Webアプリケーション診断ツール検証のための疑似脆弱性サーバー構築

氏原 悠貴[1,a)]　樋口 光輔[1,b)]　小林 良太郎[1,c)]

概要：近年，Web サーバーの脆弱性を悪用したサイバー攻撃が増加しており，その対策として脆弱性診断などが挙げられる．しかし, 現状では, 専門知識を要する手動診断が主流であり，対応可能な人材の不足が課題となっている．こうした課題を背景に，脆弱性診断を自動化するツールの開発が進められている．一方で，開発された自動化ツールの精度を客観的に評価するための多様な脆弱性を網羅した検証環境が不足している課題がある．既存の検証環境は特定の脆弱性の再現に限定されていることが多く，ツールの評価を体系的に行う上での障壁となっている．そこで本研究では，標準的な診断項目で示される多岐にわたる脆弱性を再現可能な擬似脆弱性サーバーの設計・構築を行い, 自動化ツールの体系的な精度評価に資する環境を提供することを目的とする．

キーワード：Web アプリケーション, 脆弱性サーバー, 検証

Construction of a Pseudo Vulnerability Server for Testing Web Application Vulnerability Scanners

Yuki Ujihara[1,a)]　Kosuke Higuchi[1,b)]　Ryotaro Kobayashi[1,c)]

Abstract: In recent years, cyberattacks exploiting web-server vulnerabilities have grown in frequency, underscoring the critical importance of vulnerability assessments. Currently, these assessments are performed manually and demand specialized expertise, resulting in a shortage of qualified personnel. Consequently, research into tools for automating vulnerability assessment has accelerated. However, progress in evaluating the accuracy of such automated tools is hindered by the lack of testing environments capable of emulating a comprehensive range of vulnerabilities. Most existing environments are confined to reproducing specific vulnerabilities, thereby impeding systematic evaluation. This study therefore proposes the design and implementation of a pseudo-vulnerable server that can reproduce a diverse spectrum of vulnerabilities based on standard assessment criteria, with the aim of providing a robust environment for systematic accuracy evaluation of automated vulnerability-assessment tools.

Keywords: Web server,Vulnerability assessment

1. はじめに

近年, 開発支援ツールやクラウドサービスの充実により, Web アプリケーションの導入障壁が低下している．その結果, Web アプリケーションは業務効率化や情報提供, サービス提供を目的として広く開発・活用されている．一方で, Web アプリケーションの脆弱性を利用した攻撃が増加しており, それに伴い被害も拡大している．Web サイトにおける脆弱性とは, プログラムの不具合や設計ミスなどに起因するセキュリティ上の欠陥であり, これらを早期発見するために Web 脆弱性診断の重要性が高まっている．Web 脆弱性診断の目的は, システムやアプリケーションに潜在す

[1] 工学院大学
Kogakuin University, Shinjuku, Tokyo 163-8677, Japan
[a)] j122050@ns.kogakuin.ac.jp
[b)] ed25001@ns.kogakuin.ac.jp
[c)] ryo.kobayashi@cc.kogakuin.ac.jp

るリスクを特定し, 情報漏洩や不正アクセスの防止を図ることである. 診断手法には手動診断があり, これはセキュリティ専門家が手作業で脆弱性の有無を調査する方法である. しかし, 手動診断には専門的な知識や経験が必要であり, コスト増加や診断期間の長期化といった課題がある. このため, 診断ツールを用いて脆弱性を自動診断する手法（以降ツール診断とする）が活用されている. 診断ツールの 1 つとして Burp Suite が挙げられる. Burp Suite は Web アプリケーションのセキュリティテストを実施するためのツールであり, ブラウザと Web サーバー間の通信を中断し, リクエストとレスポンスをリアルタイムで確認・編集できるなど, 多様な機能が備わっている. ツール診断はこうした機能を活用することで短時間で広範な検査が可能であり, 診断費用の削減にも寄与する. 一方で, 偽陰性や偽陽性による誤検知や見落としが発生する可能性があり, ツールの診断精度を検証する必要がある. しかし, ツール検証のために本番環境を利用することは, システム停止やデータ破損のリスクを伴う. このため, ツールの精度検証を目的として, 意図的に脆弱性を搭載したサーバーの 1 つにやられサーバーややられサイトが存在する. やられサーバーややられサイトはツールの精度検証以外にも脆弱性診断を学ぶためのものとして使用される. しかし, 既存のやられサーバーは学習用途が中心であり, その数は十分ではない. さらに, 既存のやられサーバーややられサイトでは特定の脆弱性のみであったり, データベースや apache を構築する必要がある. 以上の課題を踏まえ, 本研究では, 脆弱性の有無を切り替えることが可能な疑似脆弱性サーバーの構築を行う. 搭載する脆弱性は, Web アプリケーション診断ガイドラインに記載されている脆弱性を参照することで, 網羅的かつ実践的なツール検証環境を提供することを目的とする. 本論文の構成は以下の通りである. 第 2 章で関連研究について述べ, 第 3 章で提案システムについて説明する. 第 4 章では提案システムの検証を行い, 第 5 章でまとめおよび今後の課題について述べる.

