社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
KBSE2011-75 (2012-3)

ソフトウェアセキュリティ知識体系化に関する研究動向

櫨山淳雄

東京学芸大学 教育学部 技術・情報科学講座 情報科学分野

〒184-8501 東京都小金井市貫井北町 4-1-1

E-mail: hazeyama@u-gakugei.ac.jp

あらまし　インターネット上でのサービスの増大に伴いソフトウェアセキュリティの重要性が認識されている。これまでに数多くのソフトウェアセキュリティに関する技術開発が行われてきた。しかし、それらをどのように組み合わせて利用すればよいのかはまだ十分に明らかにされていない。そこで、本稿ではセキュアなソフトウェアを開発するための知識の概念モデルを提案し、各知識に関して技術開発されてきた実例の概要を紹介するとともに、知識間の関連を明らかにしている研究事例を紹介する。そして今後の方向性について議論する。

キーワード　ソフトウェアセキュリティ，知識体系，サーベイ，攻撃パターン，セキュリティパターン，原則，ガイドライン

Survey on a Body of Knowledge regarding Software Security

Atsuo HAZEYAMA

Department of Information Science, Tokyo Gakugei University

4-1-1 Nukuikita-machi, Koganei-shi, Tokyo, 184-8501 Japan

E-mail:　hazeyama@u-gakugei.ac.jp

Abstract Importance for software security technologies has been recognized according to increase of services on the internet. A number of research and development regarding software security have been conducted thus far. However, it is not clarified how those technologies are combined in order to utilize them effectively. This paper proposes a conceptual model for a body of knowledge regarding software security. Then this paper introduces an overview of technologies developed for each knowledge and current status of research that has clarified the relationships between pieces of knowledge. Finally the author discusses a future direction of this field.

Keyword Software Security, Body of Knowledge, Survey, Attack Pattern, Security Pattern, Principle, Guideline

1. はじめに

インターネット上でのサービスは増大の一途を辿っている。それに伴い、コンピュータセキュリティの重要性が高まってきている。特に近年、多くのサービスがソフトウェアで実現され、その複雑さが増大しているため、ネットワークセキュリティ技術(アクセス制御や暗号技術など)のみでなくソフトウェアセキュリティ技術の重要性が認識されてきた[18]。ソフトウェアセキュリティはソフトウェア開発過程全体でセキュリティを扱い[18]、セキュアなソフトウェアを開発することを目指す。近年ソフトウェアセキュリティのための方法論、開発プロセス、パターン等が活発に開発されてきた。しかし、それらの間の関連が整理されておらず、利用に困難を期している[1], [41]。

そこで、本論文では、これまでに提案されてきたソフトウェアセキュリティに関する知識の体系化を目指し、概念とその関連をモデルとして表現することを試みる。そして、これらの概念を具現化した研究開発事例や概念間の関連を明らかにした研究開発事例を紹介する。そして、今後の方向性について考察する。

2. 関連研究

ソフトウェアセキュリティ技術に関する研究動向を体系的にまとめた文献に、吉岡らによる文献[42]、Barnum と McGraw による文献[2]などがある。

吉岡らは、セキュアなソフトウェアを開発するための技術を要求、設計、実装(テストを含む)の工程ごとに紹介している[42]。セキュリティ要求分析技術として、KAOS、i*、Secure Tropos を紹介している。セキュリティ設計技術として、脅威分析手法、UMLsec、セキュリティパターン、検証手法を紹介している。実装

のための技術として、セキュリティ機能テスト、脆弱性の分類と対策、ツールを活用したテスト、プログラムの(静的/動的)検証、システム構成の確認、メトリクスを紹介している。

Barnum と McGraw は、ソフトウェアセキュリティのための知識を、7 つの知識カタログ(principle, guideline, rule, attack pattern, vulnerability, exploit, historical risk)とその関連としてモデル化している[2]。しかし、このモデルではソフトウェアセキュリティにおいて重要な要素であるプロセス、標準、方法論やセキュリティパターンといった概念を扱っていない。それとともに概念間の関連もいくつか欠落している。そのため、本研究は Barnum と McGraw のモデルをベースとし、これを拡張し、拡張したモデルをベースに、各知識に対する現状をまとめる。

