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

2003－ＱＡＩ－7　(6)
2003／5／21

クロスサイトスクリプティング脆弱性の自動判定・収集システムの提案と実装

高橋　裕樹†　オマール　イスマイル†　門林　雄基†　山口　英†

† 奈良先端科学技術大学院大学 情報科学研究科
〒630–0192 生駒市高山町 8916-5
E-mail: †{yuuki-ta,isumai-u,youki-k,suguru}@is.aist-nara.ac.jp

あらまし　Cookie のみでセッション管理を行っているウェブサイトでは，Cross-Site Scripting (XSS) 脆弱性を利用した攻撃によって Cookie が漏洩し，Cookie に関連づけられた個人情報が漏洩の危険にさらされる．この問題に対し，Web サーバ側での XSS 対策手法が存在するが，XSS 脆弱性の危険性は軽視されており，また運用面での負担が大きいことから，これらの既存の対策手法には限界がある．本論文では，クライアントが Web ページに入力した情報を利用して XSS 脆弱性の有無を自動判定し，XSS 脆弱性を持つ Web ページの情報を収集，共有するシステムを提案する．提案システムにより，XSS 脆弱性を狙った攻撃からユーザを保護することが可能となる．また，収集した脆弱性情報によって，XSS 脆弱性を保持するサーバに対して警告を発することも可能となる．

キーワード　HTTP, プロキシ，Cookie, クロスサイトスクリプティング, 脆弱性検査

A Proposal and Implementation of Automatic Detection/Collection System for Cross-Site Scripting Vulnerabilities

Yuuki TAKAHASHI†, Omar ISMAIL†, Youki KADOBAYASHI†, and Suguru YAMAGUCHI†

† Graduate School of Information Science, Nara Institute of Science and Technology
Takayama-cho 8916–5, Ikoma-shi, Nara, 630–0192, Japan
E-mail: †{yuuki-ta,isumai-u,youki-k,suguru}@is.aist-nara.ac.jp

Abstract　Cross-site scripting (XSS) attacks target web sites with Cookie-based session management, resulting in the leakage of privacy information. Although several server-side countermeastures for XSS attacks do exist, such techniques have not been applied in a universal manner, because of their deployment overhead and the poor understanding of the XSS problem. This paper proposes a client-side system that automatically detects XSS vulnerability by manipulating either client request or server response. The system also shares the indication of vulnerability via central repository. The purpose of the proposed system is two-fold: to protect users from XSS attacks, and to warn web servers with XSS vulnerabilities.

Key words　HTTP, Proxy, Cookie, Cross-Site Scripting, Vulnerability testing

1. はじめに

Hyper Text Transfer Protocol (HTTP) は様々な商取引や情報交換のインタフェースとして利用されている．しかし，HTTP はコネクション間で状態を保持しないプロトコルであるため，コネクションをまたがるセッション管理を行えない．そこで，原始的なセッション管理手法として Cookie が導入された．

Cookie はサーバから与えられたクライアント識別情報で，これを用いてコネクションをまたがるセッション管理を実現することができる。まず，クライアントはユーザ登録や認証などの手続きを経たあと，サーバから与えられた Cookie をファイルとして保持する．次に，クライアントが当該 Web サーバに再度アクセスする時に，対応する Cookie を送ることで，既存の認証情報を再利用してユーザの識別を行う．

Web サーバがパスワードなどによるユーザ認証を省略し，Cookie のみでユーザを識別した場合，Cookie の漏洩はクライアントに対して「なりすまし」の危険を生じる可能性がある．そのため，Cookie への外部アクセスを禁止する仕様や，Secure Socket Layer (SSL) による通信路の暗号化によって Cookie ファイルは保護されている．ところが，Cross-Site Scripting (XSS)

と呼ばれる Web の脆弱性を利用した攻撃では，悪意ある攻撃者が Cookie の保護機能を回避し，Cookie を入手することが可能となる．

このような XSS の問題に対し，サーバ側とクライアント側のそれぞれでの対策手法が提案されているが，XSS 脆弱性の軽視やコスト負担増といった運用面での問題によって，サーバ側での対策は進んでおらず，XSS 脆弱性のある Web ページは依然として多数存在している [1]．そこでクライアント側での XSS 対策が必要となるが，クライアントは，意図的に混入されたスクリプトと Web サーバが返す正当なスクリプトの判別ができないことが問題となる．

