

Session ID に着目した Session Fixation に対する脆弱性検査手法

内 海 正 貴^{†1} 小 菅 祐 史^{†1} 河 野 健 二^{†1,†2}

近年、ウェブアプリケーションにおけるセッション管理上の脆弱性を突いた攻撃である Session Fixation が問題となっている。Session Fixation は攻撃者が用意したセッション ID を被害者に使わせることで、被害者のセッション状態を乗っ取る攻撃である。Session Fixation 対策として、ログイン前のセッション ID は用いず、ログイン後に新たなセッション ID を発行する手法が知られている。しかし、多くのウェブアプリケーションでは適切に実装されておらず、脆弱性が存在してしまっている。本論文では、ログイン前のセッション ID とログイン後のセッション ID を取得し、両者の値を比較することで、Session Fixation に対する脆弱性を検査する手法を提案する。提案手法はウェブアプリケーションの挙動をみて検査を行うため、既に稼働しているウェブアプリケーションを容易に検査することができる。実験では、自作のウェブアプリケーション及び実在のウェブアプリケーションに対して提案機構を用いることで Session Fixation に対する脆弱性を検出することができた。

Detection of Session Fixation Vulnerabilities with Session ID Monitoring

MASATAKA UTSUMI,^{†1} YUJI KOSUGA^{†1}
and KENJI KONO ^{†1,†2}

In recent years, session fixation has become one of the most critical security flaws in web applications. Session fixation is an attack technique that forces a visitor of a web application to use a session identifier (SID) that the attacker prepared. After the visitor's login, the attacker can masquerade as the visitor by accessing the web application with the SID. It is well-known, effective technique for preventing session fixation to assign a new SID each time user logs in. However, many web applications in the real world do not properly accomplish the prevention technique. In this paper, we propose a technique to detect session fixation vulnerabilities in web applications by monitoring the change of SIDs before and after a user's login. Since our technique performs the checks web applications without requiring source code of the applications, it is

useful for check legacy web applications already running. In the experiment, our system successfully detected vulnerabilities in our original test cases and in a real world web application.

1. はじめに

近年、電子掲示板やショッピングサイトといったウェブアプリケーションが広く普及しており、それらの多くは、セッション管理を行うことで利用者ごとに専用のコンテンツを配信している。セッション管理とは、ウェブアプリケーションの利用者ごとの、ログインからログアウトまでの一連の処理を管理することである。セッション管理にはセッション ID と呼ばれる識別コードを用いる。ウェブアプリケーションは、利用者がウェブサイトにアクセスする度にセッション ID を送らせることによって、各利用者が行っている処理を認識することができる。

セッション管理を行う際には様々な脆弱性を考慮しなければならない。セッション管理において脆弱性が存在すると、攻撃者はウェブアプリケーションの正規の利用者のユーザ名やパスワードを知らなくても、その利用者がログインした後にしか行えない操作を実行できてしまう。例えば、パスワードの変更や商品の購入などといった操作を利用者の意図とは関係なく行わせたり、利用者のログイン後のページを閲覧してクレジットカード番号などの個人情報を盗んだりといったことが可能となってしまう。セッション管理上の脆弱性を突いた攻撃には Session Fixation, Cross-Site Request Forgery (CSRF), Session Hijack などがあるが、本研究では Session Fixation と呼ばれる攻撃を対象とする。

Session Fixation は、攻撃者が脆弱性のあるウェブアプリケーションから得たセッション ID を被害者に使わせることで被害者のセッション状態を乗っ取る攻撃である。Session Fixation の流れとしては、まず、攻撃者が脆弱性のあるウェブアプリケーションにアクセスしてセッション ID を入手し、入手したセッション ID をメールなどで被害者に送りつける。すると、被害者は攻撃者から送りつけられたセッション ID でログインする。その後、攻撃者は自分が指定したセッション ID を用いて脆弱性のあるウェブアプリケーションにア

^{†1} 慶應義塾大学
Keio University

^{†2} 科学技術振興機構 戦略的創造研究推進事業
JST CREST

クセスすることで、被害者のログイン後のページを閲覧することができてしまい、被害者の意図しない操作を行ったり、被害者の個人情報を盗み見たりといったことができてしまう。

Session Fixation の対策として、ウェブアプリケーションによるもの、攻撃者によるものを問わず、ログイン前のセッション ID は用いずに、ログイン後に新たなセッション ID を発行する手法が有効である。この手法を施すことによって、攻撃者にセッション ID を強要されても、被害者がログインする際には新しいセッション ID が発行されるので、攻撃者は被害者のページを閲覧することができない。しかし、WhiteHat Security¹⁾ の調査によると、ウェブアプリケーションの約 12% が Session Fixation に対する脆弱性を持っていると報告されている。また、MBSD²⁾ は対策が複雑なため実装を誤っているウェブアプリケーションや、間違った対策手法をとっているウェブアプリケーションが多く存在していると述べている。脆弱性のあるウェブアプリケーションは実際に攻撃を受けるまで脆弱性の存在自体に気づかないことが多いため、Session Fixation に対する脆弱性を早期に検出する必要がある。