2. 関連研究

Web 脆弱性診断については, 様々な診断方法が存在しており、診断ツールのための脆弱性サーバーが存在している.

2.1 自動診断ツール

Muhammad Noman らは, SQLInjection (SQLi) や Cross-Site Scripting (XSS) などの Web 脆弱性を検出するためのブラックテスト手法を提案し, これらの脆弱性のほとんどに対する検出ツールとして Web vulnerabilities Finder (WVF) を開発した [1]. WVF は, Web サイトの脆弱性発見を目的とした Web サイトを自動的に解析する手法である. また, Paulo Jorge Costa Nunes らは, オブジェクト指向プログラミングを使用して開発された PHP プラグインの脆弱性を特定を特定する静的コード解析ツールである PHPSAFE を開発した [2]. PHPSAFE は WordPress プラグインのような PHP ベースの Contents Management System(CMS) プラグインに対し、オブジェクト指向プログラミング（OOP）構造を考慮しながらソースコードを解析し、脆弱性を検出できる. また, Robert Vibhandik らは, W3AF と Nikto のツールを組み合わせて, Web アプリケーションの脆弱性評価テストを実証している [3]. J. Kartheek Reddy らは, Golang と Vue.js を使用して構築された VulnGuard という Web アプリケーションの脆弱性スキャナーを使用して, 脆弱性を検出している [4].

2.2 脆弱性サーバー

2.2.1 DVWA

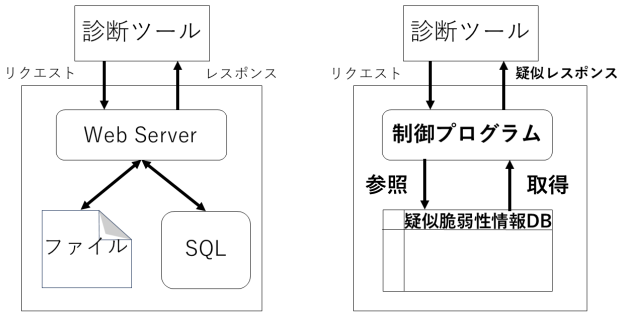
Damn Vulnerable Web Application (DVWA) は Web アプリケーションのセキュリティ学習を目的に設計されたオープンソースの脆弱性学習ツールであり, 2009 年に Ryan Dewhurst によって作成された [5]. DVWA はローカル環境で動作をし, PHP と MYSQL を使用している. また, DVWA はオープンソースであるため, コミュニティの支援を受けながら継続的に改善されている. DVWA は脆弱性の種類や攻撃の流れを模擬的に体験することができ, SQLi や XSS, クロスサイトリクエストフォージェリ (CSRF) など様々な脆弱性が搭載されている. また, DVWA には使用者に応じたレベルを設定する機能が備わっており, 初心者から上級者まで幅広いユーザーに対応している.