そこで本論文では，ユーザが入力した文字列の情報から Web ページにおける XSS 脆弱性の有無を自動で判定し，XSS 脆弱性の存在する Web ページの情報を収集し，その脆弱性情報を共有データベースへ登録するシステムを提案する．本手法により，クライアント側で XSS 脆弱性を保持する Web ページの情報を共有することができ，ユーザを XSS 脆弱性から保護することが可能となる．また，こうした情報を共有することで，XSS 脆弱性を保持するウェブサイトへの警告もはかることができる [2]．

2. クロスサイトスクリプティング (XSS) 問題

2.1 XSS とは

2000 年 2 月 2 日，XSS が CERT Coodination Center から脆弱性として勧告された [3]．XSS とは，HTTP リクエストに混入されたスクリプトが HTTP レスポンスにそのまま含まれてしまう脆弱性で，クライアントでそのスクリプトが実行されてしまうことで様々な危険性が生じる．すなわち，外部からのスクリプト入力を Web サーバ側で無条件に受け付け，そのスクリプトを含んだ出力をエスケープ処理を施さずにクライアントへ返す Web ページに XSS 脆弱性が存在する．

XSS の影響を受けるのは，HTTP リクエストに含まれるパラメータから動的にコンテンツを生成する Web ページが該当し，検索エンジン，掲示板，Web メール，e コマースサイトなどが挙げられる．XSS によって生じる危険性の中で，最も問題視されているものが Cookie の漏洩である．

2.2 XSS による Cookie 漏洩

XSS の原理について図 1 を用いて説明する．図 1 の a は XSS の被害を受けるユーザ，b は悪意ある第三者によってリンクが仕掛けられた Web ページ (図 3)，c は XSS 脆弱性を有する Web ページである．(2) の レスポンス受信時にはスクリプトは実行されないが，ユーザが仕掛けられたリンクをクリックした場合に，スクリプトは (3) で示すように標的 Web ページへのリクエストに埋めこまれて送信される．そして，返信される (4) のレスポンスにはそのスクリプトが効果を発揮する形で含まれ，ブラウザはそのスクリプトを受信して実行する．このとき，スクリプトの内容によっては Cookie の漏洩など，ユーザは不利益をこうむる．

JavaScript では，文書中にスクリプトを埋め込むために HTML タグの <SCRIPT> タグを利用する．ブラウザに対して SCRIPT タグを HTML タグとして解釈させることができれば，そのタグ内の JavaScript コードをスクリプトと解釈させて実行させることができる．

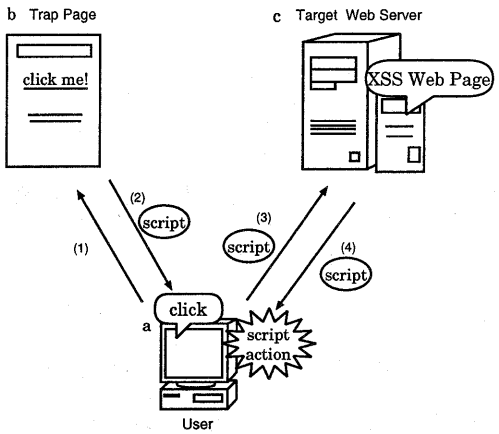

図 1 XSS の原理

```
<SCRIPT>document.write(document.cookie)</SCRIPT>
```

図 2 検索ページに入力する文字列

```
http://xss.jp/search.cgi?keyword=
<SCRIPT>document.cookie(document.cookie)</SCRIPT>
```

図 3 XSS 脆弱性ページへのリンク

例えば，検索ページのテキストエリアに，検索文字列として図 2 のような文字列を入力したとする．ここでは，JavaScript で document.cookie を指定し Cookie データの読み出しを試みている．

XSS 脆弱性が存在する Web ページでは，SCRIPT タグが HTML タグとして解釈され，混入されたスクリプトが実行される．ユーザが図 3 に示すスクリプトを埋めこまれた標的 Web ページへのリンクをクリックした場合にも，そのスクリプトを含んだ出力結果がユーザへ送信され，スクリプトが実行される．

ブラウザは HTTP レスポンスに含まれる HTML タグやスクリプトは，全て HTTP レスポンスを返した Web サーバで発行されたものだと判断するため，Cookie へアクセスするスクリプトの実行を許可する．外部から Cookie へのアクセスが可能となるため，Cookie は任意の第三者の元へ送信され，漏洩する．

