セキュリティ要件を充足するソフトウェア開発方式の検討

大久保 隆夫[†‡] 中山 裕子[†] 綿口 吉郎[†] 田中 英彦[‡]
†(株)富士通研究所 〒211-8588 川崎市中原区上小田中 4-1-1
‡情報セキュリティ大学院大学 〒221-0835 神奈川県横浜市神奈川区鶴屋町 2-14-1
E-Mail: †{okubo, nakayama.yuko, wataguchi}@jp.fujitsu.com, {dgs063104, tanaka}@iisec.ac.jp

あらまし 近年、ソフトウェアセキュリティの重要性が高まる一方で、ソフトウェア開発の現場においては、セキュリティ品質確保のために必要な作業の体系化が不十分であるのが現状である。本稿では、ソフトウェアの実装、テストと、セキュリティ要件の維持との間の関係を明確にすることを目的とし、その解決手段として、セキュリティの実装手段にある程度の制約をかけることにより、セキュリティ要件の確認が容易になることを示す。また、その関係を利用して、必要なセキュリティ作業を抽出する方式について提案する。さらに、上記の関係を利用して、セキュリティ要件を満たすために必要な作業のコスト(セキュリティ要員の必要性、費用、工数)を、開発の早期において容易に見積もる開発支援システムを構築するための方式について提案する。

キーワード ソフトウェア 開発 セキュリティ要件 実装手段 見積もり 規約 テスト

A Study on Software Development Method which Fulfills Specified Security Requirements

Takao OKUBO[†‡] Yuko NAKAYAMA[†] Yoshirou WATAGUCHI[†] Hidehiko TANAKA[‡]
† Fujitsu Laboratories ltd.
4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki Kanagawa 211-8588 Japan
‡ Institute of Information Security
2-14-1 Turuya-cho, Kanagawa-ku Yokohama, Kanagawa, 221-0835 Japan
E-Mail: †{okubo, nakayama.yuko, wataguchi}@jp.fujitsu.com, {dgs063104, tanaka}@iisec.ac.jp

Abstract There is no sufficient systemized method of secure software development in spite of growth of importance of software security. Project managers, architects, programmers, and testers do not know how to implement the required security, or how to verify it. In this paper, we clarify the correlation between the easiness of verification of implementation rule compliance, and quantity of work to meet the security requirements. Moreover, we propose a secure software development environment system on the basis of these correlation. This system gives project managers quotation of cost, man-hour and security experts at the early phase of the software development life-cycle, by offering choices of the implementing methods. Our system also has the feature generating coding conventions and testing specification document which provides guidelines at the every part of the software development life-cycle to meet the security requirements.

Key words software development, security requirement, quotation, convention, testing

1. はじめに

ソフトウェアの脆弱性を原因とするセキュリティインシデントが多発し、ソフトウェアセキュリティの重要性が叫ばれているにも関わらず、

ソフトウェア製品やシステムの開発現場には、セキュリティ品質向上のための取り組みはあまり浸透していない。その一因は、ソフトウェア開発においてセキュリティ品質を実現、保証する手法が明確になっていないことにある。セキュリティ知識に乏しい開発現場では、セキュリティ確保のために何をどれだけ行えばよいかが分からない。また、プロジェクト管理者にとっては、セキュリティ品質を確保するためのコストが見えにくく、完成したシステムの品質を評価できないため、セキュリティ投資もしにくい状況にある。

著者らは、これらの問題を解決するため、ソフトウェア開発のライフサイクル全体を通してセキュリティの品質を確保し、検証するための技術について研究を行っている。本稿では、その試みの一つとして、あらかじめセキュリティ要件が与えられた場合に、その要件を満たすソフトウェア開発方式について提案する。

2. 現状の課題と関連技術

ソフトウェア開発、特に大規模ソフトウェアの開発現場では、[Roy70]で提唱されたウォーターフォールモデルによる開発体系が用いられ、「分析工程で要件を確定」「設計工程で要件を満たす機能を設計し」「実装工程で機能を実装」「テスト工程で要件、設計通りに機能が実装されているかを確認する」という手順を通じて品質を確保、保証する手法が確立している。