3. 概念モデルの提案

本節では、Barnum と McGraw の知識とその関連のモデルを説明する。そして、このモデルをベースに拡張した筆者による提案モデル[17]を説明する。

Barnum と McGraw は、文献[2]で 7 つの知識カタログを以下のように定義している：

- Principle：経験から導き出された一般的なセキュリティ知の表明である。
- Attack pattern：多くの悪用から得られた攻撃の手口をまとめたものである。
- Guideline：意味的なレベルで、行うべきことや避けるべきことをまとめたものである。
- Rule：構文レベルで、行うべきことや避けるべきことをまとめたものである。
- Vulnerability：攻撃者が利用する可能性のあるソフトウェアの欠陥や弱点である。
- Exploit：特定の脆弱性を突いたコンピュータシステムへの攻撃である。
- Historical risk：実際のソフトウェア開発活動で識別されたリスクを意味する。

筆者は上述の 7 つに、次の 3 つを加えた：

- Process・Methodology・Standard：セキュアなソフトウェアを開発することに関わるプロセス、方法論、標準である。
- Component：プロセス、方法論、標準を構成する要素を意味する。例えば CLASP プロセスでは Activity、Sub-activity が規定されている。
- Security Pattern：セキュリティに関し繰り返し生じる問題に対する解決策を与えるものである(例えば、[26], [40])。

次に関連について述べる。Barnum と McGraw は 7 つの知識カタログ間に 9 つの関連を規定している：Attack pattern と Exploit の間、Exploit と Vulnerability の間、Exploit と Historical risk の間、Vulnerability と Historical risk の間、Principle と Guideline の間、Principle と Rule の間、Guideline と Rule の間、Guideline と Vulnerability の間、Rule と Vulnerability の間。

Buyens らは、CLASP プロセスが Principle に重きを置いており、プロセスを構成する Activity が Principle と関連を有していると述べている[7]。また Stoneburner らは、Principle 間に関連があると述べている[32]。Wassermann と Cheng は、既存のセキュリティパターンを適切に表現するためのテンプレートを提案している[36]。その中で関連として Security pattern 間、Security pattern と Principle との間の関連を言及している。さらに Attack pattern のディクショナリに CAPEC (Common Attack Pattern Enumeration and Classification)[8]がある。その構造は、Primary Schema Elements と Supporting Schema Elements に大別されている[4]。その中に他の知識との関連を表す項目として、Primary Schema Elements に Related Weaknesses, Related Vulnerabilities が、Supporting Schema Elements に Relevant Security Patterns, Related Security Principles, Related Guidelines が記されている。さらに Primary Schema Elements に Solutions and Mitigations (解決策と軽減策)があり、ここに記述される内容から対応する Pattern や Guideline を探す手掛かりが見つけられる可能性もある。親子関連などの Attack Pattern 間の関連も表現している。

以上から、Process の構成要素である Component と Principle 間、Principle 間、Security pattern 間、Security pattern と Principle の間、Attack pattern と Principle の間、Attack pattern と Security pattern の間、Attack pattern 間、Attack pattern と Vulnerability の間、Attack pattern と Guideline の間に関連を概念モデルに設定する。