2.3 既存の XSS 対策と問題点

クライアント側では，レスポンス内に含まれるスクリプトが外部から混入されたスクリプトか，Web サーバが意図して返すものかを判断することは難しい．しかし，サーバ側では，HTTP リクエストにスクリプトが混入されていた場合，外部から意図的に入力されたと判断できるため，XSS 対策はより容易である．既存の XSS 対策として知られている手法はすべてサーバ側における対策であり，Web サーバ，CGI プログラ

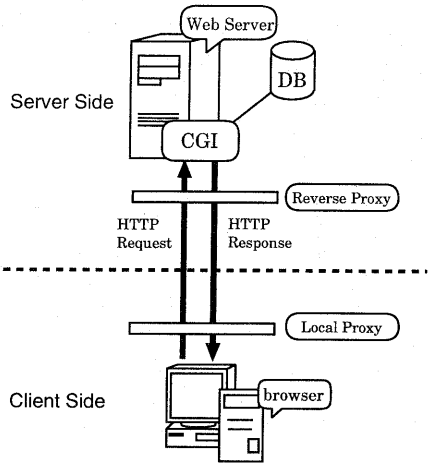

図 4　XSS の対策可能となる場所

表 1　特別な意味を持つ文字のエスケープ処理

特殊な意味を持つ文字	エスケープ処理後の文字列
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&#39;

ム，リバースプロキシなどの箇所での対策手法が提案されている (図 4).

Web サーバが有する XSS 脆弱性は，Web サーバが動的に生成して送信する HTTP エラーページなどのようなページで顕在化する．しかし，こうした脆弱性は，ベンダーから提供される修正ファイルの適用や Web サーバのバージョンアップで対策可能である．CGI プログラムでは，表 1 に示すように入力パラメータの特殊文字にエスケープ処理を施すことができる．図 5(A) のスクリプトがエスケープ処理された場合 (B) の文字列に変換され，スクリプトは実行されない．また，HTTP リクエストを検査し，スクリプトをエスケープ処理したりエラーをブラウザに返すリバースプロキシも提供されている．これらの対策は，管理下のホストが XSS 脆弱性を持つことの危険性を判断して，管理者が積極的に行う対策として有効である．

しかし，エスケープ処理を施さない Web ページやバージョンの古い Web サーバは多数存在する [1]．また，Web サーバに XSS 脆弱性があることを忠告されても対策が施されない事実も報告されている．これは，XSS がサーバ側の脆弱性でありながら，ユーザが被害を受けることによる XSS 脆弱性の軽視や，対策へのコスト負担といった運用面の問題が大きいためである．このようにサーバ側での充分な対策は期待できない．そのため，クライアント側での XSS 対策が必要となるが，既存の技術では有効な XSS 対策が実現できない．

2.4　クライアント側における XSS 対策

ブラウザの設定項目には，信頼サイトを登録する機能があり，信頼サイトに登録していない Web ページのスクリプトの

```
(A) <SCRIPT>document.write(document.cookie)
    </SCRIPT>
(B) &lt;SCRIPT&gt;document.write(document.cookie)
    &lt;/SCRIPT&gt;
```

図 5　エスケープ処理前 (A) と処理後 (B) の比較

表 2　エスケープ処理 2 回による影響

特殊文字	エスケープ 1 回	エスケープ 2 回	ブラウザ表示
&	&amp;	&	&amp;
<	&lt;	<	&lt;
>	&gt;	>	&gt;

実行を許可しないよう設定できる．しかし，XSS 脆弱性が信頼サイトのページに存在する場合，スクリプトを信頼サイトへの HTTP リクエストへ混入させることでスクリプトが実行されてしまう．XSS 脆弱性をもつ Web ページを信頼サイトとして登録することを避ければ，クライアント側における有効な XSS 対策が可能となる．

このほか，ローカルプロキシを利用した方式として，HTTP リクエストに含まれる特殊記号を全てエスケープ処理した後に，Web サーバへ送信する手法がある．しかし，サーバ側においてエスケープ処理による XSS 対策が施されていた場合，2 度のエスケープ処理が行われる可能性がある (図 2)．その結果，サーバへの入力が，本来の入力とは異なるものに変化し，入力時とは異なる文字列が表示されるなど，サーバからの出力に不具合を与える可能性がある．これを避けるために，XSS 脆弱性をもつ Web ページへの HTTP リクエストのみにエスケープ処理をすることで，クライアント側における有効な XSS 対策が可能となる．

