Web アプリケーションによるクロスサイト スクリプティング検査の提案と実装

中安 恒樹 † 山本 知典 † 上原 雄貴 ‡ 武田 圭史 †

† 慶應義塾大学環境情報学部
kouki@sfc.wide.ad.jp, lushe@sfc.wide.ad.jp, keiji@sfc.wide.ad.jp

‡ 慶應義塾大学大学院政策・メディア研究科
nakajima@sfc.wide.ad.jp

あらまし Web アプリケーションにおける脆弱性発生の防止には開発プログラムに既知の脆弱性が存在しないかを検査することが有効であるが, コスト等の制約で広く脆弱性検査が行われているとは言えない．本論文では,Web アプリケーションの脆弱性検査を Web アプリケーションで行い, 開発者が容易に利用可能な脆弱性検査システムを提案する．今回実装した Musket は, 検査対象の入力欄に文字列を自動入力, エスケープ処理の有無を確認することでクロスサイトスクリプティング脆弱性を検査する．Web アプリケーションから検査が可能となる為, セキュアなシステム開発に関し知識を持たない開発者であっても容易に検査を行うことが可能となる．

Proposal and implementation of Web application for Cross-site Scripting Inspection

Koki Nakayasu† Tomonori Yamamoto† Yuki Uehara‡ Keiji Takeda†

†Faculty of Environment and Information Studies, Keio University
kouki@sfc.wide.ad.jp, lushe@sfc.wide.ad.jp, keiji@sfc.wide.ad.jp

‡Graduate School of Media and Governance, Keio University
nakajima@sfc.wide.ad.jp

Abstract To prevent occurrence of vulnerability on Web application, it is important to inspect developing program for pre-known vulnerability. However, as usual, the inspections are not conducted because it takes huge cost. In this article, we propose an usable web application inspecting system for vulnerability, which is made as web application. Musket, which we implemented this time, inputs strings to the parameter of the Web application and checks whether the escape process is working to inspect for Cross-site Scripting vulnerability. Developers will be able to inspect with Web application, this means that the developers doesn't need profound knowledge of Secure-develop for the inspection.

1 はじめに

Web サイトの公開が容易になり，今日様々な Web サイトが公開されている．2007 年のインターネットコム株式会社の調査 [1] によると，8 割以上の企業が自社の Web サイトを運営している．その一方，Web サイトにおける脆弱性は社会的問題となっており，個人情報を扱う Web

サイトの脆弱性をついた攻撃による情報漏洩や，不正に書き換えられた Web サイトによって悪意のあるソフトウェアをダウンロードさせる攻撃等が多く発生している．実際に，2011 年 6 月には Sony Pictures が SQL インジェクションの被害を受け，およそ 5 万人の個人情報が流出する事件が発生した．[2] また著者の手動調査でも 2011 年に官公庁，金融機関を含め約 50 件の Web サイトにおいて脆弱性を発見した．

これら脆弱性の中でもクロスサイトスクリプティングは多く発見されており，情報セキュリティ白書 2008[3] によれば，Web サイトの脆弱性届出全体のうち，70%がクロスサイトスクリプティングおよび SQL インジェクションで占められている．ソフトウェア等の脆弱性関連情報に関する届出状況 (2011 年第二四半期)[4] を見ても，報告された 68%が同脆弱性と発表している．多くの Web サイトにおける脆弱性は公開前，公開後に検査を行うことで対策が可能である．しかし，以後の関連研究で述べるように，現在の脆弱性検査手法は複雑な設定をユーザ自身が行ったり，調査依頼に金銭的コストがかかるなど，個々人が行うには手軽とは言えず，小規模な開発やツールに対して，検査を行うことが困難な場合がある．

そこで，本論文ではクロスサイトスクリプティングに着眼点を置き，幅広い開発者に容易に利用可能な Web アプリケーション脆弱性検査ツール，Musket を提案し，実装した．以下で，まず既存の脆弱性検査手法について述べ，今回実装した Musket の詳細，既存手法との比較，評価，問題点を述べる．