図 1 は、提案する概念モデルを UML のクラス図として表したものである。

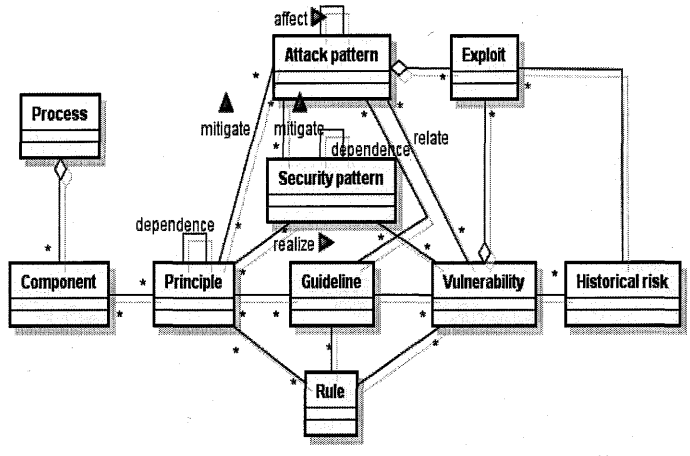

図 1．ソフトウェアセキュリティに関する知識体系の概念モデル

4. 各知識の実体に関する研究動向

本節では、前節であげた概念のいくつかに対して研究開発されている事例を紹介する。

4.1 Attack pattern

Attack pattern (攻撃パターン)[3]のディクショナリとして、Common Attack Pattern Enumeration and Classification (CAPEC)がある[8]。その構造は文献[4]に詳述されているが、主要な項目は以下のとおりである：Pattern name and classification (パターン名と分類), Attack prerequisites (攻撃の前提条件), Description (記述), Targeted vulnerabilities or weaknesses (攻撃対象の脆弱性), Method of attack (攻撃方法), Attacker goal (攻

撃者の目標), Attacker skill level required (攻撃者に求められるスキルレベル), Resources required (攻撃に必要な資源), Blocking solutions (攻撃への防御策), Context description (背景に関する記述), References (参考文献)。最新版は Release1.6 で 460 の攻撃パターンが登録されている。

4.2 Vulnerability

脆弱性のディクショナリに Common Vulnerabilities and Exposures (CVE)がある[11]。情報処理推進機構(IPA)によると、2008 年 12 月末現在 34000 以上の脆弱性が報告されている。また、脆弱性のタイプを識別するために Common Weakness and Exposures (CWE)が提供されている[12]。IPA によると、2010 年 6 月時点で 777 の脆弱性の種類が分類されている[13]。

4.3 Process・Methodology・Standard

セキュアなソフトウェアを開発するためのプロセスに、CLASP (Comprehensive Lightweight Application Security Process)[23]、Security Quality Requirements Engineering (SQUARE)[19], [31]、McGraw によるベストプラクティス[18]がある。また方法論として、Sindre と Opdahl によるセキュリティ要求獲得手法[29], [30]などが知られている。標準として Common Criteria[10], [21]がある。

(1) CLASP

CLASP プロセスは、7 つのベストプラクティス("Institute awareness programs (アウェアネスプログラムの確立)," "Perform application assessments (アプリケーションアセスメントの実施)," "Capture security requirements (セキュリティ要求の獲得)," "Implement secure development practice (セキュリティ開発の実践)," "Build vulnerability remediation procedures (脆弱性改善手順の構築)," "Define and monitor metrics (メトリクスの定義と監視)," "Publish operational security guidelines (運用に関するセキュリティガイドラインの発行)")の下に、24 のアクティビティが構成されている。アクティビティはさらにサブアクティビティから構成される。

(2) SQUARE

SQUARE はセキュリティ要求獲得のためのプロセスであり、9 ステップから構成される：1. 定義をステークホルダで合意する、2. 資産とセキュリティゴールを認識する、3. 中間成果物を作成する、4. リスクを評価する、5. 要求獲得技術を選択する、6. セキュリティ要求を獲得する、7. 要求を分類する、8. 要求に優先度をつける、9. 要求を検査する。

(3) McGraw によるベストプラクティス

McGraw は、ソフトウェアセキュリティのベストプラクティスを紹介している。それらは Abuse cases (アブユースケース)、Security requirements (セキュリティ要求)、Risk analysis (リスク分析)、External review (外部レビュー)、Risk-based security tests (リスクに基づくセキュリティテスト)、Static analysis tool (静的解析ツール)、Penetration testing (ペネトレーションテスト)、Security breaks (セキュリティ破壊)である。