このように，クライアント側での対策には，XSS 脆弱性を持つ Web ページに関する情報が必要となる．しかし，既存の XSS 検査ツールはサーバ側のコンテキストに対応できないという問題がある．XSS 検査ツールは Web ページの入力フォームが入力を期待している文字列を理解することが出来ず，それらのツールによるフォームへの制限を無視した入力によって，Web ページはエラーを返すことがある．ユーザによる手動の XSS 脆弱性検証では，Web ページに書かれている入力禁止文字や入力文字数の制限範囲などの情報を参考にすることでエラーページに示されるエラーの原因を把握できる．そのため，人間の手作業による Web アプリケーション検査サービスが多く存在する [4]．しかしながら，Web ページの数は膨大で，すべてを人間の手作業で検査することは不可能である．

2.5　提案手法による解決策

本論文では，ローカルプロキシにおいて，ユーザが Web のコンテキストに応じて入力した情報から，自動的に Web ページにおける XSS 脆弱性の有無を判定し，脆弱性を発見した Web ページの情報を収集するシステムを提案する．

ただし，ローカルプロキシでの収集では，自らが発見した XSS 脆弱性情報しか関知しない．そこで，ローカルプロキシ

は発見した XSS 脆弱性情報をデータベースサーバへ登録する．データベースサーバは，各ローカルプロキシに対して収集した XSS 脆弱性情報を提供する．これにより，収集した全ての XSS 脆弱性情報をローカルプロキシが共有することができる．

3. XSS 脆弱性の自動判定・収集システムの設計

3.1 レスポンス変更型プロキシ

XSS の特徴を考慮し，HTTP リクエストに特殊文字を含むパラメータが含まれていた場合，プロキシでパラメータの情報を保持する．対応する HTTP レスポンスから保持されていた文字列を検索し，発見された場合は，XSS 脆弱性を持つと判断する．サーバ側で特殊文字がエスケープ処理されているか削除されている場合には発見されないので，その場合は XSS 対策済みと判断する．XSS 脆弱性判定のためには HTTP リクエストのパラメータに特殊文字を含んだ文字列を含める必要がある．Web のコンテキストに対応した検査が必要なので，ユーザは特殊文字を含めた文字列を入力する．ただし，XSS 脆弱性判定を行うには，XSS 脆弱性を持つパラメータ入力と持たないパラメータ入力が同じ Web ページに複数存在することを考慮し，パラメータ毎に判定する必要がある．

また，悪意ある第三者がリンクとスクリプトを用意し，ユーザーがそのリンクをクリックした場合，そのスクリプトがユーザのブラウザで実行される危険性がある．そのため，XSS 脆弱性が発見された場合に HTTP レスポンスから，特殊文字の文字列を検索し，すべてエスケープ処理を行うことでスクリプトを無効化してブラウザに送信する．さらに，ユーザに XSS 脆弱性が発見されたことを明示的に伝えるための警告画面を出力する HTML タグを追加する．

以上で述べた提案方式を本論文ではレスポンス変更型プロキシとするが，ユーザがパラメータに含める特殊文字によっては問題が生じる．まず，HTTP リクエストの入力パラメータに "<" や "<html>" のような特殊文字列が設定された場合，これらの文字列は多くの Web ページに含まれるため，XSS 脆弱性を持つページと誤判定する．

そこで，ユーザがパラメータに入力する文字数が 10 文字以上であれば XSS 脆弱性判定を行う．10 文字以上に設定すれば，HTTP レスポンスにて一般的に含まれる文字列と一致する可能性が低くなる．また，悪意ある第三者が仕掛けたリンクのパラメータには 10 文字以上必要であるため，クリックした際にも XSS 脆弱性の判定が可能となる [6]．次に，複数のパラメータに特殊文字を含めた同じ文字列が入力された場合，どのパラメータに XSS 脆弱性が存在しているか判別できないため，ユーザには各パラメータ入力欄毎に異なる文字列を入力してもらう必要がある．

3.2 リクエスト変更型プロキシ

レスポンス変更型プロキシの入力文字列制限の問題に対処するリクエスト変更型プロキシについて述べる．まず，HTTP リクエストごとにランダムな三桁の数字を生成し，特殊文字が含まれたパラメータごとに 1 ずつ加算した数字を特殊文字の後に追加する．文字列の変更前と変更後の例を表 3 に示す．表 3 の

