our target development fields require lightweight analysis for requirements alicitation. If more precise analysis is allowed, there is another requirements elicitation approach using misuse activities which extends activity diagrams[3][1]. Misuse activities enables more granular analysis for possible threats. Then if more precise analysis is allowed, the misuse activities may be the alternative.

As mentioned in Section 4.2, though the presented patterns have ability to identify sufficient threats and solutions, currently it is difficult to prove that the proper threats are identified for the given situation. A more empirical evaluation is required, which is future work.

6. ACKNOWLEDGMENTS

We appriciate our shepherd, Eduardo Fernandez and all the atendeees of the "Security & Quality" writers' workshop for valuable comments that improved this paper.

7. REFERENCES

[1] Braz, F., Fernandez, E. B., and VanHilst, M. Eliciting security requirements through misuse activities. In *accepted for the 2nd Int. Workshop on Secure Systems Methodologies using Patterns (SPattern'07). In conjunction with the 4th International Conference onTrust, Privacy & Security in Digital Busines(TrustBus'07)* (Turin, Italy, 2008).

[2] CC. Common criteria for information technology security evaluation v3.1, 2007[Online]. Available: `http://www.commoncriteriaportal.org/public/developer/index.php?menu=2`. (accessed 2008-06-28).

[3] Fernandez, E. B., VanHilst, M., Petrie, M. M. L., and Huang, S. Defining security requirements through misuse actions. In *in Advanced Software Engineering: Expanding the Frontiers of Software Technology, S. F. Ochoa and G.-C. Roman (Eds.), International Federation for Information Processing* (2006), Springer, pp. 123–317.

[4] Howard, M., and LeBlanc, D. *Writing Secure Code Second Edition*. Microsoft, 2003.

[5] Okubo, T., and Tanaka, H. Identifying security aspects in early development stages. In *Proc. International Conference on Availability, Reliability and Security (ARES'08)* (Barcelona, Spain, Mar. 2008), pp. 1148–1155.

[6] Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., and Sommerlad, P. *Security Patterns*. Wiley, 2006.

[7] Sindre, G., and Opdahl, A. L. I. Eliciting security requirements by misuse cases. In *Proc. TOOLS Pacific 2000* (2000), pp. 120–131.

[8] Threat classification, 2007[Online]. Available: `http://www.webappsec.org/projects/threat/`. (accessed 2008-06-28).

[9] WASC. Web application security consortium threat classification: Sql injection, 2007[Online]. Available: `http://www.webappsec.org/projects/threat/classes/sql_injection.shtml`. (accessed 2008-06-28).

[10] Watkins, P. Cross-site request forgeries, 2007[Online]. Available: `http://www.tux.org/~peterw/csrf.txt`.

[11] Cert advisory ca-2000-02 malicious html tags embedded in client web requests, 2000[Online]. Available: `http://www.cert.org/advisories/CA-2000-02.html`. accessed(2008-06-28).

APPENDIX

A. APPENDIX: WEB SECURITY REQUIREMENTS PATTERNS (REST 6 PATTERNS)

A.1 "Integral data being sent" pattern

Context This pattern can be applied to a program where some data assets required integrity are sent from actors to use cases. Table A.1 and Figure 15 shows the context.

Characteristic	condition	security	data flow
Use case type	any	any	n/a
Data asset	required	integrity (I)	sent

Table 5: "Tampering data being sent" pattern: context.

Problem See Figure 16 and Figure 17. Consequently, there exist the following threats in the program.

1. Tampering data being sent on the network enabled by MITM attacks.
2. Tampering data being sent on the server enabled by injection attacks, XSS and CSRF.
3. Tampering data being sent on the server enabled by user spoofing.
4. Repudiation of data being sent by a user him/herself.

Solution See Figure 18 and 19.

1. Encryption such as SSL/TLS is useful for preventing MITM attacks which enables tampering on the network.
2. Countermeasures against injection attacks, XSS and CSRF are useful for preventing tampering on the server.
3. I&A against spoofing is useful for preventing tampering on the server.
4. Logging is useful for detecting tampering on the server.
5. Original Assurance such as digital signature, time assurance and I&A is useful for preventing repudiation.

See Also The following patterns are related to this pattern.