そこで、本研究では Session Fixation に対する脆弱性を検出する手法を提案する。既に稼働しているウェブアプリケーションに対して提案機構を用いて検査を行うことで、脆弱性のあるウェブアプリケーションを検出し、開発者に対策を施すよう促すことができる。提案機構はログイン前のセッション ID とログイン後のセッション ID の値を比較することで検査を行う。具体的には、ログイン前のセッション ID とログイン後のセッション ID の値が異なっている場合、攻撃者はセッション ID を強要することができないので、Session Fixation 対策がとられていると判断する。セッション ID の値の比較を自動化することで、すべてを手動で行うよりも容易に脆弱性を検出することが可能となる。

提案手法を実現するために、ログインページの判別、及びリクエストからのセッション ID の抽出を行った。ログインページの判別ができれば、ログインページに遷移する前のリクエストはログイン前のリクエストであり、ログインページに遷移した後のリクエストはログイン後のリクエストであると判断することができる。一方、任意のリクエストにはセッション ID 以外の変数が含まれている可能性があるため、リクエストからのセッション ID の抽出を行うことで、誤ってセッション ID 以外の変数を用いて検査を行ってしまうということを防ぐことができる。以上の提案手法を実現するためにプロキシサーバとして検査システムを実装した。

提案機構の有用性を示すために、意図的に脆弱性を埋め込んだ自作のウェブアプリケーション及びオープンソースのウェブアプリケーションに対して実験を行った。その結果、いずれの場合も正しくセッション ID を抽出し、対策が施されていないウェブアプリケーション

ンに対して Session Fixation に対する脆弱性を検出することに成功した。

本論文の構成は以下の通りである。2 章では関連研究を示す。3 章では Session Fixation 攻撃について説明する。4 章では本研究の提案手法について説明し、5 章では提案手法の実装を説明する。6 章では提案機構の有用性を確認した実験について述べる。最後に 7 章で本論文をまとめる。

2. 関連研究

2.1 セッション管理上の脆弱性とその対策に関する研究

Session Fixation の対策手法は、acros³⁾ の調査でいくつか述べられているが、MBSD²⁾ によれば、対策がとられていないことが多いのが現状である。そこで、BASS⁴⁾ は脆弱性対策を行わなければならない操作をブラックボックス化し、自動で対策を行うサーバサイド・スクリプト言語を提案している。BASS を用いてウェブアプリケーションの開発を行うことで、開発者が脆弱性について詳しくなくとも、セッション管理上の脆弱性対策を行うことができる。しかし、既に稼働しているウェブアプリケーションで BASS を用いるためにはウェブアプリケーションをすべて BASS を用いて書き直さなければならない。本研究では、ウェブアプリケーションの安全化は行わないが、既に稼働しているウェブアプリケーションの安全性をチェックすることができる。

Session Fixation と同様にセッション管理上の脆弱性を突いた攻撃の 1 つとして Cross-Site Request Forgery (CSRF) と呼ばれる攻撃がある。CSRF は被害者のセッション状態を利用して脆弱性のあるウェブアプリケーション上で被害者の意図しない操作を行わせる攻撃である。この攻撃の対策として、ランダムな文字列であるトークンを用いる手法や Referer ヘッダの URL をみる手法がある。前者の手法はログイン後のすべてのウェブページにおいてトークンを管理する必要があり、ウェブアプリケーションの大幅な改変が必要になる。そこで、NoForge⁵⁾ はトークンの管理をウェブアプリケーションとは分けて行う手法を提案している。NoForge は、クライアントとウェブアプリケーションの間にプロキシサーバとして存在し、プロキシサーバ内でセッション ID とトークンを結びつけて保存する。この手法を用いることで、既存のウェブアプリケーションの改変をほとんど行わずに CSRF 対策を施すことができる。また、後者の手法では Referer ヘッダにクライアントの検索ワードなどが含まれる可能性があり、プライバシー上の問題が生じうる。そこで、プライバシー上の問題を解決したヘッダとして、Origin ヘッダ⁶⁾ がある。Origin ヘッダは、ドメイン名のみを用いるため、プライバシー上の問題を防ぐことができる。この Origin ヘッダは実際に

Google Chrome 4⁷⁾ などのブラウザで使用されている。これらの研究はリクエストとレスポンスを用いている点で本研究と類似しているが、いずれも CSRF への防御方法に着目している点が本研究と異なる。本研究では脆弱性の検査を目的としており、対象としている攻撃は Session Fixation である。

