6R-1

脆弱性の逐次検査が可能なウェブアプリケーション開発手法の提案とその実装

大村 貴俊　　安井 浩之　　松山 実

武蔵工業大学

1. はじめに

誰でも手軽にインターネットに接続できるようになった今日，ウイルス感染，詐欺行為，プライバシーの侵害などの被害にあう危険性が高くなっている．また企業のビジネス・アプリケーションがウェブアプリケーションへ移行されつつある中，ウェブアプリケーション・サーバに対する脅威は，その種類も数も年々増加している[1]．

現在のウェブアプリケーション作成手法では，完成後に脆弱性の検査を行い，セキュリティレベルの向上を行う手順が一般的である．しかし，ウェブアプリケーションの構築と検査の工程が分離されているため，検査過程で発見された脆弱性・問題点などにより，初期工程にまでさかのぼり，設計から再構築し直さなければならない場合が多々ある．その結果，工期は遅れ，これに伴うコストも大幅に増加してしまう．

そこで本稿では，プログラムを作成する段階でウェブアプリケーションの脆弱性・問題点を逐次検査する手法を提案する．また，この手法に基づいたウェブアプリケーション開発ツールの作成を行う．これにより，脆弱性を逐次検査することが可能となり，開発効率およびセキュリティの向上が期待できる．

2. 関連研究

ウェブアプリケーション脆弱性の検査手法は主に 2 つの方法がある．ひとつがソースコードを調べる方法，もうひとつが外部からウェブサイトに自分で擬似攻撃を試み，その挙動を見て脆弱性の有無を判断する方法である[2]．一般的には後者の方法をとり，これをブラックボックス・テストという．既存製品では米 Samctum 社の AppScan[3] や米 SPI Dynamics 社の Webinspect[4]が有名である．

Tool for Web Application Development with Step-by-step Vulnerability Inspection
Takatoshi Ohmura, Hiroyuki Yasui ,Minoru Matsuyama
Musashi Institute of Technology

3. 検査手法の比較

従来のブラックボックス・テストでは，完成したウェブアプリケーションに対しての検査しか行うことができない(図 1)．そこで，ソースコードを検査するホワイトボックス･テストを使用することで，この問題を解決する．これにより，未完成のアプリケーションに対して逐次検査することが可能となり，脆弱性の早期発見と早期修正に役立てることができる(図 2)．

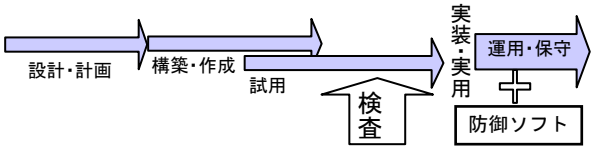

図 1 従来の検査手法のタイミング

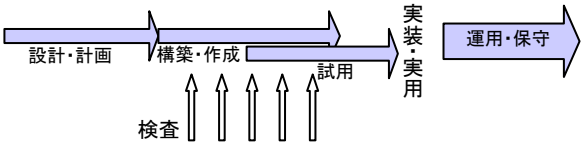

図 2 提案する検査手法のタイミング

しかし，ホワイトボックス・テストは従来手作業で行うため，大規模なウェブアプリケーションになると，ソースコードが大量かつ複雑になり，すべてのソースコードを調査すると膨大な時間と労力がかかる．そこで，この作業を軽減するために，逐次検査が可能なツールが必要となる．

4. 脆弱性と対処方法

ここでは対応するウェブアプリケーションの脆弱性を，クロスサイトスクリプティング，バッファオーバーフロー，パラメータの改ざん，SQL インジェクション，OS コマンドインジェクションの 5 つに限定する．なぜなら，これらの脆弱性の，最大の原因として，ユーザが入力したフォームのリクエストを検査していないことが挙げられ，root 権限の奪取やウェブアプリケーションが利用するデータベースの内容改ざんなど，致命的な被害を受けることが多いためで