- I&A[6]
- Logging[6]
- Secure networks (SSL)[6]

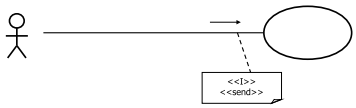


Figure 15: "Integral data being sent" pattern: context.

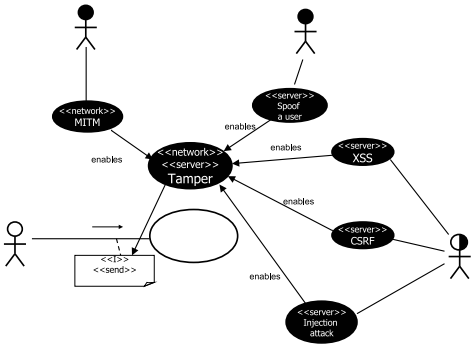


Figure 16: "Integral data being sent" pattern: problem (1).

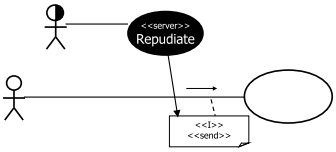


Figure 17: "Integral data being sent" pattern: problem (2).

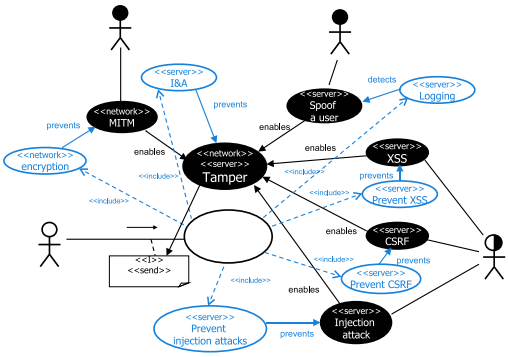


Figure 18: "Integral data being sent" pattern: solution (1).

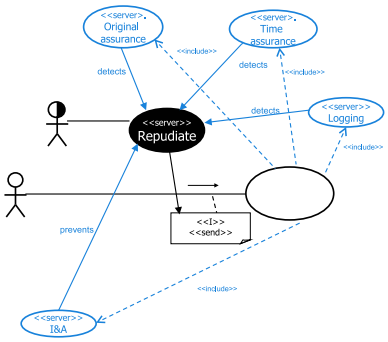


Figure 19: "Integral data being sent" pattern: solution (2).

A.2 "Shared data being received" pattern

Context This pattern can be applied to the program which shared data assets are sent from use cases to actors. Table A.2 and Figure 20 shows the context. "Shared" data means that its access is allowed to the actors of the same role. "C1" stereotype in Figure 20 represents that the data is required "shared" level of confidentiality.

Characteristic	condition	security	data flow
Use case type	any	any	n/a
Data asset	required	confidentiality (C)	received

Table 6: "Shared data being received" pattern: context.

Problem See Figure 21.

1. Disclosure shared data on the client enabled by vewing caches.
2. Disclosure shared data on the client enabled by shoulder hacking.
3. Disclosure shared data on the network enabled by MITM attacks.
4. Disclosure shared data on the server enabled by user spoofing and injection attacks.

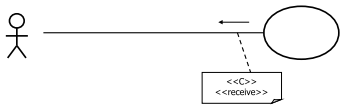


Figure 20: "Shared data being received" pattern: context.

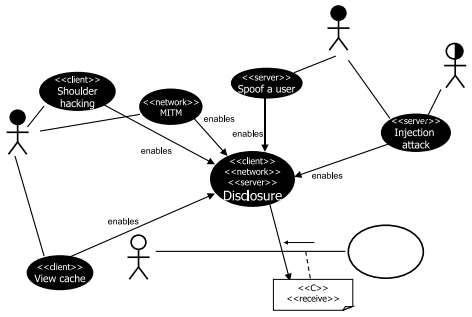


Figure 21: "Shared data being received" pattern: problem.

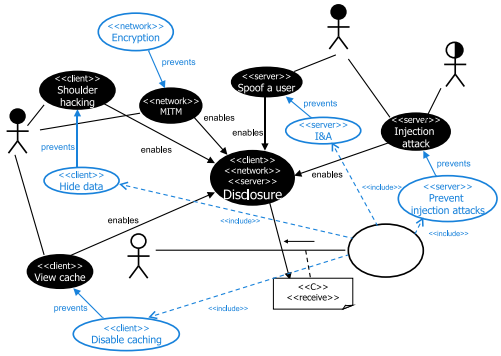


Figure 22: "Shared data being received" pattern: solution.