(4) Sindre と Opdahl のセキュリティ要求獲得手法

Sindre と Opdahl は、5 ステップからなるセキュリティ要求獲得手法を提案している：1. 重要なアセットの識別、2. 各アセットに対して、セキュリティゴールの設定、3. 各セキュリティゴールに対する脅威の識別、4. 脅威によるリスクの識別と分析、5. 脅威に対するセキュリティ要求の決定。

(5) Common Criteria (CC:コモンクライテリア)

IT を用いた製品やシステムが備えるべきセキュリティを評価するための国際標準に ISO/IEC15408 がある。そのもとになっている規格に Common Criteria (CC:コモンクライテリア)がある[10]。評価に際し、開発者は以下のことを行い文書化する[21]：1. 利用者の視点で製品に求められるセキュリティ要求を収集整理する、2. 評価対象の範囲と Evaluation Assurance Level (EAL：評価保証レベル)を定める、3. 保護対象資産に関する脅威モデルの定義・分析・運用環境における対抗策を定義する、4. IT による対抗策を実現するための機能要件を定義する、5. 機能要件の仕様化、6. Security Target (ST:セキュリティターゲット)文書として記述する。

4.4 Principle

Principle について、いくつかの提案がなされている([5], [14], [15], [24], [25], [32], [34])。

Saltzer と Schroeder は、セキュリティ Principle に関する先駆的な研究を行った。8 個の Principles (Open design, Fail-safe defaults, Least privilege, Separation of privilege, Economy of mechanism, Complete mediation, Least common mechanism, Psychological acceptability)を提示している[25]。

Viega と McGraw は、10 個の Principles (Secure the weakest link, Defense in depth, Fail securely, Least privilege, Compartmentalize, Keep it simple, Promote privacy, Hiding secrets is hard, Reluctant to trust, Use your community resources)を示している[34]。

Graff と Wyk は、30 個の Principles を提案している[15]。Stoneburner らは、33 個の Principles を提案している[32]。Daswani らは、9 個の Principles を提案している[14]。Barnum と Gegick は、13 個の Principles を提案している[5]。CLASP は、9 個の Principles を示している[24]。

4.5 Security pattern

セキュリティパターンは Yoder と Barcalow[39]によって初めて提案された[37]。彼らは、7 つのパターン(Single access point, Check point, Roles, Session, Full view with errors, Limited views, Secure access layer)を提案している。各パターンは次の項目が述べられている：Alias (別名), Motivation (動機), Problem (問題), Forces (効力), Solution (解決策), Example (例), Consequences (結果), Related patterns (関連するパターン), Known uses (既知の利用), Non-security known uses (セキュリティ以外の既知の利用)。

その後数多くのセキュリティパターンが提案されてきた。そしてそれらをまとめて分類を行ったり、関連付けを行った文献が報告されている ([37], [16], [40], [36])。

Wiesauer と Sametinger は、セキュリティパターンは数が多いため、どれを使えばよいかわかりにくいという問題提起をしている。そこで彼らはパターンを分類する必要があるとして、攻撃パターンをベースにした分類を提案している[37]。

吉岡らは、開発工程に従いセキュリティパターンを分類し、分析、設計、実装の3つの工程に関わるセキュリティパターンを紹介している[40]。

Hafiz らは、96 個のセキュリティパターンのカタログを構築し、パターン間の関連を構築する方法を示し、全パターン間の関連を構築している[16]。

Wassermann と Cheng は、セキュリティパターンのテンプレートを提案している。テンプレートにはデザインパターンのテンプレートに加え、振舞い、制約、Principle との関連を扱うことを提案している[36]。

