社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－DPS－125 (8)
2005－EIP－ 29 (8)
2005／11／14

Webシステムにおけるセキュリティ検査手法の検討

坂田 匡通* 中山 弘二郎* 西木 健哉* 石崎 健史* 富坂 稔**

*（株）日立製作所 システム開発研究所
**（株）日立製作所 ソフトウェア事業部

概要 現在，IT製品の設定ミス等による脆弱性を根絶するために，脆弱性を自動検査するツールが開発されている．しかし，これらのツールはOSのセキュリティパッチのバージョンなど製品単体で検査は可能だが，複数のコンポーネントが影響しあうWebシステムの中で，どの部分が問題の根本なのかといった詳細な検査は実現されていない．本報告では，Webシステムのコンポーネント間の関係やリクエストのコンテキスト情報を用いて詳細なセキュリティ検査を実現する手法について提案する．

Security Analysis Technique in Web System

Masayuki SAKATA* Kojiro NAKAYAMA* Ken'ya NISHIKI* Takeshi ISHIZAKI*
Minoru TOMISAKA**

*Hitachi Ltd. Systems Development Laboratory
*Hitachi Ltd. Software Division

Abstract There are some tools for analyzing vulnerabilities caused by miss configurations of IT products. These tools are useful to check configurations of each product like OS security patch version. However, it couldn't analyze which part has root problems in the web system that consists of many components. In this paper, we propose security analyzing technique using the relationship of components and request context.

1. はじめに

近年インターネットにおいて，情報漏洩やWebページの書き換えなど様々な被害が報告されている[1]．これらは，Webアプリケーションの開発やシステム構築時に作りこまれるセキュリティホールを突いたものや，ミドルウェアやOSに含まれる脆弱性を攻撃されることにより引き起こされている．

ただし，これらの脆弱性は既に原因と解決策が広く知られているものが多い[2]．それにも関わらず，脆弱性がなくならない原因の一つとして考えられるのが，セキュリティの知識不足である．短期間でのシステム開発が求められる現代の開発では，全ての開発・構築・運用者に高いセキュリティ知識を要求することは困難である．特にWebシステムは表1に示すように非常に多くのコンポーネントとセキュリティ機能を有しているため，システム構築者がミスなくセキュアなWebシステムを構築するためには，セキュリティに関する広い知識に加え，各製品の持つ膨大な数の機能を正しく設定するノウハウが必要となっている．

このような問題に対して，米国では，NIST（National Institute of Standards and Technology）が中心となり，脆弱性を根絶するための施策“Security Configuration Checklists Program for IT Products”が進められている．この施策では，製品固有のセキュリティ設定を検査するためのXML形式のフォー

マット（XCCDF：Specification for the Extensible Configuration Checklist Description Format）が定められている[3]．IT製品ベンダが，このフォーマットに準拠したチェックリストを用意し，VA（Vulnerability Assessment）ツールと呼ばれる検査ツールにより自動検査することにより，IT製品の脆弱性を根絶することが期待されている．

しかし，現在提供されているVAツールは，OS用セキュリティパッチのバージョンなど製品単体で検査可能な項目の検査や，システム外部からのネットワークを介した脆弱性スキャンは可能だが，表1のような複数のコンポーネントから成るWebシステムの中で，どの部分が問題なのかといった詳細な検査は実現されていない．

本報告では，このような問題を解決するために，分散コンポーネントからなるWebシステムに対して適用可能なセキュリティ設定検査手法について検討を行った．以降，2章では従来手法と課題，3章で本提案手法の詳細，4章では実装について説明する．

表1 Webシステムのコンポーネント構成例

#	コンポーネント	セキュリティ機能
1	ファイアウォール	パケットフィルタリング VPN
2	IDS	パケット監視 侵入検知
3	SSL アクセラレータ	通信路暗号 認証
4	プロキシサーバ	リクエストサイズ制限 リクエストタイプ制限
5	Web サーバ	強制ブラウジング対策 認証/認可/アクセス制御 サーバ情報公開制御
6	アプリケーションサーバ	認証/認可/アクセス制御 SSL 要請, Secure Cookie サニタイジング ウィルス検査, 署名・暗号
7	データベース	認証/認可/アクセス制御 監査ログ
8	LDAP ディレクトリ	認証/認可/アクセス制御 監査ログ
9	運用管理サーバ	認証/認可/アクセス制御 監査ログ

2. 従来手法の課題

2.1. 従来手法

製品やシステムの脆弱性を検査するVA手法／ツールは，調査の方法によって大きくネットワーク型とホスト型の2種類に分けることができる（図1）[4][5].