Solution See Figure 22.

1. Disabling caches is useful for preventing disclosure data on the client enabled by vewing caches.
2. Hiding the data is useful for disclosure data on the client enabled by shoulder hacking.
3. Encryption such as SSL/TLS is useful for preventing disclosure on the network enabled by MITM attacks.
4. I&A is useful for preventing disclosure on the server enabled by user spoofing.
5. Countermeasures against injection attacks are useful for preventing disclosure on the server.

See Also The following patterns are related to this pattern.

- I&A[6]
- Secure networks (SSL)[6]

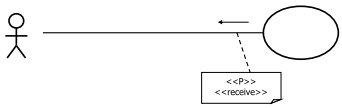


Figure 23: "Private data being received" pattern: context.

A.3 "Private data being received" pattern

Context This pattern can be applied to the program which private data assets are sent from use cases to actors.

Table A.3 and Figure 23 shows the context. "Private" data means that its access is allowed to the owner only. "P" stereotype in Figure 23 represents that the data is required "private" level of confidentiality.

Characteristic	condition	security	data flow
Use case type	any	any	n/a
Data asset	required	privacy (P)	received

Table 7: "Private data being received" pattern: context.

Problem See Figure 24.

1. Disclosure private data on the client enabled by vewing caches.
2. Disclosure private data on the client enabled by shoulder hacking.
3. Disclosure private data on the network enabled by MITM attacks.
4. Disclosure private data on the server enabled by user spoofing and injection attacks.
5. Disclosure private data on the client enabled by inappropriate access control.

Solution See Figure 25.

1. Disabling caches is useful for preventing disclosure data on the client enabled by vewing caches.
2. Hiding the data is useful for disclosure data on the client enabled by shoulder hacking.

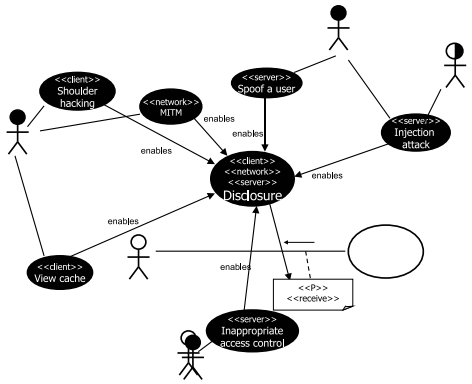


Figure 24: "Private data being received" pattern: problem.

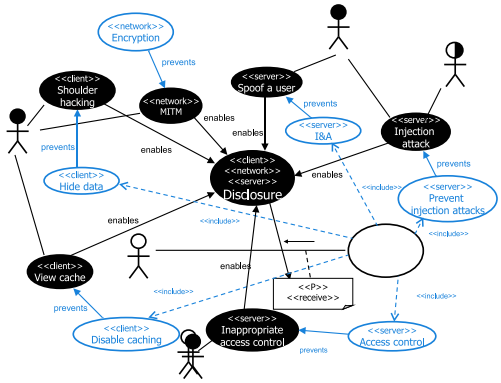


Figure 25: "Private data being received" pattern: solution.

3. Encryption such as SSL/TLS is useful for preventing disclosure on the network enabled by MITM attacks.
4. I&A is useful for preventing disclosure on the server enabled by user spoofing.
5. Countermeasures against injection attacks are useful for preventing disclosure on the server.
6. Appropriate access control is useful for preventing unintended data accesses on the server.

See Also The following patterns are related to this pattern.

- I&A[6]
- Access Control[6]
- Secure networks (SSL)[6]

A.4 "Integral data being received" pattern

Context This pattern can be applied to a program where some data assets which require integrity are sent from use cases to actors. Table A.4 and Figure A.4 shows the context.

Problem See Figure 27 and 28.

1. Tampering data on the network enabled by MITM attacks.
2. Tampering data on the server enabled by injection attacks and user spoofing.

Characteristic	condition	security	data flow
Use case type	any	any	n/a
Data asset	required	integrity	received

Table 8: "Integral data being received" pattern: context.

Solution See Figure 29 and 30.