ある．対処方法は言語によって異なるが，サニタイジング(無毒化)関数を使用するのが一般的である．

5. 本手法を用いた開発ツール

5.1 概要

本開発ツールでは，テキストエディタをベースとし，ウェブアプリケーションの脆弱性を逐次検査する機能を持たせている．検査機能は脆弱性を次項の方法で検査し，作成者に警告を与える．自動的に修正する機能を持たせなかったのは，脆弱性の大半が作成者の脆弱性に対する無知が挙げられるため，これら脆弱性を認知しなければ脆弱性を含むソースを作り続けてしまうと考えたためである．

また，検査対象の言語を，データベースと親和性が高く，近年多く使われている PHP に限定した．

5.2 検査手法

本開発ツールでは，ソースコードを字句解析・意味解析しパターン化を行う．その後脆弱性のパターンファイルとの間でパターンマッチングを行う．一致した箇所は脆弱性を含んでいるとみなし，ユーザに警告を出す．パターンファイルは拡張性を持たせるため，外部から読み込む形をとる．パターン化とマッチングの具体例を図 3 に示す．図 3 のソースコードは典型的なクロスサイトスクリプティングに関する脆弱性であり，ユーザから送られてくる情報が収納された変数に対してサニタイジングを行っていないために起こる．

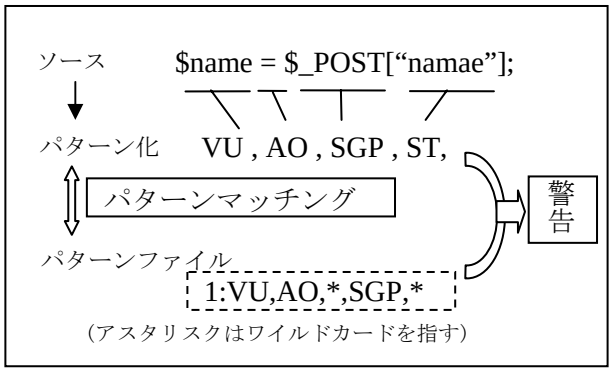

図 3 パターン化と解析の手順

5.3 パターンファイルルール作成

脆弱性パターンファイルは，脆弱性の種類とパターン要素の組み合わせで成り立っている．

重要視している要素としない要素により，同じ種類(関数なら関数)でも複数種類のパターン要素を定義する．要素は約 15 種類あり，それ以外にも任意の関数などを別途定義できる．表 1 に脆弱性のパターンファイルの一部を示す．

表 1．脆弱性パターンファイルの一部

```
#1:クロスサイトスクリプティング
1:F,*,SGG,*
1:F,*,SGP,*
1:VU,AO,*,SGG,*
1:VU,AO,*,SGP,*
                ⋮
```

5.4 脆弱性を検出するタイミング

本ツールが脆弱性を検出するタイミングは，ユーザが検査実行をするとき以外にも，ファイルを読み込んだときと，数行テキストが記入されたときも検査を行う．これにより逐次性を高めている．

6. まとめと今後の課題

脆弱性パターンファイルを導入し，本稿で作成したツールを使用することで，未完成のウェブアプリケーションに対しても，脆弱性を検査することが可能となった．ホワイトボックス・テストにおいて，ツールを導入することができたことで，脆弱性の検出頻度があがった．

しかし，本ツールでは複数文にまたがって存在する脆弱性には対処できず，また HTML 文中に組み込まれた PHP に対しても検出漏れが多い結果となった．今後，実用化に向け，上記問題を解決することと，GUI 面を洗練し，脆弱性の箇所をより明示できるようにツールを改良する必要がある．同時に，複数文にまたがった脆弱性の検出を可能とするパターンファイルの改良を行う必要がある．

また，ウェブアプリケーションの分野は，現在成長過程にあり，言語の仕様も日々変化しているため，ウィルスソフトの定義ファイルのように，脆弱性パターンファイルも更新していく必要がある．

参考文献

[1] 高木 浩光,”Web アプリケーションにおける脆弱性”，情報処理，vol.46，No.6，pp.636-642，June. 2005

[2] 徳丸 浩，Web アプリケーションのセキュリティ，日経コミュニケーション pp.66-69, 2005.11.25

[3] AppScan: http://www.watchfire.com/

[4] WebInspect: http://www.spidynamics.com/