(1) ネットワーク型VA

ネットワーク型VAはネットワーク越しに検査対象サーバにパケットを送り，その反応を調べることで脆弱性を調査する手法である．ホスト型と比べ，サーバにエージェントをインストールする必要がないという利点がある．

(2) ホスト型VA

一方，ホスト型VAは検査対象のサーバにエージェントをインストールし，エージェントがサーバのコマンドを利用したり，設定ファイルを直接検査することで，セキュリティポリシに合致しているかを内部から検査する．

一般的なセキュリティ知識と，製品固有のセキュリティ機能をマッピングすることで，詳細なレベルでの自動検査を可能とし，設定ミスを防止することができる．また，XCCDFを用いることで，様々なコンポーネントからなるWebシステムに対しても，統一された検査機構を提供することができる．

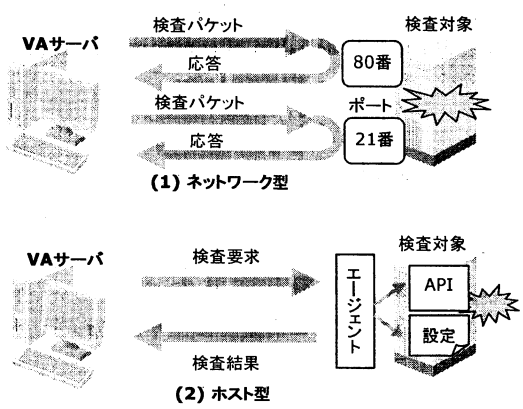

図1 ネットワーク型VAとホスト型VA

2.2. 課題

ネットワーク型 VA は，サーバにエージェントをインストールする必要がないという利点がある一方で，検査可能な項目がネットワーク越しで検査可能な範囲に限定される．そのため，例えば監査用ログの設定や，パスワードの暗号化設定がセキュリティポリシに合致しているか検査できない．また，脆弱性の存在は分かっても，システム内の何処が問題で，どのように修正したら良いのかまでは分からない．

一方ホスト型は，詳細なレベルでの検査が可能であるが，機器や機能間での設定矛盾を検出することができないといった課題があった．これに対し文献[5]では，ファイアウォールおよび IDS を対象にして，各機器のセキュリティ機能設定を関連付けて分析している．

しかし，例えば Web サーバの後段に複数のアプリケーションサーバが配備されていて，リクエスト URL によって Web サーバからアプリケーションサーバへの転送先が異なるような Web システムでは，システム構成や転送設定に応じて，影響を及ぼす範囲が異なる．このような，サーバ間の影響範囲に応じた分析は行われていない．

また，Web の場合は URL 毎に受信されるデータの意味が異なる．例えば，ある URL へのリクエストデータはデータベースアクセスに利用され，また別のリクエストデータは確認用にユーザへのレスポンス画面に表示される．そのため，SQL インジェクションやクロスサイトスクリプティングを防ぐためには，これらのデータに対するサニタイジング（無害化）処理が必要になる．サニタイジング処理の設定は，例えば対象 URL とパラメータ名，サニタイジング方法を指定することで実現されている．

しかし，上記のようなセキュリティ機能が適切に設定されているか検査するためには，リクエスト内のパラメータが使用される場所や意味（コンテキスト）を理解しなければならない．現状の VA ツールではこのようなコンテキストを加味した検査は実現されていない．

以上の検討により，Web システムのセキュリティ検査項目には以下の 4 種類があることが分かった．次章以降では現在実現されていない (3),(4)の項目の検査方法について説明する．

(1) ネットワーク越しの検査で検査可能な項目
(2) コンポーネント内部からの単体検査が必要な項目
(3) 構成情報や他のコンポーネントの設定に影響を受ける項目
(4) コンテキスト情報が必要な項目

3. 提案方式

本研究では，幅広い範囲のセキュリティ設定検査が可能なホスト型 VA をベースにして，さらに上記課題を解決するために，リクエストのコンテキスト情報およびコンポーネント間の設定影響情報を用いた検査手法を提案する．