1. Countermeasures against injection attacks, XSS and CSRF are useful for preventing tampering on the server.
2. Encryption such as SSL/TLS is useful for preventing MITM attacks which enables tampering on the network.

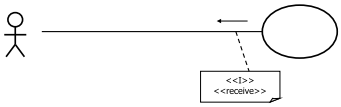


Figure 26: "Integral data being received" pattern: context.

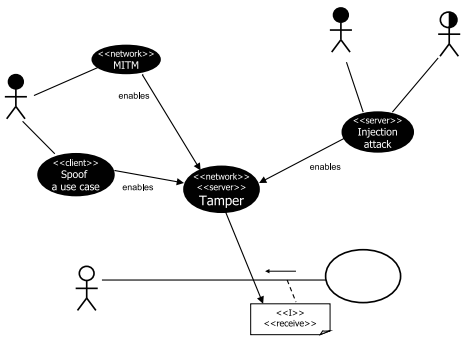


Figure 27: "Integral data being received" pattern: problem (1).

A.5 "ID/Password authentication" pattern

Context This pattern can be applied to the program where ID/password authentication has been introduced. ID/password authentication is the major authentication method in the web application domain. Application of this pattern may be examined after the introduction of I&A is determined with other patterns,
Table A.5 and Figure 31 shows the context.

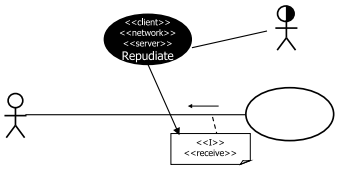


Figure 28: "Integral data being received" pattern: problem (2).

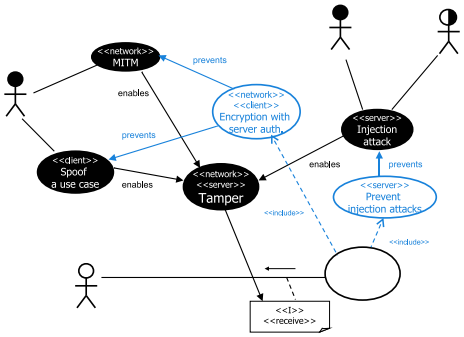


Figure 29: "Integral data being received" pattern: solution (1).

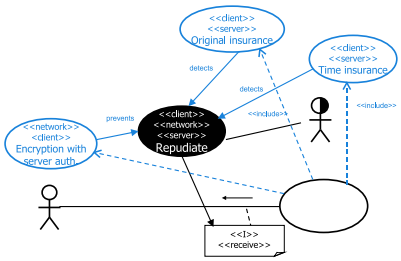


Figure 30: "Integral data being received" pattern: solution (2).

Problem See Figure 32. Data Asset "password" has been newly added to the program. Therefore additional threats may be identified for the new assets. The new identified threats are:

1. Brute force attacks on a password which enables user spoofing.
2. Dictionary attacks on a password which enables user spoofing.
3. Disclosure of a password on the client and the network.
4. Disclosure of a stored password.

Solution See Figure 33.

1. Hiding passwords is useful for preventing disclosure on the client.
2. Encryption of communication is useful for preventing disclosure of passwords on the network.
3. Strong password specification which rejects passwords with short length and a small repertoire of characters is useful for preventing brute force attacks.
4. Rejecting passwords which are easy to guess such as dictionary words, phone numbers and birth dates is

Characteristic	condition	security	data flow
Use case type	any	any	n/a
Data asset	any	any	any
	use ID/password		
Other	authentication	n/a	n/a

Table 9: "ID/Password authentication" pattern: context.

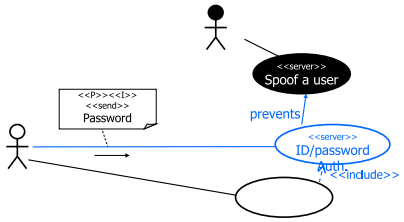


Figure 31: "ID/Password authentication" pattern: context.

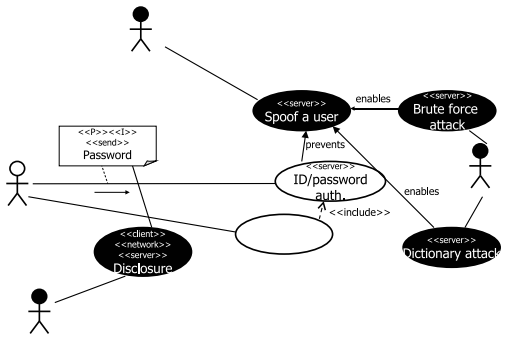


Figure 32: "ID/Password authentication" pattern: problem.