2.2 既に稼働しているウェブアプリケーションの脆弱性検査手法

Session Fixation に対する脆弱性を検査する研究は今のところ存在しない。そこで、本研究と同様に既に稼働しているウェブアプリケーションに対して脆弱性を検査する手法を以下に挙げる。

SecuBat⁸⁾ と WAVES⁹⁾ は検査対象のウェブアプリケーションに攻撃を送りつけ、そのときの挙動を解析することによって脆弱性の検出を行う。両者は SQL インジェクションとクロスサイト・スクリプティングに対する脆弱性の検出を目的としている。これらの攻撃は、ウェブアプリケーションに悪意のある文字列を送りつけ、その文字列が SQL クエリや、HTML や JavaScript で構成されたウェブページに出現することによって攻撃が成功する。そのため、送りつけた文字列が安全化されずに出現するか確認することによって脆弱性の検出を行う。

SecuBat は世界中のウェブアプリケーションにどの程度脆弱性があるかを調べた研究である。悪意のある危険な文字列を入力し、返ってくるエラーから危険度を計算して、それが閾値を超えた場合には脆弱性があると判断する。

一方、WAVES は既存の研究よりもより多くのウェブページを自動で収集し、脆弱性検査を行う研究である。多くのウェブページを収集するために、入力フォームにどのような値を入れれば良いかを自己学習する。この自己学習により、既存の研究では収集できなかった入力フォームに値を入力しなければ収集することができないウェブページも収集することができる。

3. Session Fixation

3.1 Session Fixation の仕組み

Session Fixation とはセッション管理上の脆弱性を突いた攻撃の 1 つで、攻撃者が指定したセッション ID をウェブアプリケーションの利用者（被害者）に使わせることで被害者のセッション状態を乗っ取る攻撃である。この攻撃が成功すると、攻撃者は被害者がログインした後のページを閲覧することが可能となり、被害者は個人情報の漏洩やログイン後にしか行えない操作を攻撃者によって勝手に行われてしまうといった被害を被ることになる。

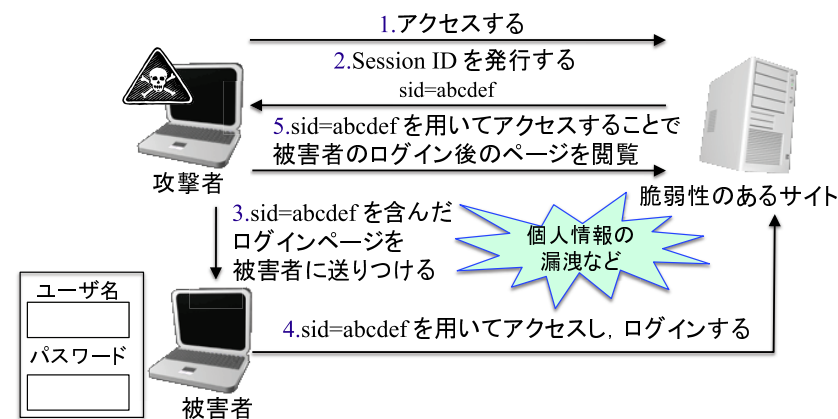


図 1 Session Fixation の攻撃の流れ

Session Fixation の攻撃の流れを図 1 に示す。まず、攻撃者は脆弱性のあるウェブサイトで使われているセッション ID を入手する（手順 1, 手順 2）。以後、攻撃者はこのセッション ID を用いて Session Fixation を行う。次に、攻撃者は入手したセッション ID を被害者に送りつけることで攻撃を仕掛ける（手順 3）。Session Fixation は攻撃者がセッション ID を得るだけでなく、そのセッション ID を被害者に使わせることで初めて成立する。そのため、攻撃者はメールやハイパーリンクなどを用いてセッション ID を被害者に送りつけ、被害者に自分が得たセッション ID を使わせようとする。そして、被害者は攻撃者から送られてきたセッション ID を用いて脆弱性のあるウェブサイトにログインする（手順 4）。これにより、脆弱性のあるウェブサイトは攻撃者が得たセッション ID を用いて被害者のアカウントを管理ようになる。最後に、攻撃者は自分が得たセッション ID を用いて脆弱性のあるウェブサイトにアクセスする（手順 5）。脆弱性のあるウェブサイトでは攻撃者が得たセッション ID で被害者を管理しているので、攻撃者は被害者のログイン名やパスワードを知らなくても、自分が得たセッション ID を用いれば被害者の専用ページにアクセスできるようになる。よって、攻撃者は被害者のアカウントを乗っ取ることができ、個人情報の閲覧やログイン後にしか行えない操作を勝手に行うことが可能となる。

3.2 Session Fixation 対策

Session Fixation の対策として、ウェブアプリケーションによるもの、攻撃者によるものを問わず、ログイン前のセッション ID は用いずに、ログイン後にはセッション ID を付け替