表 3 リクエストパラメータの変換

パラメータ名	変更前の文字列	変更後の文字列
parameter1	<s>test</s>	<234s>234test<234/s>234
parameter2	<s>test</s>	<235s>235test<235/s>235
parameter3	<	<236
parameter4	<html>	<237html>237

表 4 データベースに格納される情報

要素	内容
hostName	XSS が存在するホスト
pathName	XSS が存在するパス
request	XSS 発見時のリクエスト
parameterName	XSS が存在するパラメータ名
parameter	XSS 発見時のパラメータの内容
date	HTTP レスポンス発行時間
cookieScript	Cookie へのスクリプト混入の有無
deleteFlag	脆弱性情報か修正情報
accessTime	データベース更新時間情報

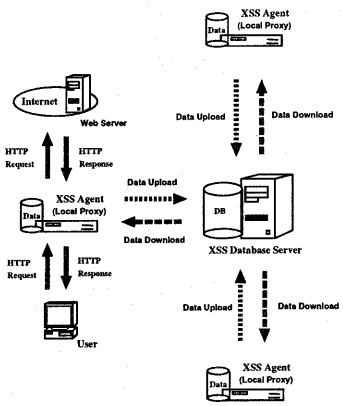

図 6 XSS 脆弱性の自動判定・収集システム

parameter 1, 2 の例では，パラメータ毎の XSS 脆弱性の判定が可能となる．また，parameter 3, 4 の例では，HTTP レスポンスに静的に含まれて誤判定される問題が解決される．

ここで，数字が付加されたパラメータに対するレスポンスを，ユーザにそのまま返すわけにはいかない．そこで，XSS 脆弱性判定用とブラウザへ返すレスポンス獲得用の合計 2 回，リクエストを送信する必要がある．

リクエスト変更型プロキシの問題として，サーバ側におけるパラメータの文字列の長さによる制限によりエラーが返される可能性がある．しかし，特殊文字ごとに 3 文字数が追加されることを考慮して入力文字数を調整すればよい．

3.3 XSS 脆弱性情報の収集

ローカルプロキシで取得する脆弱性情報は，XSS 脆弱性が発見されたホスト，パス，パラメータ名に，発見された時の証拠を加えた情報である．(図 4)

図 6 は XSS 脆弱性の自動判定・収集システムの概要である．ローカルプロキシは発見した XSS 脆弱性情報をデータベースサーバへ送信し，収集された XSS 脆弱性情報が各ローカルプロキシに返されることにより，XSS 脆弱性情報を共有する．

次に，ローカルプロキシとデータベースサーバの通信方式にて，各要素から XML データに変換されるまでの概要を図 7 に示す．プロキシからデータベースサーバへの送信形式は，脆弱性情報の送信 (addData)，脆弱性修正情報の送信 (removeData)，

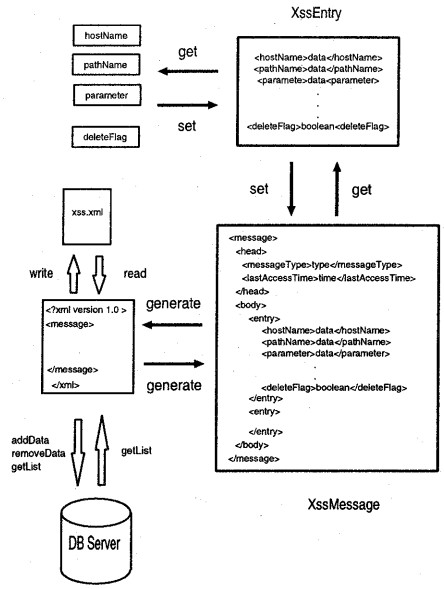

図 7　XSS 脆弱性情報の各データ方式

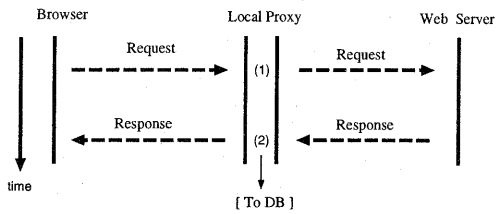

図 8　レスポンス変更型プロキシ

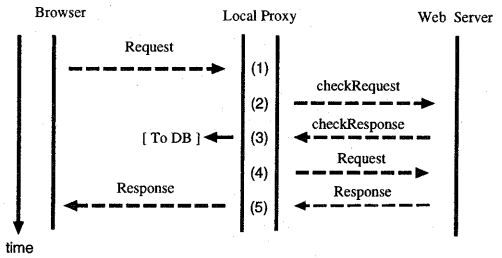

図 9　リクエスト変更型プロキシ

脆弱性更新情報の要求 (getList) の 3 形式とした.