しかし、セキュリティに関する品質の確保、保証という点では、従来のソフトウェア開発体系は十分ではない。ソフトウェアにおける一般的な機能の設計仕様は UML 等のような形式で明確に定義が可能である。 しかし、セキュリティの仕様は UML 等で記述可能なものとは性質が異なる[1]。[PM04] 、[Tom03]で指摘されているように、セキュリティ脆弱性は、ソフトウェアにおける一般的なバグとは異なり、本来の仕様を満たしていても、それ以外に意図しない挙動をする場合がある。それらは「仕様通りか」の確認を行うテストでは検出できないし、「意図しない挙動をしない」という設計仕様の記述方法も従来の手法では困難である。

また、仮にセキュリティ要件を満たすような設計を行ったとしても、実装の段階で脆弱性が混入してしまう場合がある。脆弱性の代表例であるバッファオーバーフロー脆弱性の多くは、この実装段階に原因がある。

セキュリティに必要な機能を設計、実装する方式としては、ISO/IEC 15408 として標準化された Common Criteria(CC、[CC05])がある。CC は、システムに必要なセキュリティ機能を仕様として整備するための体系としては有用であるが、従来のソフトウェア開発手法に基づいた機能的側面に重点が置かれているため、上記のような実装段階の脆弱性の防止に対して有効とはいえない。

現在セキュリティの品質検証の主流となっているのは、第三者によるブラックボックステスト[McG98]、[Wei95])である。一方で、テスト工程での検証に頼る方法では手戻りのコストが大きいという問題がある。そのため、上流でのセキュリティ対処の必要性が指摘されており、[HDL04]をはじめ、いくつかの手法が提案されている。しかし、ソフトウェア開発の現場にこれらの手法はなかなか浸透していない。

このように、上記に挙げた研究、技術が存在するにも関わらず、それらが「銀の弾丸」[2]となりえていないのは、開発の各工程間の密接な関連性を十分活用している研究が少ないためであると考えられる。本質的な問題解決のためには、セキュリティに関わる各工程間の関連性を明らかにすることが必要となる。

3. 提案方式とその特徴

筆者らは Web アプリケーションを中心に、

---

[1] UML の拡張を利用してセキュリティの設計を行う研究([Jür04])もある。

[2] Fred Brooks の言葉で、ソフトウェア工学で用いられる「多くの問題を一発で解決するような手法」の比喩。

システム開発の上流工程からセキュリティのレビューと改善支援を行うとともに、システム開発に必要なセキュリティ作業のノウハウを蓄積してきた。筆者らはこのノウハウを利用して、セキュリティの知識に乏しいシステム開発者に対しても、開発ライフサイクル全体を通してセキュリティ支援、教育を行えるような技術の研究を行っている。

筆者らは分析工程でセキュリティ要件を決定した後の設計以降の工程において、いかに決定した要件が守れるかという点に着目し、(1)実装手段の限定と、(2)実装手段間の相互関係と代替手段の定義が有効と考えた。

(1) 実装手段の限定

実装工程において、あるセキュリティ要件を実現する手段は、プログラミング言語やフレームワーク、ライブラリを一意に決定したとしても、通常は数多くの手段が存在する。しかし、各実装手段のセキュリティ要件に対する充足度はさまざまである。その実装手段をプログラマの選択に委ねるとすると、テスト段階ではどの実装手段が採用されているかは不明なので、セキュリティ要件充足の確認は非常に困難になる。

しかし、コーディング規約などの制約によって実装手段を限定すれば、制約遵守(または違反)を確認することで、セキュリティ要件の充足の確認をその実施手段の範囲に限定できる。また、その実装手段の実施によってセキュリティ要件を充足できることが分かっている時、実装手段の遵守の確認の精度が、セキュリティ要件の充足度(保証度)として利用できる。

上記の性質を利用して、実装手段の確認の容易性から作業量や作業内容を導出することにより、セキュリティ要件に必要な作業の見積もりを容易にできる。

(2) 実装手段間の相互関係と代替手段

(1)で挙げた各実装手段を同一のセキュリティ要件に対する充足という関係でみると、重複や手段間の依存関係が存在する。そのため実装手段により作業の見積もりを行う際には、重複の排除や不足手段の補完を行う必要がある。筆者らは各実装手段間の相互関係について、計算機で処理が可能なように以下の関係に整理した。