2.2.2 BadToDo

BadToDo は, 脆弱性診断実習用のアプリであり, EG セキュアソリューションズの CTO の徳丸浩氏が開発した [6]. BadToDo は, Web ブラウザ上で動く ToDo リストアプリとして動作するが, 情報処理推進機構 (IPA) の「IPA ウェブ健康診断仕様」や国際 Web セキュリティ標準機構の「OWASOP Top 10」で紹介されている脆弱性を網羅的に含む, 脆弱性のアプリとなっている. BadToDo には, SQLi や XSS, バッファオーバーフローなどの脆弱性が搭載されている.

2.3 Web アプリケーション診断ガイドライン

Web 脆弱性診断のガイドラインとして, 「Web アプリケーション診断ガイドライン」が公開されている. これは、JNSA のセキュリティオペレーションガイドライン WG と OWASP Japan が主催する共同ワーキンググループ「脆弱性診断士スキルマッププロジェクト」によって策定された [7]. WG は自動診断ツールを使用した脆弱性診断だけでなく, 手動診断を併用した脆弱性診断が望ましいと考えている. しかし, 手動診断には脆弱性診断士としての経験やスキルによる差異が生じる. そのため, Web アプリケーション診断ガイドラインでは, 一定レベルの手動診断によ



(a) 従来の脆弱性サーバー　(b) 疑似脆弱性サーバー

図 1 脆弱性の診断に使用されるサーバー

る脆弱性診断をできるように, 最低限必要な診断項目や手順が定義されており, 手動診断の初学者であっても分かりやすく, 最低限の診断レベルを担保できると考えられている. Web アプリケーション診断ガイドラインでは, 「脆弱性名称」, 「ペイロード・検出パターン」, 「診断方法」, 「脆弱性があると疑われる挙動」, 「備考」等の項目が定義されている.

3. 提案システム

本研究は, 脆弱性診断の自動化ツールの体系的な精度評価に資する環境を提供することを目的としており, Web アプリケーション診断ガイドラインに基づいた疑似脆弱性サーバーの構築を行うものである. 図 1 に示すように, 提案システムは制御プログラムと脆弱性情報 DB の 2 つのモジュールで構成されている.

3.1 制御プログラム

制御プログラムでは, Web インターフェースを使用して有効にする脆弱性名称を脆弱性情報 DB に送信する. また, 診断ツールからリクエストが送られた際には, リクエストの内容からペイロードだけを抽出する. その後, 脆弱性情報 DB を参照し, 抽出したペイロードと脆弱性情報 DB 内のペイロードと照合する. 照合を行い一致するペイロードが存在した場合, そのペイロードと対になっている疑似レスポンスを受け取る. 疑似レスポンスは DB 内の脆弱性が有効・無効によって変化する. 有効状態の場合には, DB から疑似脆弱性レスポンスを受け取る. 脆弱性が無効の状態の場合は, 正常レスポンスを受け取る. 受け取った疑似レスポンスをツールに返却する. 制御プログラムでは, レスポンスが DB に依存しているため, ある脆弱性が無効の状態でその脆弱性が持つペイロードを含むリクエストが送信された場合でも常に正常レスポンスを返すようになっている.

3.2 脆弱性情報 DB

3.2.1 DB が持つ情報

脆弱性情報 DB には, 脆弱性の種類, ペイロード, レスポンスなどが定義されており, これらをペアで管理している. そのため, データベースサーバーや Web サーバーから実際のレスポンスを返却するのではなく, アプリケーション層でレスポンスを返却する仕組みとしている. 脆弱性情報 DB の中身は図 2 のようになっており, 各項目とその役割について以下に示す.

- MODE: 脆弱性の有効・無効を示すフラグ。1 が有効、0 が無効
- 脆弱性名称: 対象となる脆弱性の種類
- ペイロード: 脆弱性を検証するための入力値
- 脆弱性レスポンス: 脆弱性が有効な場合のサーバーからの応答
- 正常レスポンス: 脆弱性が無効な場合のサーバーからの応答
- ステータスメッセージ: サーバーの状態を示すメッセージ
- ステータスコード: HTTP ステータスコード

3.2.2 脆弱性情報 DB の動作

管理時は, 制御プログラムから送信される脆弱性名称を受けとり, 送られた脆弱性名称と同じ脆弱性名称の行を DB 内から探し, 有効状態に設定する. また, 脆弱性情報 DB に存在するが送られなかった脆弱性名称は, 無効状態に設定し保存している. 診断時では, 診断ツールからリクエストが送られると, 脆弱性情報 DB を参照し、実際のデータベースサーバーや Web サーバーにアクセスする代わりに, ペアになっているレスポンスを返却する. これにより, 診断ツールは特定脆弱性の有無を個別に検証でき, 誤検知や漏れ検知の特定が可能となる.