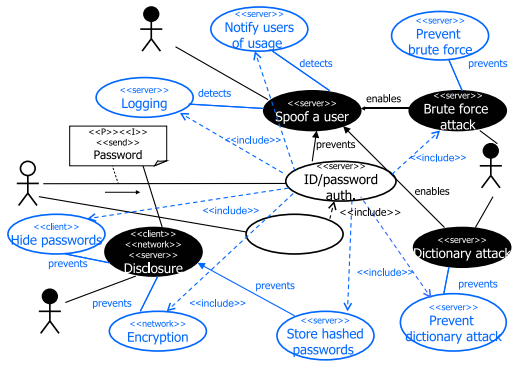


Figure 33: "ID/Password authentication" pattern: solution.

useful for preventing dictionary attacks and guessing passwords.

See Also The following patterns are related to this pattern.

- Password design and use[6]

A.6 "Session management" pattern

Context This pattern can be applied to the program where session management has been introduced. Session management is usually used for maintaining web session data, especially for authenticated users. Therefore application of this pattern may be examined after the introduction of I&A is determined with other patterns, Table A.6 and Figure 34 shows the context.

Characteristic	condition	security	data flow
Use case type	any	any	n/a
Data asset	any	any	any
Other	use session management	n/a	n/a

Table 10: "Session management" pattern: context.

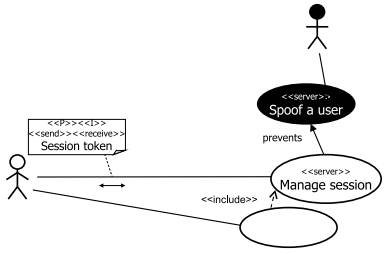


Figure 34: "Session management" pattern: context.

Problem See Figure 35. By introducing session management, session token is identified as a new data asset. Therefore analysts have to consider threats for the new asset. The new threats for a session token are:

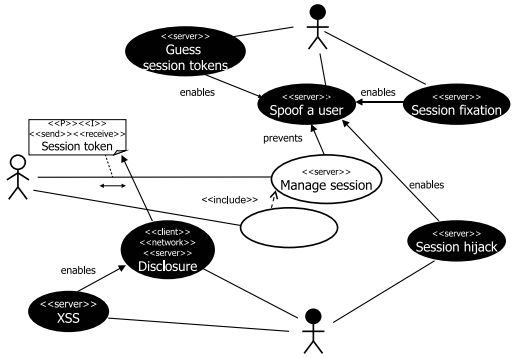


Figure 35: "Session management" pattern: problem.

1. Guessing a session token which enables user spoofing.
2. Session fixation attacks which enable user spoofing.
3. Session hijack attacks which enable user spoofing.
4. Disclosure session tokens on the client.
5. XSS which enables disclosure session tokens on the client.
6. Disclosure session tokens on the network with MITM attacks.

Solution See Figure 36.

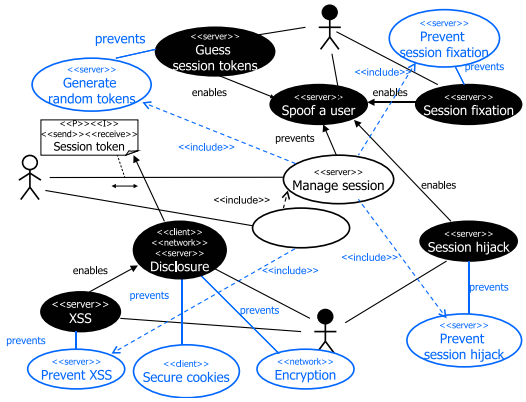


Figure 36: "Session management" pattern: solution.

1. Strong (random) session token generation which makes tokens difficult to guess.
2. Countermeasures which prevent session fixation.
3. Countermeasures which prevent session hijack.
4. Secure cookie management such as setting secure attribute, proper domain and expiration date are useful for preventing disclosure session tokens on the client.
5. Countermeasures against XSS useful for preventing disclosure on the client.
6. Encryption of communication useful for preventing disclosure session tokens on the network.