2 既存の手法

現在，脆弱性検査の既存手法は二種類ある．自動化された検査と，手作業の検査である．

本項では各分野の脆弱性検査について述べる．

2.1 脆弱性自動検査

既存の Web アプリケーション脆弱性を自動検査する研究に，小菅らの提案する Amberate フレームワーク [5] がある．このフレームワークは，検査クライアントをホストにインストールし，実行するモデルが想定される．そしてユーザは，フレームワークにプラグインとして必要な検査を追加することで，様々な脆弱性の検査に対応することが可能である．既存の検査ツールには，Google[6] の提供する検査ツール ratproxy[7] が存在する．ratproxy はクライアントにインストールし，プロキシサーバとして動作する検査ツールである．プロキシサーバを経由し，ユーザが Web アプリケーションにアクセスすることで検査する．

また同様のツールとして，オープンソースの自動脆弱性テストフレームワーク w3af[8] が存在する．w3af はクライアントにインストールし，プラグインを組み合わせて設定を行い，対象の Web サイトに検査する．

しかし，これら既存の自動検査ツールは，ユーザが検査クライアントをダウンロードし，インストール，プラグインの組み合わせを始めとした設定が必要である．そのため，セキュリティやオペレーションに精通していない開発者が検査することは困難である．

2.2 手作業の検査

実際に作成中，作成後の Web アプリケーションにおける脆弱性を自ら手作業で確認する方法も広く行われている．作成者が多い場合，脆弱性検査の担当者を準備し検査を行うケースも考えられる．しかし，開発者が自ら確認する場合，検査漏れが発生することがあり，複雑な Web アプリケーションを網羅的に検査することができない．

また，既存の検査サービスには，企業の行う脆弱性検査サービスが存在する．これらは依頼者のアプリケーションに対して，企業のチームが手動，自動の検査ツールを利用して細かく検査を行う．また，有償の検査であり，個人を対象としたものではない．また検査について打ち合わせ，報告を行い情報を提供する．しかし，これらの調査を利用するには，ユーザに検査費用がかかり，一般のユーザが公開する Web サ

イトや小規模な Web サイトで利用することは困難である．

3 Musket

本論文では脆弱性検査 Web アプリケーション，Musket を提案，実装した．

3.1 概要

Musket は Web アプリケーションとして実装した，Web アプリケーション脆弱性検査ツールであり，自動的にクロスサイトスクリプティングを検査する．ユーザは利用時にブラウザを用いて Musket にアクセスし，所定の方法で検査対象 Web アプリケーションの URL を入力，検査を実行する．ユーザは利用に際して，検査対象の Web サイトにファイルをアップロードし，Musket にアクセスするだけで検査が可能で導入が容易である．検査認可を行うことで Web サイト管理者以外が他サイトへ攻撃に用いることを防止する機能を設けた．

3.2 検査の流れ

図 1 に示すように，本手法では以下の手順でユーザが対象 Web サイトの脆弱性検査を行う．ユーザは Musket にアクセスし，まずメールアドレスを入力，検査ツールを利用するためのトークンを取得する．次にユーザは取得したトークンを，検査対象の Web サイトにアップロードした後，Musket から検査対象の URL を入力する．

Musket はトークンを取得，Musket のデータベースと照合すると，実際に指定されたページを取得し，ページ中のフォーム (form タグ) を検出する．フォームを検出すると，そのフォームの送信先 (action 属性)，メソッド (method 属性) を取得する．POST メソッドで送信するフォームの場合，curl 関数群でパラメータを設定，GET メソッドの場合は action 先 URL にパラメータを追加して実行する．検出したフォーム内のパラメータには事前に用意したクロスサイトスクリプティングに用いられる記号類を含んだクエリを投稿する．