3.1. 複数コンポーネントに跨る検査

Web システムを構成するコンポーネントのセキュリティ設定が，他のコンポーネントに与える影響を考慮した検査を行うために，検査間の関係情報に従って検査を実行する仕組みを導入した．以下に詳細を説明する．

例えば「アプリケーションサーバの認証時には Transport Guarantee 機能を使用する」といったポリシーを検査するためには，「前段 Web サーバ等で SSL が設定されている」といった前提検査が必要な場合がある．このような場合は，検査処理の動作を規定している VA ツール用検査定義ファイルに上記二つの検査定義を記述し，検査間の関係を示す前提検査タグ <precheck>内に，対応する検査定義へのリファレンスを記述する（図 2）．

```
<check id="SSLRequired">
  <prechecks>
    <precheck href="SSLSetting"/>
    ...
  </prechecks>
  <target>xxx</target>
  <impl name="xxx"/>
  <output>
    <threat-info>xxx</threat-info>
    <result level="warn" counter-measure="xxx">
    ...
  </output>
</check>
<check id="SSLSetting">
  ...
</check>
```

図 2 検査定義ファイル

VA ツールの管理サーバ（以下 VA サーバ），は設定ファイルを読み取ると，前提検査条件の指定内容と，システム構成情報から検査フローを決定し，検査フローに沿って各コンポーネント上に配備した VA エージェントに対して検査要求を行う．VA エージェント上には予め検査定義に対応した検査ロジックを組み込んでおく．

検査エージェントは検査を行うと，検査結果を VA サーバに返信する．VA サーバはフローに従い，次のコンポーネント上の VA エージェントに対して検査要求を行う．この際，検査要求には，前提検査の結果を含めて送信する．

また，構成情報は別途入力するか，構成情報を管理している構成管理サーバより取得するものとする（図 3）．

以上により，構成情報や他のコンポーネントの設定に影響を受ける項目についての検査を実現する．

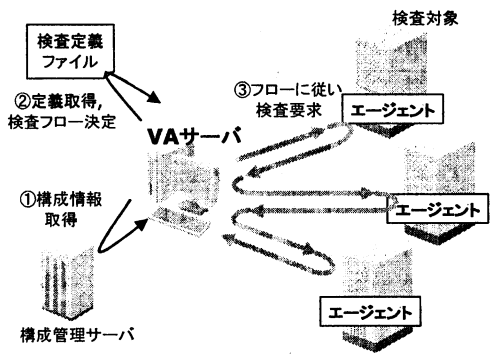

図 3 複数コンポーネントに跨る検査

3.2. コンテキスト情報を利用した検査

前章で示したように，SQL インジェクション等の Web レベルでの攻撃に対するセキュリティ機能（例えばサニタイジング処理）が適切に設定されているか検査するためには，リクエスト内のパラメータが使用される場所（コンテキスト）を入力として VA ツールに与えなければならない．しかし，コンテキストを全て手動で設定するのは非常に手間がかかり，またシステム内部の構成に精通していなければならない．そこで，コンテキスト収集機能の検討を行った．

(1) コンテキスト情報収集機能

コンテキスト収集機能は，擬似クライアントと監視フィルタ，解析部とから構成される．擬似クライアントは，検査対象の Web システムに対して，パラメータの値としてトレース値を設定したリクエストメッセージを送信する．監視部はシステムの各箇所においてトレース値が使用されているかどうか監視を行なう．解析部は監視結果を解析し，リクエストメッセージ内のパラメータが使用される箇所を特定する．