3.3 システム処理

本システムは, 診断ツールから HTTP リクエストが送られた際に, 受信したリクエストの内容を確認し, 脆弱性情報 DB の中身を参照してレスポンスを決定する. 具体的には, 診断ツールからペイロードを含むリクエストが送信された際に, 脆弱性名称に基づいて脆弱性情報 DB を検索し, MODE 列の値を確認する. MODE 列の値が 1 の場合は脆弱性が有効な状態であるため, 脆弱性レスポンスを返却する. 一方で, MODE 列の値が 0 の場合は脆弱性が無効な状態であるため, 正常レスポンスを返却する. このようにして, 診断ツールからのリクエストに対して脆弱性の有効・無効に応じた疑似レスポンスを返す仕組みとなっている. また, ユーザーが Web インターフェース上で脆弱性の有効・無効を切り替えると, 選択された脆弱性名称と同じ脆弱性名称の MODE 列が脆弱性情報 DB 内で 0 から 1 に変更され, 選択されなった脆弱性の名称の MODE 列が 1 から 0 に変更される. これにより, ユーザーは有効にする脆弱性の組み合わせを自由に設定できる.

```
MODE,脆弱性名称,ペイロード,
脆弱性レスポンス,
正常レスポンス,ステータスメッセージ,ステータスコード
0,SQLInjection,',
SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual
that corresponds to your MariaDB server version for the right syntax to use near ''''' ',
問題ありませんでした,Internal error,500

0,SQLInjection,OR'1'='1--,
SQLInjection error,
問題ありませんでした,Internal error,500

0,XSS,"">'><s>XSS",
<s>XSS' SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check
the manual that corresponds to your MariaDB server version for the right syntax to use near 'XSS'' </s>,
脆弱性がなかったためエスケープされました。,Internal error,500

0,XSS,"<script>alert(1)</script>",
SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual
that corresponds to your MariaDB server version for the right syntax to use near '>'',
脆弱性がなかったためエスケープされました。,Internal error,500
```

図 2　脆弱性情報 DB の中身

4.　提案システムの検証

本研究では, 提案システムの実装を行った. そして実装した疑似脆弱性サーバーに対して Burp Suite を使用して HTTP リクエストを送信し, レスポンスが正常に返せているか検証を行った. 検証では, Web インターフェースで切り替えた脆弱性名称が脆弱性情報 DB に正しく反映されているかを確認した. また, 脆弱性を有効化した状態で特定のペイロードを送信した場合に, 脆弱性レスポンスが返却されるかを検証した. 加えて, 脆弱性を無効化した場合に, 正常なレスポンスが返却されるかを検証した. これにより, 構築した疑似脆弱性サーバーが正常に動いているかを確認した.

4.1　実装

提案システムの実装に用いたフレームワーク及び仮想環境は以下の通りである.

- ホスト OS: Windows 11 Pro
  - プロセッサ: Intel Core i7-1165G7
- 使用ソフトウェア: VirtualBox 7.0.14
- ゲスト OS: kali-linux-2024.1
  - CPU コア数: 2
  - メモリ: 2048GB
- プログラミング言語: Python-3.11.8
- フレームワーク: Flask-2.2.5, Werkzeug-2.3.8

4.2　実行時の挙動

本研究で構築した疑似脆弱性サーバーが正常に動作することを確認するために, 例として SQLi の脆弱性を有効化し, ペイロードとしてシングルクォーテーション「'」を送信した.

4.2.1　有効にする脆弱性の選択

サーバーを起動すると図 3 のような画面が表示される. 図 3 に表示されているように, すべての脆弱性を有効化することや複数存在する脆弱性から任意の組み合わせを有効化することが可能になっており, 各脆弱性についての簡単な説明も表示される. ユーザーは有効にしたい脆弱性名称を選択し, 送信ボタンをクリックする.