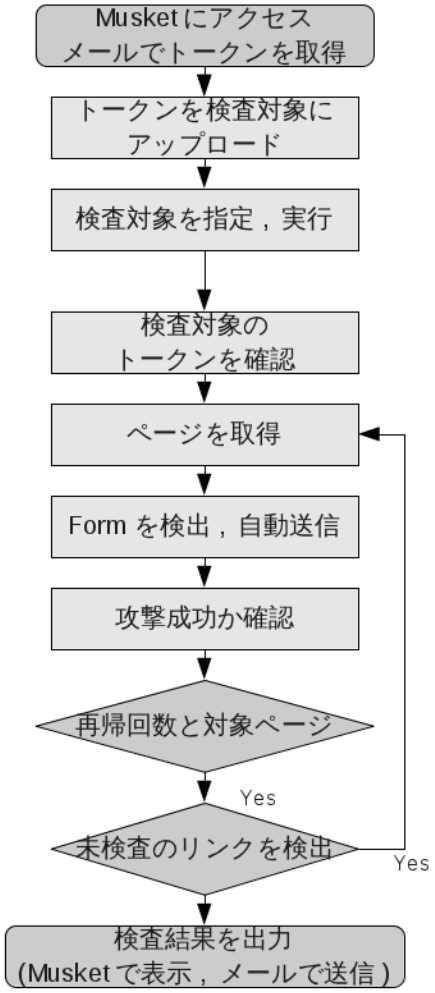

図 1: Musket の動作フロー

クエリには表 1 の通り，HTML タグを用いるもの，記号類のみを用いるもの，実際に iframe タグを用いてページを埋め込むもの，スクリプトを送信するものを用意している．ユーザは検査時にクエリを選択する事が可能であり，独自のクエリを作成し検査を行うことも可能である．Musket はこれらの選択したクエリが，投稿後のページに含まれているか PHP の strpos 関数を用いて確認し，検出された場合は攻撃に成功したと判断し，検査する．

Musket は取得したページの検査が終わると，次に同ドメイン内へのリンクを辿り，再帰的に検査を行う．その後，設定された再帰回数に達

するまで検査を行い，検査対象を自動的に検査することができる．検査を行ったページは配列に保存することで，同ページを複数回確認せず検査を効率化している．

検査が終了すると，図 2 のように検査結果は

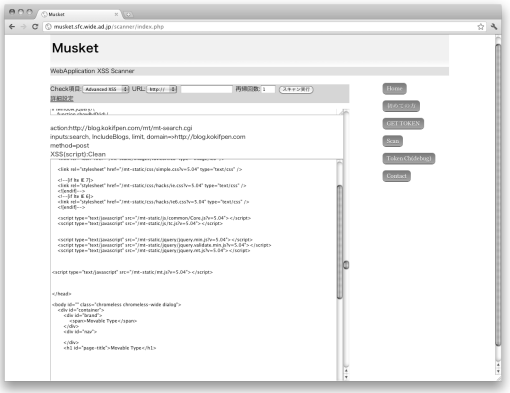

図 2: 検査結果の表示

ブラウザで表示され，ユーザは検出されたページ，送信したクエリを確認できる．また，あらかじめログの送信を設定すれば，検査のログをメールで送信可能である．

表 1: クエリ

Name	Query
Tag (s)	”><s>musket</s>
Symbol	”;–musket–
Iframe	<iframe class=musket>
Script	<sctipt>alert(”xss”);</script>

3.3 検査認可の仕組み

Musket は能動的にサーバから検査対象に自動ポストするシステムであるため，検査対象の管理者以外が利用できないよう検査認可の仕組みが必要である．管理者以外がが用いることで脆弱性を発見され攻撃に利用されるリスクが発生するからである．そこで Musket はユーザにメールアドレスの入力を求め，そのアドレス宛にトークンを発行する．トークンを検査対象に設置していなければ検査が認可されず，Musket が第三者によって悪用されることを防止する．また，Musket はトークンに 30 分の有効期限を設けることで，トークンが放置された場合も第三者に利用されるリスクを減少させる．

3.4 実装

本ツールは PHP と MySQL で実装を行っており，CentOS6 のサーバ上で運用を行っている．PHP 上で検査対象を読み取る際には，curl 関数群を利用し，パラメータを含めリクエストを送信する．取得した内容は PHP Simple HTML DOM Parser ライブラリ [9] を用いて，解析する．また，Web アプリケーションとして Musket を公開することで，誰もが無償で利用可能なアプリケーションとした．

4 評価

本手法では，ユーザが検査を行う際に Web アプリケーションを用いて検査を行うため，従来の手法よりもユーザに要求するスキルやコストを大幅に削減した．既存手法で述べた各手法と比較しても関連ユーザは Web アプリケーションにアクセスするのみで調査可能であり，表 2 にあげるように，無償で検査可能な上，操作，設定上でのコストも小さい．また，実際にそれぞれの手法を利用する際に，ユーザが行う作業ステップも削減した．Musket 利用時，ユーザはトークンの取得，アップロード，Web 上での検査の 3 ステップを実行すれば検査可能である．しかし，他のツールを用いた手法である Amberate[5] は，クライアントソフトウェアのインストール，ソフトウェアの設定，プラグインの導入，検査の 4 ステップが必要である．同様のツールである，w3af を利用する際は，ソフトウェアのインストール，実行環境の構築，設定，実行の 5 ステップが必要である．また，Google[6] の提供する ratproxy[7] の場合，プロキシサーバソフトウェアのダウンロード，ratproxy の設定，ブラウザの設定，アクセス，ログ解析の 5 ステップが必要である．