(a) AND 関係
　要件の充足には、他の手段との併用が必須

(b) OR 関係
　他の手段を含むいずれか一つの手段で要件の充足が可能

(c) XOR 関係
　いずれか一つの手段で要件の充足が可能でかつ、他の手段との併用が不可能

また、実装手段と区別して、代替手段も定義した。代替手段は実装手段が実施されない場合、セキュリティ要件を充足するために必要な次善の手段として定義されるもので、実装手段とは OR の関係にあるが、選択候補として提示する際には実装手段の方が優先される。

4. 提案方式

図 1 に提案方式を用いたシステムの構成と流れを示す。

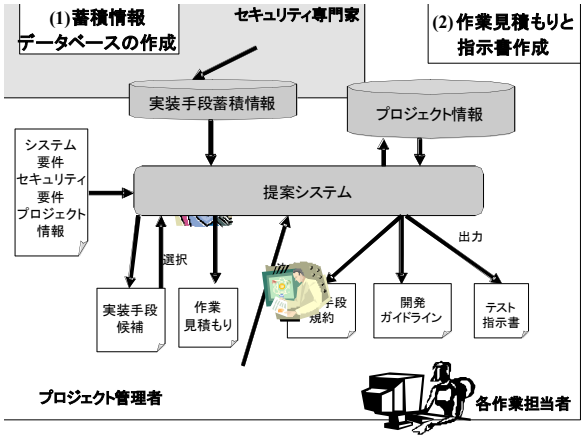

図 1：提案方式のシステム構成と流れ

提案方式は、大別して二つの部分から構成される：

(1) セキュリティ専門家による、実装手段蓄積情報データベースの作成

(2) プロジェクト管理者による作業見積もりと、各開発担当者向け指示書の生成

以下に、これらの詳細を述べる。

(1) セキュリティ専門家による、実装手段蓄積情報データベースの作成

セキュリティ専門家が、各システム要件およびセキュリティ要件ごとに、実装手段の候補一覧を作成する。これらの候補は、実施を強制することでセキュリティ要件を満たすことが可能なものとする。その際、前節で述べた相関関係を利用するため、下記のデータの入力、定義を行わせる。

(a) 実装手段間の AND/OR/XOR 関係

(b) 代替手段候補の定義および AND/OR/XOR 関係

(b) 実装手段の限定方法
(特定の部品や実装手順の利用を強制する、特定の部品や実装手段の利用を禁止するなど。これらのパターン化により、下記のセキュリティ知識の要否、精度、費用、工数等の計算を効率化できる)

(c) 実装手段に対する三つの観点(制約の実施、実装箇所の検出、制約違反の検出)ごとの、セキュリティ知識の要否、精度、調達費用、実装手段の選択により新規に追加となる工数等のパラメータの概算値
(データの詳細を表 1 に示す)。

(A)	制約または代替案実施に必要な作業情報	(A-1)	セキュリティ専門家による追加工数(、kstep あたり人日)
		(A-2)	非専門家による追加工数
		(A-3)	偽陽性(%)
		(A-4)	偽陰性(%)
		(A-5)	ツールの利用可能性
		(A-6)	調達費用(円)
		(A-7)	セキュリティ知識の要否
(B)	実装箇所の検出に必要な作業情報	(A)と同様の構造	
(C)	制約の違反検出に必要な作業情報	(A)と同様の構造	

表 1 実装手段情報のデータ構造(詳細)

(2) プロジェクト管理者による作業見積もりと、各開発担当者向け指示書の生成

各アプリケーションの開発プロジェクトにおいて、プロジェクト管理者が要件とプロジェクト情報を入力し、入力情報と(1)の蓄積情報をもとに提案方式のシステムが作業情報の見積もりと指示書の生成を行う。手順は以下の通り：

1. プロジェクト管理者は、システム要件、セキュリティ要件および、各プロジェクトの情報をシステムに入力する。
プロジェクト情報として入力するデータの例を表 2 に示す。