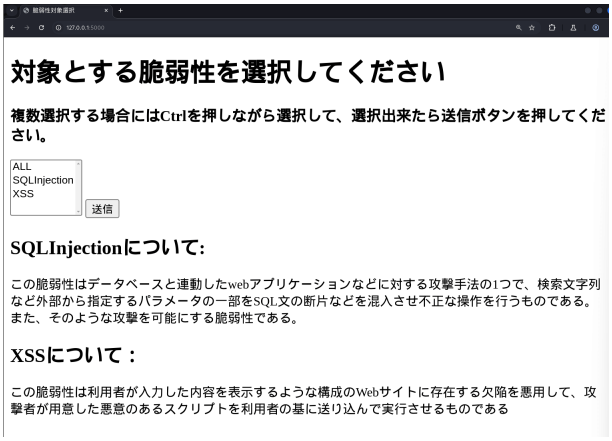


図 3　脆弱性選択画面

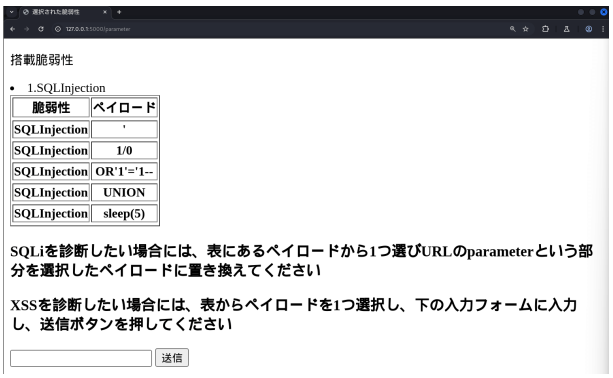


図 4　脆弱性選択後の画面

4.2.2　脆弱性の有効化

送信ボタンをクリックすると, 図 4 のような画面が表示される. 図 4 に表示されるように, 有効化した脆弱性名称とそれに対応するペイロードが表示される. これによりユーザーが有効化した脆弱性とそのペイロードを一目で確認できるようになり, 特定の脆弱性に対するペイロードがどのようなものがあるか把握することができる. また, 脆弱性情報 DB の中身は図 5 のようになっており, 図 2 と比較

```
MODE,脆弱性名称,ペイロード,
脆弱性レスポンス,
正常レスポンス,ステータスメッセージ,ステータスコード

1,SQLInjection,',
SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check
the manual that corresponds to your MariaDB server version for the right syntax to use near ''''' ,
問題ありませんでした,Internal error,500

1,SQLInjection,OR'1'='1--,
SQLInjection error,
問題ありませんでした,Internal error,500

0,XSS,""">'><s>XSS",
<s>XSS' SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax;
check the manual that corresponds to your MariaDB server version for the right syntax to use near
'XSS'' at line 1</s>,
脆弱性がなかったためエスケープされました。,Internal error,500

0,XSS,""">""><script>alert(document.cookie)</script>""",
SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check
the manual that corresponds to your MariaDB server version for the right syntax to use near '>'',
脆弱性がなかったためエスケープされました。,Internal error,500
```

図 5 脆弱性情報 DB の中身の変化

すると有効化した脆弱性名称と同じ脆弱性名称の MODE 列が 0 から 1 に変更されている.

4.2.3 リクエストの送信

図 4 で表示されたペイロードを URL パスの「parameter」またはフォームに入力し, 送信すると脆弱性が有効な場合脆弱性レスポンスが表示される. 図 6 に示すように, SQLi の脆弱性を有効化し, ペイロードとしてシングルクォーテーション「'」を送信した場合, SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ”””’ という脆弱性レスポンスが返却されることが確認できた. また, 図 7 に示すように, 脆弱性を無効化した場合には正常レスポンスが返却されることも確認できた.