える手法が有効である．3.1 節で述べた Session Fixation 攻撃はログイン前に発行したセッション ID をそのまま用いてウェブアプリケーションの利用者のアカウントを管理しているため成功していた．しかし、この手法を用いることで、攻撃者によって送りつけられたログインページに含まれているセッション ID と被害者がそのページからログインした際に発行されるセッション ID が異なるため、攻撃者は自分が得たセッション ID を使ってウェブアプリケーションにアクセスしてもウェブアプリケーションは被害者のページを返さないの

3.3 現 状

ウェブアプリケーションの開発におけるセキュリティ機能はコンテンツの配信などの本来の目的とは異なるため、疎かになりがちである．WhiteHat Security¹⁾ によると、ウェブアプリケーションの約 12% が Session Fixation に対する脆弱性を持っているという報告がある．また、MBSD²⁾ によると、対策が複数のページにまたがるため複雑であり実装を誤るケースや間違った対策手法を用いているケースが多く存在すると述べている．このように、対策が不十分なまま稼働しているウェブアプリケーションは実際に Session Fixation 攻撃が発生し、利用者が被害を被るまで脆弱性があること自体に気づかないことが多い．そこで、既に稼働しているウェブアプリケーションにおいて Session Fixation に対する脆弱性を早期に検出する必要があるといえる．

4. 提 案 機 構

4.1 概 要

本研究では Session Fixation 対策がとられているかを検査する手法を提案する．提案機構を用いて、既に稼働している様々なウェブアプリケーションに対して検査を行うことで、対策が不十分なまま稼働しているウェブアプリケーションに脆弱性があることをそのウェブアプリケーションの開発者に通知することを目的としている．

4.2 アプローチ

本研究はログイン前のセッション ID とログイン後のセッション ID を比較することで Session Fixation に対する脆弱性の検査を行う．Session Fixation の対策の特徴から、ログイン前とログイン後でセッション ID の値が異なっている場合、対策がとられていると考えることができる．これは逆に、ログイン前からセッション ID を発行していて、かつログイン後にそのセッション ID の値が変わっていないウェブアプリケーションには Session Fixation に対する脆弱性があることを示している．そこで、提案手法ではログイン前のセッ

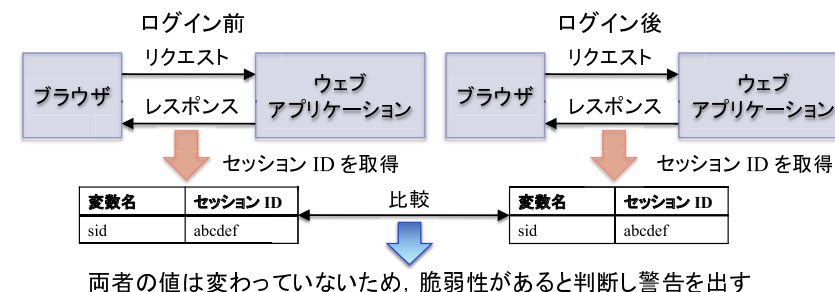


図2 提案機構による検査イメージ

ション ID とログイン後のセッション ID の比較を行い、両者のセッション ID の値が変わっていない場合には脆弱性があると判断する．提案機構による検査のイメージを図2に示す．

提案機構は、検査を行うために、任意のリクエスト・レスポンスから、どれがログイン前またはログイン後のものであるかを判断する．また、リクエストに含まれる複数の変数から比較を行うためのセッション ID の抽出も行う．以上の2点について、4.3 節及び 4.4 節で説明する．

4.3 ログインページの判別

ログイン前とログイン後のセッション ID を得る必要があるため、提案機構は自動でどのページがログインページであるかを判別する．ログインページを判別することができれば、そのページに遷移するまでのリクエストはログイン前のリクエストであり、そのページに遷移したあとのリクエストはログイン後のリクエストであることが分かる．

ログインページの特徴として、まず入力フォームを含んでいることが挙げられる．これは、ログインを行うためにはユーザが値を入力する必要があるためである．ただし、検索フォームなども入力フォームで構成されているため、他の指標を用いてログインページの判別を行う必要がある．

そこで、提案機構はウェブページが入力フォームを含んでいて、かつその入力フォームにテキスト領域とパスワード領域の2つがある場合、そのページをログインページであると判断する．一般的に、ログインを行うためにはユーザ名とパスワードを入力する必要があるためである．具体的には、図3に示すように、赤字で示した文字列が順番に含まれていたならログインページであると判断する．現段階では正規表現による文字列比較によって判断を行っているが、今後は HTML のパースを行う予定である．

```
<form method="GET" action"./login_check.php?sid=$sid">
<input type="text" name="user"> //テキスト領域
<input type="password" name="pwd"> //パスワード領域
<input type="submit" name="submit">
</form>
```