4. XSS 脆弱性の自動判定・収集システムの実装

4.1 概　　要

提案システムを，Vine Linux (kernel 2.4.18) 上で実装した．脆弱性情報を格納するデータベースは PostgreSQL 7.2，開発言語には Java(J2SDK 1.4.1) を使用し，産業技術総合研究所 [5] の高木浩光氏から提供されたローカルプロキシ vulcan に，XSS 脆弱性の自動判定・収集機能を追加した．判定方式として，リクエスト変更型とレスポンス変更型の 2 種類を実装した．

本実装では起動時にデータベースサーバへ脆弱性更新情報の要求を送信し，脆弱性情報を取得する．

4.2 レスポンス変更型プロキシの処理

図 8 にレスポンス変更型プロキシの動作概要を示し，各番号で示される部分の処理を以下に示す.

（1） リクエストの受信

リクエストを検査し，リクエストのパラメータ部分に特殊文字が含まれていた場合にはリクエストの情報を保存しておく．

（2） レスポンスの受信

保存されていたリクエストに対応するレスポンスを受信すると，リクエスト時に格納されていたパラメータと同じ文字列を検索し，含まれていれば XSS 脆弱性ありとする．XSS 脆弱性検査の結果に応じて，リストにない新しい情報であれば，データーベースサーバへ脆弱性情報，修正情報を送信する．つぎに，レスポンスに含まれているパラメータと同じ文字列は，すべてエスケープ処理し，警告画面を出力する HTML タグを追加し，ブラウザに送信する.

4.3 リクエスト変更型プロキシの処理

図 9 にリクエスト変更型プロキシの動作概要を示し，各番号の処理を以下に示す.

（1） リクエストの受信

レスポンス変更型プロキシ同様に，リクエストを検査する.

（2） 検査リクエストの作成と送信

リクエストに含まれるパラメータ値のみ変更する．リクエストごとにランダムな数字を作成し，パラメータごとにプラス 1 文字ずつ加えた値を各パラメータの特殊文字に追加後，サーバ側へ送信する.

（3） 検査レスポンスの受信と処理

レスポンス変更型同様に，XSS 脆弱性の判定を行い，判定結果に応じてデータベースサーバへ情報を送信する.

（4） リクエストの変更と送信

脆弱性が発見された場合，リクエストに含まれる特殊文字が含まれるパラメータは，全てエスケープ処理されたパラメータに変換後 Web サーバへ送信する．発見されなかった場合には，リクエストは変更せずに Web サーバへ送信する.

（5） レスポンスの受信と処理

警告画面を出力する HTML タグを追加し，ブラウザに送信する.

4.4 XSS データベースサーバ

XSS データベースサーバは，ローカルプロキシの送信タイプに応じた以下の処理をする.

- addData

データベースに書き込まれる時間の情報を脆弱性情報に追加し，データベースに追加する.

- removeData

データベースに書き込まれる時間の情報を修正情報に追加し，データベースの脆弱性情報を修正情報に更新する.

- getList

ローカルプロキシが前回データベースにアクセスした時間と

表 5　レスポンス変更型プロキシの評価結果

	パラメータ数	脆弱性判定	情報収集	無効化
GET	1	○	○	○
	2	△	△	○
POST	1	○	○	○
	2	△	△	○

表 6　リクエスト変更型プロキシの評価結果

	パラメータ数	脆弱性判定	情報収集	無効化
GET	1	○	○	○
	2	○	○	○
POST	1	○	○	○
	2	○	○	○

比較し，更新情報をすべて取りだし，ローカルプロキシに送信する．

5. XSS 脆弱性の自動判定・収集システムの評価

5.1 テスト環境における評価

XSS 脆弱性情報自動判定・収集システムを評価するため，XSS 脆弱性を有する Web ページを用意し，パラメータに特殊文字やスクリプトを混入した様々な HTTP リクエストを送信して実装の評価を行った．評価基準は Web ページにおける XSS 脆弱性の有無の判定，XSS 脆弱性及び修正情報の収集，及び XSS 脆弱性が発見された場合，スクリプトを無害化可能とした．