4.6 Guideline

アメリカ合衆国の Department of Homeland Security, National Cyber Security Division の Software Assurance 戦略的イニシアティブである BuildSecurityIn (BSI)がガイドラインを提供している[38]。このガイドラインでは人間の振舞いが脆弱性をもたらすということと技術には脆弱性があることを念頭に入れるべきであるという仮定のもと、技術面に関してはさらに 8 項目の分類 ("Follow the Rules Regarding Concurrency Management," "Design Configuration Subsystems Correctly and Distribute Safe Default Configurations," "Carefully Study Other Systems Before Incorporating Them into Your System Through Delegation," "If Emulation of Another System Is Necessary, Ensure that It Is as Correct and Complete as Possible," "Handle All Errors Safely," "Validate All Input as Precisely as Possible," "Use All Security Mechanisms Correctly," "Do Not Allow Your System to Ever Use or Depend on Language Behaviors that Are "Undefined"")を示している。8 項目のうち、"Validate All Inputs as Precisely as Possible (すべての入力に対してできるだけ正確に妥当性検査を行うべき)," "Use All Security Mechanisms Correctly (セキュリティのメカニズムを正しく利用すべき)," "Do Not Allow Your System to Ever Use or Depend on Language Behavior that Are "Undefined" (「未定義」の言語の振舞いを使用したり、それに依存してはならない)"はさらに小項目を示している。

Mozilla プロジェクトは、Web アプリケーションのためのセキュアコーディングガイドラインとして 12 項目 (Authentication, Session Management, Access Control, Input Validation, Output Encoding, Cross Domain, Secure Transmission, Content Security Policy, Logging, Admin Login pages, Uploads, Error Handling)を提供している[9]。

他にも Java 言語のセキュアコーディングガイドラインがいくつか提供されている[20], [22], [28]。

4.7 Rule

前述の BuildSecurityIn は、C/C++を対象としたコーディングルールのカタログを提示している[6]。これらは静的解析ツールの中に組み込まれている。各ルールを説明するための属性が規定されている：タイトル、攻撃カテゴリ、脆弱性カテゴリ、ソフトウェアのコンテキスト、ロケーション、記述、API(関数名、コメント)、攻撃方法、例外の基準、解決策、シグネチャの詳細、コード例(良い例と悪い例)、参考情報、推奨情報、OS、言語。

5. 異なる知識間の関連付けに関する動向

本節では、知識間の関連を扱った事例を紹介する。

5.1 プロセスと他の知識との関連

Buyens らは、CLASP プロセスと Principles の関連について明らかにしている。CLASP プロセスには 85 の Activity があり、それらのうち少なくとも 30 が Principle と関連づいていることを明らかにしている[7]。例えばアクティビティ"Identify, implement, and perform security tests"では、"Defense in depth"との関連性を言及している。

清水と櫨山は、Sindre と Opdahl のセキュリティ要求獲得手法に、セキュリティパターン、STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege)[33]、WASC[35]の 3 つを関連付けた Web アプリケーションのためのセキュリティ要求獲得手法を提案している[27]。

5.2 セキュリティパターンと Principles との関連

Wassermann と Cheng は、セキュリティパターンと Principle との関連を扱うことを提案している[36]。Principle は Viega と McGraw による Principle [34]を対象としている。セキュリティパターンは Yoder と Barcalow による 6 つのパターン (Single access point, Check point, Roles, Session, Full view with errors, Limited views) [39] に Authorization と Multilevel Security Pattern を加えた 8 つのパターンを対象としている。Single access point パターンは"Secure the weakest link," "Keep it simple," "Reluctant to trust,"の 3 つの Principle との関連性を言及している。Check point パターンは"Secure the weakest link," "Defense in depth," "Least privilege," "Reluctance to trust" の 4 つの Principle との関連性を言及している。Roles パターンは"Keep it simple," "Least privilege"の 2 つの Principle との関連性を言及している。Session パターンは"Keep it simple," "Promote privacy" の 2 つの Principle との関連性を言及している。Full view with errors パターンは"Fail securely," "Keep it simple" の 2 つの Principle との関連性を言及している。Limited views パターンは"Defense in depth," "Least privilege," "Promote privacy," "Hiding secrets is hard," "Reluctant to trust" の 5 つの Principle との関連性を言及している。Authorization パターンは "Least privilege," "Compartmentalize," "Promote privacy"の 3 つの Principle との関連性を言及している。Multilevel Security パターンは"Defense in depth," "Least privilege," "Promote privacy," "Hiding secrets is hard," "Reluctant to trust" の 5 つの Principle との関連性を言及している。