(PI-1)	セキュリティ専門家要員数
(PI-2)	非専門家の要員数
(PI-3)	プログラム規模
(PI-4)	作業者の作業単価

表 2 入力するプロジェクト情報

2. 提案システムは実装手段情報データベースから要件に該当する実装手段群を抽出し、必要な実装手段の選択肢をプロジェクト管理者に与える。
3. ここでプロジェクト管理者が実装手段を選択すると、提案システムは(1)で定義した情報とプロジェクト情報から、セキュリティのために新規に発生する作業の情報およびコストを導出し、プロジェクト管理者に提示する。作業情報の導出の例を表 3 に示す。

	作業見積もり情報	見積もりの導出方法
(PO-1)	セキュリティ保証レベル	100−[(A−4),(B−4),(C−4),の最大値]
		(((A-1)+(B-1)+(C-1))×(PI-1)+((A-2)+(B-2)+(C-2))×(PI-2)) ×(PI-3)/((PI-1)+(PI-2))
(PO-3)	セキュリティに必要な追加費用	(A-6)+(B-6)+(C-6)+(PO-2)×(PI-4)
(PO-4)	セキュリティに必要な追加期間	(PO-2)/((PI-1)+(PI-2))

表 3 作業見積もりと導出方法

4. プロジェクト管理者は実装/代替手段の組み合わせを変えて繰り返し見積もりを行い、最終的にプロジェクトの要件に合致する手段を確定する。
5. システムは蓄積情報データベースの実

装手段情報から規約ドキュメント、開発ガイドライン、テスト指示書の雛形を生成し出力する。

5. 提案方式の机上検討

筆者らは提案した開発方式のうち、(1)セキュリティ専門家による実装手段案の作成と、(2)各開発プロジェクトにおける作業情報の取得の有効性について、Web アプリケーションのセキュリティ要件を対象に、机上検討を行った。検討の目的は以下を確認することである。

- 実装手段に関する情報をセキュリティ専門家が洗いだし、本提案方式のデータ構造に従って定義できる
- 各プロジェクト管理者に対して、規約として与えるべき実装手段の候補が提示され、かつ全体として必要な作業の質および量、コストが判断材料として明示できる

Web アプリケーションのセキュリティ要件から SQL インジェクションを例として選び、検証を行った。システム要件はすべて Java+Servlet+JSP とした。なお、要件に対して挙げる実装手段(脆弱性への対策)の十分性、網羅性などは、本稿では議論の対象としない。

Java における SQL インジェクション対策として、セキュリティ専門家が実装手段を 2 つ候補として抽出したと仮定し、本提案方式の検証を行う。

(1) セキュリティ専門家による実装手段蓄積情報の作成

まず、実装手段間の関係を定義し、対応する代替手段を定義する。代替手段を含めた実装手段の候補例を表 4 に示す。

(実装 1)	SQL 文の実行には java.sql.PreparedStatement のみを使用する
(実装 2)	SQL 実行の直前で危険な文字をエスケープする
(代替 1)	ブラックボックス形式による第三者監査によってインジェクションが起きないことを確認する。

表 4 実装手段候補例

セキュリティ専門家としては、表 4 の中では、(実装 1)を推奨したいものの、この時点ではその根拠が提示できない状態である。

次に、上記手段候補の詳細情報を定義する。定義した例を表 5 に示す。

	情報	(実装 1)	(実装 2)	(代替 1)
(A-1)	セキュリティ専門家による工数	0	0.1	0.1
(A-2)	非専門家による工数	0	0.2	2
(A-3)	偽陽性(%)			50
(A-4)	偽陰性(%)			50
(A-5)	ツールの利用可能性	手作業	手作業	ツール+手作業
(A-6)	費用(円)	0	0	100 万
(A-7)	セキュリティ知識の要否	不要	要	要
(B)		(略)	(略)	(略)
(C-1)		0	0.2	
(C-2)		0	0.4	
(C-3)		0	30	
(C-4)		0	30	
(C-5)		ツール	手作業	
(C-6)		0	0	
(C-7)		不要	要	

表 5 実装手段情報の定義例

表 5 の結果を見ると、(実装 1)の違反の検出作業において(C-5)はツールが利用できるため、違反の検出が最も容易であることがわかる。

(2) プロジェクト管理者による作業見積もり

ここではまず、プロジェクト情報を入力する。

(PI-1)	セキュリティ専門家要員数(人)	0
(PI-2)	非専門家の要員数(人)	10
(PI-3)	プログラム規模(kstep)	50
(PI-4)	作業者の作業単価(円/日)	3 万