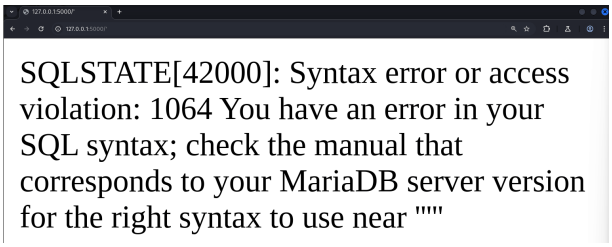


図 6 疑似レスポンスの表示

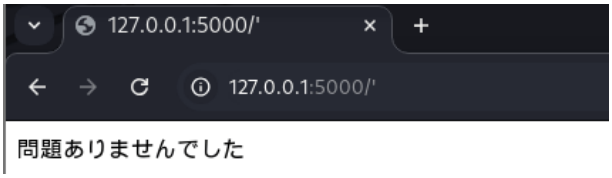


図 7 正常の表示

5. まとめ

本研究では, Web アプリケーション診断ガイドラインに基づき, 診断ツールの精度評価や検証に資することを目的として疑似脆弱性サーバーの構築を行った. 構築したサーバーに対して実際にペイロードを含むリクエストを送信し検証を行った結果, 今回搭載した SQLi と XSS の 2 種類の脆弱性について, 脆弱性の有効・無効状態によってレスポンスが変化することが確認できた. これにより, 既存の脆弱性サーバーとは異なり, 複数存在する脆弱性から有効化するものをユーザが自由に選択することが可能となった. 一方で, 本研究で実装した脆弱性は SQLi と XSS の 2 種類に限られており, Web アプリケーション診断ガイドラインに記載されている全ての脆弱性を網羅できていない課題がある. また, Flask は簡潔な記述で HTTP ベースの診断環境を構築できる利点があるものの, より低レイヤな通信制御を行うことは困難である. そのため, プロトコルレベルでの詳細な挙動再現や, 診断ツールとの通信制御の柔軟性を高める観点から, Flask 以外のフレームワークの採用も検討する必要がある. 今後の課題として, Web アプリケーション診断ガイドラインに記載されている脆弱性を網羅的に診断可能とするために, 疑似脆弱性サーバーに搭載する脆弱性の種類を段階的に拡充していく予定である. さらに, Flask の代替としてより低レイヤな制御が可能な TCP/IP ソケットサーバーの実装を検討し, Flask では実現できないプロトコルレベルでの脆弱性再現環境の構築を目指す. これにより, 診断ツールの検証精度向上だけでなく, ツール動作の詳細な動的挙動の分析や教育・演習用途での活用も可能となることが期待される.

謝辞
本研究の一部は，JSPS 科研費 23K11108 の支援により行った.

参考文献

[1] M. Noman, M. Iqbal, K. Rasheed and M.M. Abid, “Web Vulnerability Finder (WVF): Automated Black-Box Web Vulnerability Scanner,” Proceedings of the 2020 International Conference on Information Technology and Computer Communications (ITCS), pp. 1-6, 2020.

[2] P.J.C. Nunes, J. Fonseca and M. Vieira, “phpSAFE: A Security Analysis Tool for OOP Web Application Plug-ins,” Proceedings of the 45th Dependable Systems and Networks (DSN), pp. 1-6, 2015.

[3] R. Vibhandik and A.K. Bose, “Vulnerability Assessment of Web Applications: A Testing Approach,” Proceedings of the 4th International Conference on e-Technologies and Networks for Development (ICeND), pp. 1-6, 2015.

[4] J.K. Reddy, D. Aman, B. Yashwanth, P. Sowmya, R. Srikanth and Sv Vasantha, “Enhancing Web Security with VulnGuard: An Advanced Vulnerability Scanning Approach,” Proceedings of the International Conference on Engineering, Technology and Management (ICETM), pp. 1-7, 2025.

[5] Robin Wood, “Damn Vulnerable Web Application (DVWA),” https://github.com/digininja/DVWA (Accessed 2025-07-14).

[6] H. Tokumaru, “badtodo,” https://github.com /ockeghem/badtodo/tree/main (Accessed 2025-07-16).

[7] kazu1130, “WebAppPentestGuidelines.pdf,” https://github.com/WebAppPentestGuidelines /WebAppPentestGuidelines/blob/master/ WebAppPentestGuidelines/ WebAppPentestGuidelines.pdf (Accessed 2025-07-14).