5.3 Attack pattern と他の知識との関連

Web アプリケーションにおける代表的な攻撃としてクロスサイトスクリプティングと SQL インジェクションがある。それらは CAPEC のディクショナリ[8]にそれぞれ CAPEC-86、CAPEC-66 として登録されている。ここではこの 2 つの攻撃に関する記述から知識間の関連について記述されている部分を紹介する。

(1) クロスサイトスクリプティング

クロスサイトスクリプティング(CAPEC-86)に関し

て、関連する Weakness として 11 の CWE (CWE-79、CWE-184、CWE-348、CWE-96、CWE-20、CWE-116、CWE-86、CWE-692、CWE-697、CWE-713、CWE-71)が記されている。関連する Vulnerability として 2 つの CVE (CVE-2006-5442、CVE-2006-3918)が記されている。関連するアタックパターンとして 3 つの CAPEC (CAPEC-18、CAPEC-220、CAPEC-107)が記されている。"Solutions and Mitigations"に、設計に関する事項(「クライアントサイドのスクリプトを許可しないブラウザ技術を使用すること」、「厳密の型、文字、エンコードの強制を利用すること」、「リモートのコンテンツに対して http プロキシがサーバサイドでセットアップされている場合、クライアントブラウザはそのデータの出所を知るすべがないので、サーバサイドの開発者は XHR や他の方法でコンテンツのプロキシ化するべきでない」)と、実装に関する事項(「クライアントに配信されるすべてのコンテンツは確実に無害化すること」、「入力の妥当性検査を行うこと」、「出力の妥当性検査を行うこと」、「JavaScript のようなスクリプト言語の実行を不可にすること」、「特定のホストに対してセッショントークンを送る」、「ソフトウェアにパッチをあてること」)が記されている。「入力妥当性検査を行うこと」、「出力の妥当性検査を行うこと」の 2 項目はガイドライン[9]において示されている。

(2) SQL インジェクション

SQL インジェクションに関して、関連する Weakness として 7 つの CWE (CWE-89、CWE-74、CWE-20、CWE-390、CWE-697、CWE-713、CWE-707)が記されている。関連する Vulnerability として 1 つの CVE (CVE-2006-5525)が記されている。関連するアタックパターンとして 5 つの CAPEC (CAPEC-152、CAPEC-352、CAPEC-7、CAPEC-109、CAPEC-110)が記されている。関連する Principle として"Reluctant to trust," "Failing securely," "Defence in depth,"が記されている。関連する Guideline として"Never use unvalidated input as part of a directive to any internal component(いかなる入力コンポーネントに対して妥当性が確認されてない入力を命令の一部として決して使用しないこと)," "Handle all errors safely(すべてのエラーを安全に扱え)"の 2 つが記されている。この Guideline は BSI[38]に登録されているものである。"Solutions and Mitigations"には 3 つの事項が"Strong input validation (強力な入力妥当性検査)," "Use of parameterized queries or stored procedures (パラメータ化したクエリあるいはストアドプロシージャの利用)," "Use of custom error pages (カスタムエラーページの利用)"が記されている。ガイドライン[9]にはこの 3 項目すべてが記載されており、攻撃パターンと Guideline との関連を示している。

5.4 考察

本節では、ソフトウェアセキュリティに関する知識間の関連に言及している研究事例を紹介した。現状では、CAPEC が最も多くの知識間の関連を扱っていることが分かった。しかし、筆者が提案した概念モデルのすべての関連を扱っていない。CAPEC は攻撃を起点として、その脆弱性や、それに対する具体的な解決策としてセキュリティパターンや guideline との関連付けを行っている。一方、Buyens らや清水は、プロセスや方法論と、principle やセキュリティパターンを関連付ける試みを行っている。これらはトップダウンで開発を行う方法を示唆している。今後はその両者を融合し、セキュアなソフトウェア開発をライフサイクル全体で扱うことを可能にする知識体系の構築が求められる。