図 3 ログインページの例

4.4 セッション ID の抽出

リクエストにセッション ID を含める方法は URL リライティングによる方法と、クッキーを用いる手法がある。本節ではそれぞれの場合に、どのようにセッション ID を取り出すかを示す。また、本節まではリクエストに含まれる変数がセッション ID のみの場合を想定していたが、セッション ID を得る際にセッション ID 以外の変数が含まれている可能性もある。そこで、本節の最後に様々な変数の中からセッション ID のみを抽出するアルゴリズムを示す。

4.4.1 URL リライティングを用いている場合

URL リライティングを用いている場合、URL は

after_login.php?sid=abcdef

のようなものとなる。そこで、まず URL に“?”が含まれている場合、URL リライティングを用いてセッション ID の搬送を行っているとは判断し、“?”以降の記述を保存する。その後、比較を行うためさらに“=”で値を区切り変数名と実際の変数の値を提案機構に格納する。

4.4.2 クッキーを用いている場合

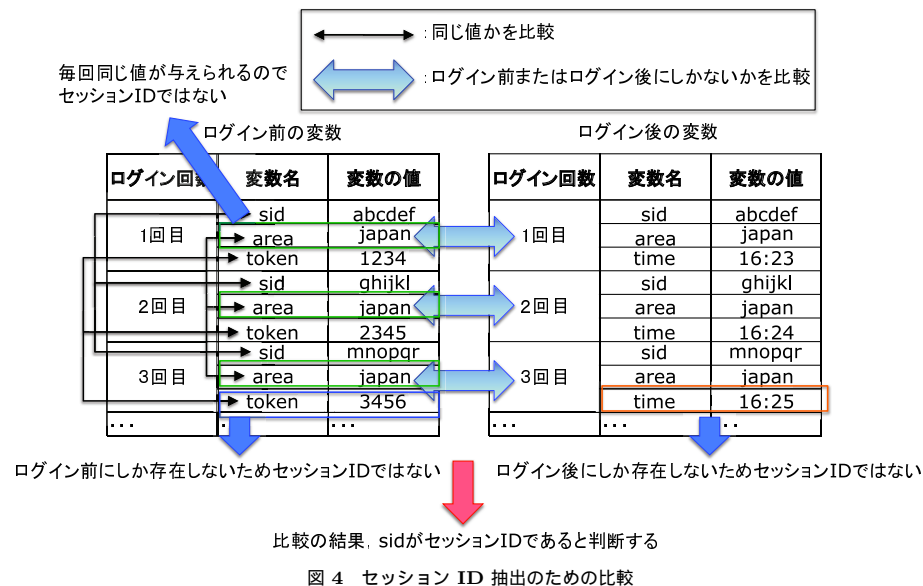
クッキーを用いている場合、リクエストのヘッダ部分の Cookie フィールドには

Cookie: sid=abcdef

のようにセッション ID が格納される。そこで、ヘッダ部分に Cookie フィールドがある場合には、セッション ID の搬送にクッキーを用いているとは判断し、Cookie フィールドに格納されている変数を保存する。そして、URL リライティングを用いている場合と同様に“=”で値を区切り、変数名と変数の値を提案機構に格納する。

4.4.3 セッション ID の判別手法

セッション ID を取得するために、提案機構はリクエストに含まれる変数の内、どれがセッション ID かを判別する。リクエストにはセッション ID 以外の変数が含まれている可能性があるが、提案機構ではセッション ID とはアクセスしてきたクライアントを識別するため



の変数であることに着目し、以下に示す変数はセッション ID ではないと判断する。

- 検査対象のウェブページに訪れると毎回同じ値が割り振られる変数
もしセッション ID が毎回同じものであれば、サーバ側ではセッション ID によってクライアントを判別することができなくなってしまう。そのため、セッション ID はアクセスする度に異なる値となるはずである。よってウェブページに訪れると毎回同じ値が割り振られるような変数はセッション ID ではないと判断する。
- 検査対象のウェブページのログイン前またはログイン後にしか存在しない変数
ログイン前にしか存在しない変数としてはログイン時に送るユーザ名やパスワードなど、ログイン後にしか存在しない変数としてはログイン時間などといったログイン後のクライアントの情報などが考えられる。セッション ID はクライアントを識別する値なので、ログイン前にしか存在しないケースはあり得ない。ログイン後にしか存在しないケースは Session Fixation 対策としてログイン前の変数を無視して、ログイン後のみセッション ID を発行している場合がある。しかし、この場合は Session Fixation 対策が既に施されているということなので、セッション ID ではないと判断しても問題な