表 5 にレスポンス変更型プロキシの評価一覧を，表 6 にリクエスト変更型プロキシの評価一覧を示す．各表においては，○が評価基準を満たしていることを示し，△は特定の条件において評価基準を満たさなかったことを示す．

レスポンス変更型プロキシにおいて，複数のパラメータに特殊文字を含む文字列を挿入した場合，XSS 脆弱性判定および脆弱性情報の収集における評価基準を満たせない場合があり△とした．これは 3 章で指摘した，特定の条件の場合パラメータごとの正確な脆弱性判定を行えない問題である．この問題に対処すべく新たに提案したリクエスト変更型プロキシを使用することにより解決されることが表 6 により示された．

5.2 実環境における評価

XSS Blacklist [7] に登録されている Web ページのうち実際に XSS 脆弱性が存在していることを確認した 32 の Web ページを用いた．XSS Blacklist に掲載されている，パラメータにスクリプトを混入した URL を用いて実装の評価を行った．

表 7 に，レスポンス型及びリクエスト変更型プロキシの実環境における評価を示す．表における○は，前節で示した評価基準を満たしていることを示し，×は満たさなかったことを示す．

表 7 のように，32 の Web ページのうち，一部の Web ページで評価基準を満たさなかった．しかし，括弧で示したデータはパラメータに特殊文字が含まれる XSS 脆弱性判定処理を行う本提案システムの条件を満たしていないため，多くの Web ページにおいて有効であることが示された．したがって，テスト環境及び実環境における評価より，提案システムは評価基準を満たし，XSS 脆弱性の自動判別・収集システムとして有効であることが示された．

表 7　実環境における両提案プロキシの評価結果

ページの分類	合計	レスポンス型		リクエスト型	
		○	×	○	×
News Media	7	7	0	7	0
Government Agencies	3	3	0	3	0
Online Stores	9	9	0	8	1
Search Engines	3	3	0	3	0
Technology	6	4	(2)	4	(2)
Miscellaneous	4	4	0	3	1
合計	32	30	2(2)	28	4(2)

6. おわりに

本論文では，XSS 脆弱性が蔓延している原因はサーバ側の運用面の問題であることを指摘し，ユーザの自衛によるクライアント側の XSS 対策の必要性と，クライアント側で XSS 対策を行うために XSS 脆弱性を有する Web ページの情報が必要であることを述べ，XSS 脆弱性の自動判定・収集システムの提案及び実装を行い，評価により有効性を示した．

テスト環境及び実環境における評価では，提案システムを用いた XSS 脆弱性の自動判定能力，XSS 脆弱性情報の収集能力，そして，クライアント側における安全性の向上を検証し，提案システムの有効性を示した．これにより，ユーザが提案システムを使用した場合，クライアント側における XSS 対策を実現しつつ，正確な XSS 脆弱性の有無を判定する機能により，多くの XSS 脆弱性情報を収集可能とした．

今後の課題として，データベースへ送信される Web ページの脆弱性情報の真偽を確認する仕組みを用意する必要があると考えている．

文　献

[1] 高木浩光，関口智嗣，大蒔和仁，“クロスサイトスクリプティング攻撃に対する電子商取引サイトの脆弱さの実態とその対策”，情報処理学会コンピュータセキュリティ研究会 (CSS 2001).

[2] cgisecurity, “The Cross Site Scripting FAQ”, http://www.cgisecurity.com/articles/xss-faq.shtml

[3] CERT/CC, “CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests”, http://www.cert.org/advisories/CA-2000-02.html

[4] 三井物産 GTI プロジェクトセンタ，“安全な Web アプリケーション構築をサポートする新サービスの開始”，http://www.gtisec.net/news/news005_1007.html

[5] 産業技術総合研究所 グリッド研究センター　セキュアプログラミングチーム，http://securit.etl.go.jp

[6] Gunter Ollmann, “HTML Code Injection and Cross-site scripting”, http://www.technicalinfo.net/papers/CSS.html

[7] Point Blank Security, “The CSS Blacklist”, http://www.pointblanksecurity.com/css