そして、その構築がなされた際には、活用時の計算機によるナビゲーション支援、知識体系の実開発での適用可能性に関する実証実験を行う必要がある。

6. おわりに

本稿では、セキュアなソフトウェアを開発するための知識体系化のための概念モデルを提案し、各知識に関して技術開発されてきた実例の概要を紹介するとともに、知識間の関連を明らかにしている研究事例を紹介した。そして今後の方向性について議論した。

謝辞

本研究の一部は、科学研究費補助金基盤研究 (C) 22500910 の助成のもとで行われた。ここに記して謝意を表す。また、攻撃パターンの内容に関して議論していただいた情報セキュリティ大学院大学の清水啓人氏に謝意を表す。

文　献

[1] Apvrille Axelle and Pourzandi Makan, Secure Software Development by Example, IEEE Security&Privacy, Vol.3, No.4, pp.10-17, 2005.

[2] Sean Barnum and Gary McGraw, Knowledge for Software Security, IEEE Security&Privacy, Vol.3, No.2, pp.74-78, 2005.

[3] Sean Barnum and Amit Sethi, Attack Patterns as a Knowledge Resource for Building Secure Software, 2007, (Accessed 3 Jan. 2012) http://capec.mitre.org/documents/Attack_Patterns-Knowing_Your_Enemies_in_Order_to_Defeat_Them-Paper.pdf.

[4] Sean Barnum, Common Attack Pattern Enumeration and Classification (CAPEC) Schema Description, 2008.

[5] Sean Barnum and Michael Gegick. Build security in – Design principles, 2005, https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/principles.html, (Accessed 3 Jan. 2012).

[6] Sean Barnum, Cigital: Coding Rules Overview, 2005, https://buildsecurityin.us-cert.gov/bsi-rules/33-BSI.html, (Accessed 3 Jan. 2012).

[7] Koen Buyens, Riccardo Scandariato and Wouter Joosen, Process Activities Supporting Security Principles, Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC2007), Vol.2, 2007.

[8] CAPEC, http://capec.mitre.org.

[9] Michael Coates, Chris Lyon and Mark Goodwin, WebAppSec/Secure Coding Guidelines, https://wiki.mozilla.org/WebAppSec/Secure_Coding_Guidelines, (Accessed 3 Jan. 2012).

[10] Common Criteria, http://www.commoncriteriaportal.org/.

[11] CVE: Common Vulnerabilities and Exposures, The

MITRE Corporation, http://cve.mitre.org/, (Accessed 3 Jan. 2012).

[12] CWE: Common Weakness Enumeration, The MITRE Corporation, http://cwe.mitre.org/, (Accessed 3 Jan. 2012).

[13] CWE, http://www.ipa.go.jp/security/vuln/CWE.html

[14] Neil Daswani, Christoph Kern and Anita Kesavan, Foundations of Security -What Every Programmer Needs to Know-, APRESS, 2009.

[15] Mark G. Graff and Kenneth R. V. Wyk, Secure coding: principles and practices. O'Reilly, 2003.

[16] Munawar Hafiz, Paul Adamczyk and Ralph Johnson, Growing a Pattern Language (for Security), Pattern Languages of Programs Conference 2011 (PLoP2011), 2011, http://www.hillside.net/plop/2011/papers/A-39-Hafiz.pdf, (Accessed 3 Jan. 2012).

[17] 櫨山淳雄, ソフトウェアセキュリティ知識体系化のための概念モデル, 情報処理学会第 74 回全国大会, 2012 (発表予定).

[18] Gary McGraw, Software Security, IEEE Security&Privacy, Vol.2, No.2, pp.80-83, 2004.

[19] Nancy R. Mead, 吉岡信和, SQUARE ではじめるセキュリティ要求工学, 情報処理, Vol.50, No.3, pp. 193-197, 2009.

[20] JPCERT/CC, セキュアコーディングスタンダード, http://www.jpcert.or.jp/java-rules/, (Accessed 3 Jan. 2012).