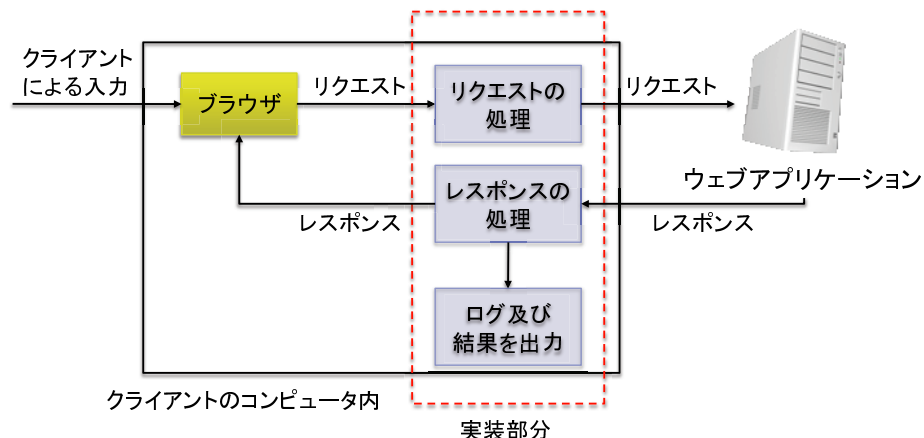


図5 提案機構の全体図

いと考えられる。そこで、ウェブページのログイン前やログイン後にしかないような変数も提案手法ではセッション ID ではないと判断する。

検査に用いる値はセッション ID のみにしたいため、提案機構の利用者が数回ログイン/ログアウトを行うことで変数の絞り込みを行う。ウェブアプリケーションで用いられている変数がこれらの条件に当てはまるかどうかは実際に何度かログイン/ログアウトを行い、リクエストに含まれる変数の値を比較しなければ分からないため、実際にセッション ID の比較を行って検査をする前にログイン前後の変数をすべて取得し、図4に示すようにそれらの変数の比較を行うことで、セッション ID として用いられている変数の絞り込みを行う。その後、絞り込んだセッション ID に着目して再度ログイン/ログアウトを行うことで、セッション ID の値が変わっているかを検査する。

5. 実装

提案手法に基づき、クライアントのコンピュータ内で動作するプロキシサーバとして実装を行った。提案手法のユーザに検査対象となるウェブアプリケーションのウェブページを手動で辿ってもらい、ログインに関してはユーザにユーザ名とパスワードを入力してもらうことで、セッション ID の付け替えが行われているかを解析するツールとして実装した。実装言語は Java である。

表1 使用したセッション ID の搬送法及びリクエストに含めた変数

ケース	対策	セッション ID の搬送法	セッション ID 以外に使用した変数
1	なし	URL リライティング	なし
2	なし	URL リライティング	毎回同じ値を与える変数 ログイン前のみ存在する変数
3	なし	クッキー	なし
4	なし	クッキー	毎回同じ値を与える変数
5	あり	URL リライティング	なし
6	あり	クッキー	なし

提案機構は図5に示すような構造になっている。リクエストから4.4節に示した手法を用いてセッション ID の取得を行い、レスポンスから4.3節で示した手法を用いてログイン前なのかログイン後なのかの判別を行う。そして、ログイン後のセッション ID を取得した時点でログイン前とログイン後のセッション ID を用いて解析を行う。

6. 実験

6.1 概要及び実験方法

提案手法の動作確認を行うために意図的に脆弱性を埋め込んだ自作のウェブアプリケーションに対して実験を行った。また、提案手法の有用性を示すためにオープンソースのウェブアプリケーションである phpBB に対して Session Fixation に対する脆弱性検査を行った。使用したブラウザは firefox 3.5.7 である。

実験の方法としては、まず3回ログイン/ログアウトを繰り返すことで、ウェブアプリケーションで使われている変数の中からどれがセッション ID であるかを判断し、その後もう1回ログイン/ログアウトを行うことでセッション ID の付け替えが行われているかの検査を行う。

6.2 自作ウェブアプリケーションに対する実験

6.2.1 ウェブアプリケーションの構成

自作のウェブアプリケーションに対する実験では、提案手法の動作確認を行うため、表1に示すセッション ID の搬送法及びセッション ID 以外の変数を用いて実験を行った。また、対策がとられていない場合だけでなく対策がとられている場合には脆弱性がないと判断するかも合わせて実験した。表1に示した各ケースにおけるセッション管理の方法は以下の通りである。

- ケース1～ケース4

表 2 自作ウェブアプリケーションでの実験結果

ケース	対策	提案機構による結果
1	なし	脆弱性あり
2	なし	脆弱性あり
3	なし	脆弱性あり
4	なし	脆弱性あり
5	あり	脆弱性なし
6	あり	脆弱性なし