表 6 プロジェクト情報入力例

入力したプロジェクト情報と実装手段情報により作業情報を導出した結果を表 7 に示す。

	(実装 1)	(実装 2)	選択なし
セキュリティ保証レベル	100	70	50
必要工数(人日)	0	35	100
必要な費用(円)	0	105 万	400 万
必要な期間(日)	0	3.5 日	10 日
セキュリティ専門家の要否	不要	要	要

表 7 作業情報の導出

プロジェクト管理者は表 7 で得られた情報と自己のプロジェクトの予算や納期、セキュリティ専門家要員などの情報を比較し、プロジェク

トの許容する範囲で実装手段を選択できる。表 7 の場合では、セキュリティ専門家として推奨したい実装手段(実装 1)が、本提案方式の試算により、プロジェクト管理者にとっても最善(コストが最小、かつ最も高い保証レベル)になることを提示できた。

6. まとめ

本稿では、実装方式の限定が、セキュリティ要件の維持、確認に必要な作業に影響を与えることに着目した。また、その関連に基づき、実装方式を限定することにより、セキュリティ要件を満たすために必要な作業の情報を開発の早期において得られる開発方式の提案を行った。また、Web アプリケーションにおける一般的なセキュリティ要件に対して提案方式の机上検証を行い、セキュリティ上も妥当な結果が導出されることを確認した。今回、机上検証に用いた作業のコスト試算のためのパラメータは、あくまで概算でしかなく、試算の精度には向上の余地がある。ただし、本稿の主な目的は試算精度の向上よりも、むしろ実装手段を限定することで、セキュリティ知識の希薄なプロジェクト管理者にも、「セキュリティ品質と作業コスト」という、明確な指標を提示し得ることを示すことにあった。その目標については、一例ではあるが、今回の検証で確認できたと考える。

紙面の都合上割愛したが、他の机上検証の結果、ツールが未整備であるか、技術的に困難などの理由でツールのみでは実装手段の遵守、違反の確認が難しいものも多くあることがわかった(Session Fixation 対策の認証時のセッション切り替えや、Cross site Request Forgeries に対する対策など)。これらの対策が確実に実施されていることを保証するためには、確認手段の自動化技術についても、さらに検討する必要があると考える。

筆者らは、今後、提案方式を実現するシステムの試作を行い、実際のプログラム開発に適用して、その有効性を検証する予定である。

また、本稿の提案方式では、実装手段情報の作成時において、要件からそれを満たすような実装手段候補を生成する方法や、実装手段候補がセキュリティ要件を満たすものかどうかの検証方法が、いまだ人手の作業に依存している。今後は、これらの作業に対する技術的な支援方法についても研究していく予定である。

文献

[CC05] Common Criteria for Information Technology Security Evaluation v2.3, http://www.commoncriteriaportal.org/public/developer/index.php?menu=2, 2005.

[HDL04] M. Howard and D. LeBlanc, “Writing Secure Code Second Edition, Microsoft Press, 2002.

[McG98]G. McGraw, Testing for security during development: Why we should scrap penetrate-and-patch”, IEEE Aerospace and Electronic Systems, 1998.

[PM04] B. Potter and G. McGraw, “Software security testing”, Security & Privacy Magazine, IEEE Volume: 2 Issue: 5 Sept.-Oct. 2004 pp. 81-85, 2004.

[Ray03] Amab Ray: “Security check: a formal yet practical framework for secure software architecture”, Proceedings of the 2003 workshop on New security paradigms, pp. 59-65, 2003.

[Roy70] W. W. Royce, “Managing the Development of Large Software Systems”, Proceedings of the IEEE WESCON, IEEE Press & Proceedings of the Ninth International Conference on Software Engineering, IEEE Press, 1970.

[Tom03] H. H. Thompson., “Why security testing is hard”, Security & Privacy Magazine, IEEE Volume: 1 Issue: 4 July-Aug. 2003 pp. 83- 86, 2003.

[Wei95]C. Weissman. “Penetration Testing”, Information security: an integrated collection of essays, IEEE Computer Society Press, Silver Springs, MD, 1995

.[Jür04] Jan Jürjens, “Secure Systems

Development with UML", Springer, 2004.