次に，ユーザに求められるスキルを比較する．他の手法のうち，ユーザのみで行えるものではユーザ自身が設定，調査を行う必要がある中で，

Musket は Web 閲覧と構築の知識のみで検査が可能である．検査にかかる日数/時間を比較すると，手作業で行う調査や外部に調査を依頼する場合，規模に応じて数日から数週間の時間が必要である．Musket は他の脆弱性検査ツールと同様に自動的な検査が可能なため，比較的短時間ですべての検査を実施可能だ．また，金銭的コストにおいて，検査を外部に依頼する場合，数万円から数十万円のコストがかかるが，Musket は無料で公開するため，ユーザに金銭的コストはかからない．

ただし，現状では Musket はクロスサイトスクリプティングのみの検出を目的とし，他の脆弱性の対応は対象としていない．しかしこれに対し，他の手法では Web アプリケーションの他のリスクに対応している．

また実際に，配布，利用されている掲示板システムを筆者のサーバ上で稼働させた上で，Musket を用いて検査を行った．結果，3 システム検査したうち，1 システムでクロスサイトスクリプティングが検出，確認された．

表 2: 既存検査手法との比較

項目	Musket	ツール	手作業	サービス
スキル	○	×	×	○
日数	○	○	×	×
金銭	○	○	○	×
項目	△	○	△	○

5 本手法の課題

5.1 検査可能項目

評価の項目で述べたように，本ツールで検査可能な項目は，クロスサイトスクリプティングのみである．また，Musket は単純にパラメータにスクリプトが挿入されて被害が出る脆弱性を想定しており，ページ内でパラメータがやりとりされる中で発生するクロスサイトスクリプティングや，SQL インジェクション，セッション管理における問題など別種類の脆弱性が存在する．これらの脆弱性を検出するためには，Musket をさらに拡張する必要があり，対応させる予定である．

5.2 効率化

本ツールを作成した際，検査の効率化，高速化を行う為，同ページの検査を複数回行わない設計を行った．また，再帰回数を設定することができることで必要な検査を行うことができる．ただし，現状では検査対象の全ページを PHP の Curl 関数を利用し，取得を行うため，検査に時間がかかっている．今後ユーザの増加や検査対象が拡大すれば，検査の効率化がさらに必要となる．

6 まとめ

Web アプリケーションが拡大し，脆弱性による被害も拡大している．現状の検査手法では，検査のコストが大きく Web アプリケーションの検査が広く，十分に行われているとは言えない．本論文では，セキュア開発に精通していない開発者でも利用可能な Web アプリケーション脆弱性検査手法を提案した．この Web アプリケーションによる検査システムを公開，運用することで開発者は作成した Web アプリケーションの検査にかかるコストを下げることができた．しかし，現状ではクロスサイトスクリプティングのみの検査であるため，今後 Musket を拡張することでさらに多岐に渡る脆弱性を検査可能にする．

参考文献

[1] 8 割以上の企業が Web サイトを持つが、ログ解析は 5 割以下 http://japan.internet.com/research/20070709/1.html

[2] Sony investigating another hack http://www.bbc.co.uk/news/business-13636704

[3] 情報処理推進機構 ”情報セキュリティ白書 2008”（実教出版，2008）

[4] ソフトウェア等の脆弱性関連情報に関する届出状況 2011 年第 2 四半期 http://www.ipa.go.jp/security/vuln/report/documents/vuln2011q2.pdf

[5] 小菅祐史 河野健二 ”効果的な攻撃テストによる Web アプリケーションの脆弱性検出手法” (IPSJ，2009)

[6] Google http://www.google.com

[7] ratproxy - passive web application security assessment tool http://code.google.com/p/ratproxy/

[8] w3af - Web Application Attack and Audit Framework http://w3af.sourceforge.net/

[9] PHP Simple HTML DOM Parser http://simplehtmldom.sourceforge.net/

[10] 総務省 平成 23 年版 情報通信白書

[11] 後藤峰陽 ”PHP サイバーテロの技法 攻撃と防御の実際” (ソシム，2005)

[12] 佐名木智貴 ”セキュア Web プログラミング Tips 集” (ソフト・リサーチセンター，2008)