ログインページに遷移する段階でクライアントにセッション ID を発行する．発行した後はセッション ID の付け替えは一切行わない．

・ ケース 5, ケース 6

ケース 1～ケース 4 と同様に，ログインページに遷移する段階でセッション ID を発行するが，ログインに成功するとセッション ID の付け替えを行う．

実験環境として，ウェブサーバは Apache 2.2.13 (Unix)，データベースサーバは MySQL 5.1.39 を用いた．

6.2.2 実験結果

表 1 に示した各ケースにおける実験結果を表 2 に示す．これにより，すべてのケースで正しく脆弱性の判断を行っていることが分かる．ここで，提案機構による実験結果が本当に正しいものであるかを検証するために，手動での攻撃を行った．攻撃の方法は，まず攻撃者を模したブラウザによってログインページのセッション ID を取得し，被害者を模したブラウザから，取得したセッション ID を用いてログインを行う．その後，攻撃者を模したブラウザから，最初に取得したセッション ID を用いてログイン後ページに移動した際に被害者専用のページを見ることができるかを試した．手動での攻撃の結果，提案機構による結果と同様にケース 1～ケース 4 では攻撃に成功し，ケース 5 とケース 6 では攻撃に失敗した．

6.3 オープンソースウェブアプリケーションに対する実験

6.3.1 ウェブアプリケーションの構成

提案機構の有用性を示すためにオープンソースのウェブアプリケーションである phpBB に対して実験を行った．実験に用いた phpBB は現在 SourceForge¹⁰⁾ において 410,449 件ダウンロードされており，世界で広く使われている．phpBB は NoForge⁵⁾ で CSRF に対する脆弱性が検出されており，Session Fixation に対する脆弱性も存在する可能性があるため実験の対象とした．

実験環境として，ウェブサーバは Apache 2.0.63 (Unix)，データベースサーバは MySQL 5.1.37 を用いた．phpBB のソースコードは SourceForge¹⁰⁾ から取得し，使用した phpBB のバージョンは 2.0.12 である．

6.3.2 実験結果

phpBB に対して提案機構を用いたところ，Session Fixation に対する脆弱性を検出することができた．検査のログ及び結果を図 6 に示す．図 6 の CookieAnalyze beforeSet 部分からログイン前のクッキーには phpbb2mysql_data, phpbb2mysql_sid, PHPSESSID の 3 つが変数として含まれ，CookieAnalyze afterSet 部分からログイン後のクッキーにもログイン前と同じ変数が含まれている．また，phpBB ではログアウトを行うと，ログインページに戻るため，提案機構はログイン前の状態になったと判断し，URLRewritingAnalyze beforeSet 部分に logout, sid という変数が表示されている．また，URLRewritingAnalyze beforeSet 部分が 2 回連続で出力されているのは，ウェブアプリケーションからクッキーを新たに得るためにブラウザで一度クッキーを破棄した後にログイン前のトップページを再度読み込んでいたためである．検査結果より，クッキーに含まれている 3 つの変数のうち phpbb2mysql_sid がセッション ID であると判断し，phpbb2mysql_sid はログイン前とログイン後で値が変わっていないため，提案機構は phpBB には Session Fixation に対する脆弱性が存在すると判断している．

確認のため，提案機構によってセッション ID であると判断された phpbb2mysql_sid を用いて実際に手動で Session Fixation 攻撃を引き起こせるかを検証した．検証法は自作のウェブアプリケーションのときに行った方法と同様に，攻撃者を模したブラウザから phpbb2mysql_sid の値を取得し，取得した phpbb2mysql_sid の値を用いて被害者を模したブラウザからログインを行い，さらにその値を用いて攻撃者を模したブラウザからログイン後のページに遷移すると被害者専用のページを見ることができるかというものである．手動で検証した結果，phpbb2mysql_sid を用いて攻撃者を模したブラウザから被害者専用のページを見ることができた．以上のことから，提案機構の有用性を示すことができたと考えられる．ただし，現在公開されている phpBB のバージョンは 3.0.6 であり，そちらですでに Session Fixation 対策がとられている．

7. ま と め

本論文では，既に稼働しているウェブアプリケーションに対して Session Fixation 対策がとられているかを検査する手法を提案した．提案機構を用いて検査を行うことで，対策が