ただし，これらの検査は実運用前のシステム構築時に行うことを想定しているため，監視フィルタは既存システムに影響を与えることなく導入可能である必要がある．

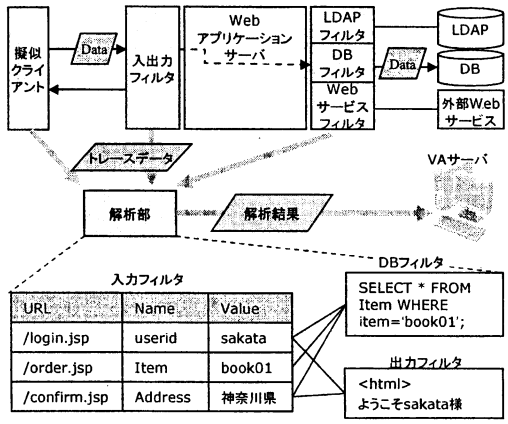

図 4 コンテキスト情報の収集

(2) コンテキストからの検査ルール生成

VA サーバは，擬似クライアントおよび監視部による追跡結果と，予め設定した対策ポリシーとを元に，検査ルールを生成しエージェントに送信する．対策ポリシーには，様々なパラメータのコンテキストに対する推奨対策が規定してある．そのため，追跡結果と付き合わせることで，検査に必要な検査ルールを容易に生成することができる．

対策ポリシー例を表 2 に示す．ここでは，SQL やレスポンスといった，パラメータが使用される場所をコンテキストとして挙げている．ただし，同じ SQL を用いたクエリーでも，通常のクエリーとバインドメカニズムを用いたクエ

リーでは対策が異なるため，別々のコンテキストとした[6]．同様にレスポンスもパラメータの記述される位置に応じて対策が異なるため，記述されている位置に応じた対策が必要である．

表 2 コンテキストに応じた対策ポリシー例

#	コンテキスト	対策	推奨設定値
1	SQL 通常	サニタイズ A	禁止文字:;=
2	SQL バインド	不要	-
3	レスポンス 本文	サニタイズ B	禁止文字 &<>"'
4	レスポンス ヘッダ	サニタイズ C	禁止文字 %0D%0A
5	ファイル出力	ウィルス検査	multi-part

4. 実装

4.1. 試作システム

試作システムは，アプリケーションとしてオンラインショッピングサイトを想定し，フロントエンドに Web サーバ，ミドルレイヤーに J2EE サーバ，バックエンドにデータベースからなる Web 3 階層構成とした．また，本提案での有効性を検証するために，Web サーバおよび J2EE サーバを検査対象サーバとし，各サーバ上には VA エージェントを配備した（図 5）．

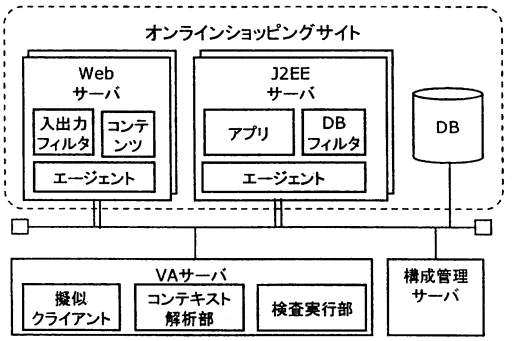

図 5 試作システム構成

また，パラメータを追跡するために，Web サーバ上に入出力フィルタ，J2EE サーバ上に DB フィルタを配備した．監視フィルタは既存システムに変更を加えずに導入する必要がある．

Java ではデータベースへアクセスする標準インタフェースとして JDBC API[7]が提供されており，各データベースベンダはこのインタフェースを実装した JDBC ドライバを提供している．また，J2EE で提供されているデータソース機能を使うことにより，設定変更だけで JDBC ドライバの切換えを行うことができる．

そこで，JDBC API を実装し，かつ各ベンダの JDBC ドライバをラッピングする JDBC フィルタとして DB フィルタの実装を行った（図 6）．これにより，J2EE サーバの設定を変更するだけで，監視を実現することができる．

また同様に，既存システムに影響を与えないため，入出力フィルタは Web プロキシとして実装を行い，HTTP リクエストのパラメータ (Cookie を含む)を検出した．

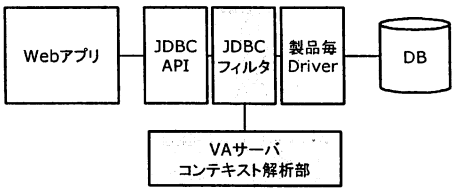

図 6 DB フィルタの実装

4.2. 複数コンポーネントに跨る検査

通常 Web サーバでは，特定の URL パターンにマッチしたリクエストを J2EE サーバに転送する．そこで，本試作ではまず転送設定を各 Web サーバより取得することで，Web サーバと J2EE サーバの接続関係を検査し，さらに検査定義ファイルの前提検査条件とにより検査フローを決定し，検査を実行した（表 3，図 7）．これにより，コンポーネントに跨る検査を実現した．