[21] 金子浩之, コモンクライテリアにおけるセキュリティ要求の規定の現状と課題, 情報処理, Vol.50, No.3, pp.222-229, 2009.

[22] Oracle, Secure Coding Guidelines for the Java Programming Language, Version 3.0, http://www.oracle.com/technetwork/java/seccodeguide-139067.html (Accessed 3 Jan. 2012).

[23] OWASP, CLASP Activities, https://www.owasp.org/index.php/Category:Activity, (Accessed 3 Jan. 2012)

[24] OWASP, CLASP Security Principles, http://www.owasp.org/index.php/CLASP_Security_Principles, (Accessed 3 Jan. 2012).

[25] Jerome H. Saltzer and Michael D. Schroeder, The Protection of Information in Computer Systems, Proceedings of the IEEE, 63(9), pp. 1278–1308, 1975.

[26] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann and Peter Sommerlad, Security Patterns: Integrating Security and Systems Engineering John Wiley & Sons, 2006.

[27] 清水啓人, 櫨山淳雄, Web アプリケーション開発におけるミスユースケースを利用したセキュリティ要求獲得手法の提案, 情報処理学会第 73 回全国大会, 2011 年 3 月.

[28] Trupti Shiralkar and Brenda Grove, Guidelines for Secure Coding, 2009, http://www.atsec.com/downloads/pdf/secure-coding-guidelines.pdf, (Accessed 3 Jan. 2012).

[29] Guttorm Sindre and Andreas L. Opdahl, Eliciting Security Requirements by Misuse Case, Proceedings of the 37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-Pacific 2000), pp.120-131, IEEE CS Press, 2000.

[30] Guttorm Sindre and Andreas. L. Opdahl, Eliciting security requirements with misuse cases, Requirements Engineering Journal, Vol.10, pp.34–44, 2005.

[31] SQUARE, (Accessed 3 Jan. 2012), http://www.cert.org/sse/square/square-pubs.html.

[32] Gary Stoneburner, Clark Hayden and Alexis Feringa, NIST Special Publication 800-27 Rev A, Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A, 2004, http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf, (Accessed 3 Jan. 2012).

[33] STRIDE, http://msdn.microsoft.com/ja-jp/magazine/cc163519.aspx, (Accessed 3 Jan. 2012).

[34] John Viega and Gary McGraw. Building Secure Software: How to Avoid Security Problems the Right Way. Addison-Wesley, 2002.

[35] WASC, http://www.webappsec.org/, (Accessed 3 Jan. 2012).

[36] Ronald Wassermann and Betty H.C. Cheng, Security Patterns, (Accessed 3 Jan. 2012), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.7818&rep=rep1&type=pdf.

[37] Andreas Wiesauer and Johannes Sametinger, A SECURITY DESIGN PATTERN TAXONOMY BASED ON ATTACK PATTERNS - Findings of a Systematic Literature Review -, Proceedings of the International Joint Conference on eBusiness and Telecommunications, INSTICC Press, pp.387-394, 2009, http://www.swe.uni-linz.ac.at/publications/pdf/TR-SE-09.03.pdf, (Accessed 3 Jan. 2012).

[38] Laurie Williams, Guidelines Overview, 2005, https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/guidelines/324-BSI.html, (Accessed 3 Jan. 2012).

[39] Joseph Yoder and Jeffrey Barcalow, Architectural Patterns for Enabling Application Security, Proceedings of the 4th Conference on Patterns Language of Programming (PLoP'97), 1997.

[40] 吉岡信和, 鷲崎弘宜, 丸山勝久, セキュリティパターン技術に関する研究動向, 情報処理学会研究報告, 2007-SE-158, pp.39-46, 2007.

[41] 吉岡信和, セキュリティの知識を共有するセキュリティパターン, 情報処理, Vol.52, No.9, pp.1134-1139, 2011.

[42] 吉岡信和, 大久保隆夫, 宗藤誠治, セキュリティソフトウェア工学の研究動向, コンピュータソフトウェア, Vol.16, No.5, pp.78-95, 2011.