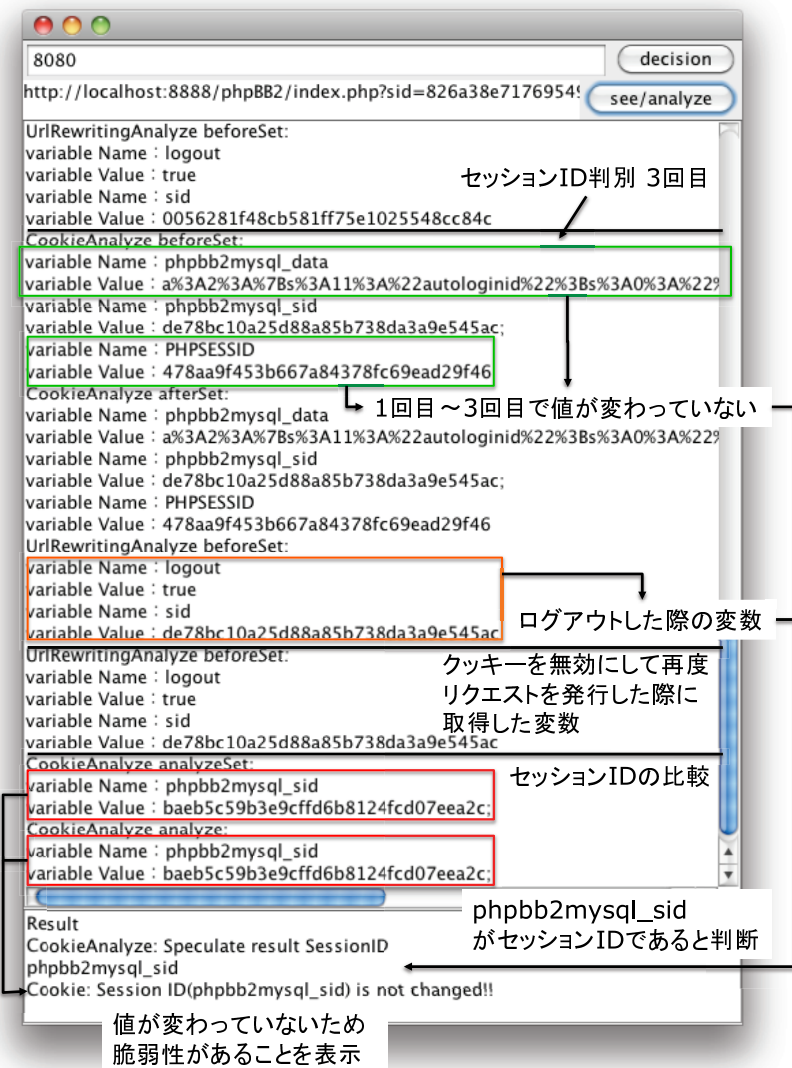


図 6 phpBB における検査結果

不十分なまま稼働しているウェブアプリケーションに脆弱性があることを開発者に通知することを目的とし、すべてを手動で行うよりも容易に脆弱性を検出することが可能となる。提案手法はログイン前のリクエストとログイン後のリクエストからセッション ID を取得し、両者を比較することで対策が施されているかを検査する。実験では、実際に運用されているウェブアプリケーションに Session Fixation に対する脆弱性が存在することを示した。

今後の課題として、さらに多くのウェブアプリケーションに提案手法を検査できるようにログインページの判別やセッション ID の抽出精度の向上を行うことが挙げられる。また、現段階ではページの遷移やユーザ名、パスワードの入力は手動で行っているため、検査をより容易に行えるようにするために検査の自動化を行いたいと考えている。

参考文献

- 1) WhiteHat Security, Inc.: Website Security Statistics Report 8th Edition Fall 2009, <http://www.whitehatsec.com/home/resource/stats.html>.
- 2) Mitsui Bussan Secure Directions (MBSD), Inc. : 2008 年度 Web アプリケーション脆弱性検査レポート, http://www.mbsd.jp/news/pressrelease_img/20090717/websec2008report.pdf.
- 3) acros: Session Fixation Vulnerability in Web-based Applications, http://www.acrossecurity.com/papers/session_fixation.pdf.
- 4) Yu, D., Chander, A., Inamura, H. and Serikov, I.: Better abstractions for secure server-side scripting, *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pp.507–516 (2008).
- 5) Jovanovic, N., Kirda, E. and Kruegel, C.: Preventing Cross Site Request Forgery Attacks, *Securecomm '06: Second International Conference on Security and Privacy in Communication Networks*, pp.1–10 (2006).
- 6) Barth, A., Jackson, C. and Mitchell, J.C.: Robust Defenses for Cross-Site Request Forgery, *CCS '08: Proceedings of the 15th ACM conference on Computer and Communications Security*, pp.75–88 (2008).
- 7) Google, Inc.: Google Chrome, <http://www.google.com/chrome>.
- 8) Kals, S., Kirda, E., Kruegel, C. and Jovanovic, N.: SecuBat: a web vulnerability scanner, *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pp.247–256 (2006).
- 9) Huang, Y.-W., Huang, S.-K., Lin, T.-P. and Tsai, C.-H.: Web application security assessment by fault injection and behavior monitoring, *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pp.148–159 (2003).
- 10) SourceForge, Inc.: phpBB, <http://sourceforge.net/projects/phpbb/>.