表 3 サーバ間の接続関係

#	転送元サーバ	URL	転送先サーバ
1	Web サーバ A	/login	J2EE サーバ A
2	Web サーバ B	/register	J2EE サーバ B

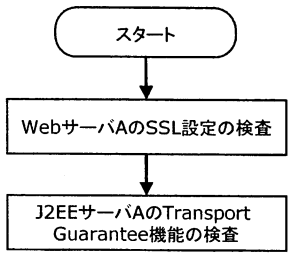

図 7 検査フローの決定

4.3. コンテキスト情報解析結果

コンテキスト情報の解析結果を表 4 に示す．表より，例えば入力された認証データが，そのまま通常の SQL のパラメータとして使用されており，サニタイジングの処理が必要であることが分かる．この結果を基に，実際のサニタイジング処理機能を検査することが可能となる．

表 4 コンテキストの解析結果

リクエスト URI	入力パラメータ	コンテキスト			対策
		レスポンス本文	SQL 通常	SQL バインド	
/login	uid	○	○		サニタイズA, B
/login	password		○		-
/	JSESSIONID				-
/menu	category	○		○	サニタイズB
/order	model	○		○	サニタイズB
/order	product_id	○		○	サニタイズB
/order	number	○		○	サニタイズB

4.4. 検査レポート例

検査結果は HTML 形式のレポートとして管理者に報告される（図 8）．検査レポートには検査対象サーバ情報，脅威の説明，結果サマリ，結果詳細などが記載されている．本提案方式により，精度の高い検査が可能となったため，結果詳細には問題のある現状の設定値情報，具体的な推奨対策方法を明示している．これにより，システム構築者は容易にセキュリティホールを防ぐことができる．

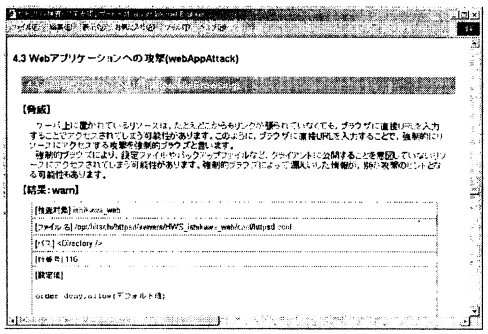

図 8 検査レポート例

5. おわりに

本稿では，複数のコンポーネントからなる Web システムのセキュリティホールを検査する手法として，リクエストのコンテキスト情報およびコンポーネント間の設定影響情報を用いた検査手法を提案した．これにより，精度の高い Web システムのセキュリティ検査が可能となり，設定ミスの防止や構築コストの低減につながると考えられる．

今後の課題としては，コンテキストの拡充が挙げられる．本報告ではパラメータが使用される場所をコンテキスト情報として利用しているが，個人情報のように使用場所に関らず，意味に応じたセキュリティが必要なデータに対しては，意味を理解することで適切なセキュリティ設定や検査が可能であると考えられる．

6. 参考文献

[1] The Open Web Application Security Project,「Web アプリケーションの脆弱性トップ 10」（2004.1）
http://www.owasp.org/

[2] 高木浩光,「なくならないシステムの脆弱性」,
http://java-house.jp/~takagi/paper/ecom-security-seminar-4-takagi-dist.pdf

[3] NIST,「Specification for the Extensible Configuration Checklist Description Format(XCCDF)」,
http://csrc.nist.gov/checklists/docs/xccdf-spec-1.0.pdf

[4] 日経 BP,「サーバーやネット機器のぜい弱性を調べる VA」, 日経 NETWORK, 2004/07 号

[5] 岡城純考 他,「セキュリティ運用管理における機器設定統合分析システム」情報処理学会研究報告, 2005-DPS-122, pp.303-308(2005)

[6] 独立行政法人 情報処理推進機構,「セキュアプログラミング講座」
http://www.ipa.go.jp/security/

[7] JDBC Technology.
http://java.sun.com/products/jdbc/.

なお，本研究の一部は，平成 17 年度総務省「ユビキタスネットワーク認証・エージェント技術の研究開発」の研究助成によるものである．