ウェブアプリケーションにおけるセッション固定化脆弱性の検出支援

廣田 一貴† 中安 恒樹† 露木 航平† 山本 知典‡ 武田 圭史†
村井 純†

†慶應義塾大学 環境情報学部
252-0882 神奈川県藤沢市遠藤 5322

‡慶應義塾大学大学院 政策・メディア研究科
252-0882 神奈川県藤沢市遠藤 5322

あらまし ユーザのログイン状態を保持する必要のあるウェブアプリケーションではブラウザに対してセッション ID を割り振って管理することが一般的である．これによって HTTP のようなステートレスなプロトコルでもユーザのアクセス状態を保つ事が可能となる．しかしこのセッション ID を第三者に知られた場合，パスワードなどの認証情報を知られずともセッションを乗っ取りそのユーザとして操作を行われてしまう問題がある．そこで本研究ではセッション管理手法における脆弱性の一つであるセッション固定化問題に着目し，任意のアプリケーションについてこの脆弱性の有無を容易に確認するための手法を提案し，その手法を適用した検査ツールを実装した．

Assist for Detection of Session Fixation Vulnerability in Web Application

Kazuki Hirota† Koki Nakayasu † Kohei Tsuyuki† Tomonori Yamamoto‡
Keiji Takeda† Jun Murai†

†Faculty of Environment and Information Studies, Keio University.
5322 Endo, Fujisawa-shi, Kanagawa 252-0882, JAPAN
‡Graduate School of Media and Governance, Keio University.
5322 Endo, Fujisawa-shi, Kanagawa 252-0882, JAPAN

Abstract Web applications usually manage a user's login status by allocating a session ID to their browser. Although HTTP is a stateless protocol, this method makes it possible to manage access states of a user. However, this method has a problem that malicious users can hijack accounts without knowing authentication information such as password when he gets the Session ID. In this paper, we focused on the vulnerability known as Session Fixation, proposed a technique to detect this vulnerability, and implemented a tool based on the technique.

1 はじめに

今日のウェブアプリケーションでは，ログイン状態やページ遷移などを保持する際にセッション ID を各ユーザの利用するブラウザに割り振って管理する手法が一般的に用いられている．これによって HTTP のようなステートレスなプロトコルでもユーザのアクセス状態を保つ事が可能となる．ステートレスとは通信を処理単位で切断し前の状態を保持しない仕様である．

一方でこの管理方法においてはユーザのセッション ID は攻撃者に対して秘匿する必要があり，攻撃者がセッション ID を入手したり推測出来る事が大きな問題となる．サーバは各ユーザを通信ごとに送られてくるセッション ID によって識別認証しているため，ID やパスワード

といったログインに必要な情報を知らずとも，セッション ID を獲得することでそのユーザとして操作を行う事が出来るからである．

セッション ID を不正に入手する手法として，セッション固定化攻撃 [1]（Session Fixation）がある．これは後述するセッション固定化の脆弱性を持つアプリケーションに対して行われる攻撃であり，WhiteHat Security の調査 [2] では，同社が調査したウェブアプリケーションのうち 14 %にセッション固定化の脆弱性があると報告されている．

本研究では，このセッション固定化の脆弱性に着目し，クライアントサイドからのテストのみで本脆弱性の有無の検査及び検査支援を行う手法を考案し，ツールとして実装を行った．

2　セッション管理

2.1　セッション ID

今日のウェブアプリケーションでは一般にセッション ID を各ユーザの利用するブラウザに割り振って管理することが多い．サーバはこのセッション ID を通信のたびに受け取り，サーバ側ではこの ID を鍵（key）として値（value）を取り出すことでユーザの状態情報を擬似的に保持する．この機能は主要なプログラミング言語やライブラリには標準で実装されており，各アプリケーションの開発者はそれらを使うことで容易にセッション管理を行える．

2.2　クッキー

セッション ID は主に RFC6265[3] で定義されるクッキーを使って受け渡しされることが多い．サーバはセッション管理を行う時に Set-Cookie ヘッダを利用してブラウザにセッション ID を記録させ，アクセス毎に Cookie ヘッダでこのセッション ID を受け取る．クッキーは期限やドメインの指定が出来，そのセッション ID を使える期限やサーバを決められる点でセッション管理に向いている．

しかしこの仕様によりクッキーは自分の管理するサイト以外からも発行される可能性がある．クッキーで指定出来るドメインは，一般にクッキー発行時のページのドメイン及びそのドメインのサブドメインである．一般的に，.jp や.com のようなトップレベルドメインや，.tokyo.jp のような地域型 JP ドメイン等に対してクッキーを発行することは出来ず，それが可能な状態はクッキーモンスターバグとよばれる．

3　セッション固定化攻撃

3.1　攻撃対象

セッション固定化攻撃は，ログイン前後でセッション ID が変化しないウェブアプリケーションと，それを利用しているユーザに対して行われる．またログインを伴わない処理，例をあげると EC サイトにおけるログイン前の買い物カゴ機能等でも起こりうる．しかし，ログイン前は機密情報を保持していることが少ない．よって実際に問題となることは少ない．そのため，本論文ではログイン処理を伴うウェブアプリケーションにおけるセッション固定化攻撃を主な対象とする．

またセッション ID をユーザの機密情報から生成し，画面ごとに復元してチェックを行っている場合，セッション固定化攻撃は原理的に不可能である．例えばユーザ名とパスワードとログイン時刻を結合し，それを可逆な暗号方式で暗号化してセッション ID として使い，セッションから情報を取り出す際に復号して確認を行っている場合だ．一部のウェブアプリケーションはこういった独自手法でセッション管理を行っているが，多くのウェブアプリケーションでは言語やフレームワークに備わっている汎用的なセッション管理機構を利用しており，そういった機構で生成されるセッション ID は乱数を元にして生成される．よって，本論文ではセッション ID がユーザの機密情報とは関係なく生成されるウェブアプリケーションを対象とする．

3.2　攻撃手法

前節で述べたように，本攻撃はログイン前後でセッション ID が変わらない事を利用する．攻

撃者にとって既知のセッション ID で被害者にログインさせ，そのセッション ID を使って被害者のアカウントに成りすまし操作を行う．一般的なウェブアプリケーションでは一度ログインすると，その後の処理ではセッション ID からどのユーザとしてログインしてるかを判断するため，ユーザの認証情報を知らずともアカウントハイジャックが可能となる．

典型的なセッション固定化攻撃は以下の手順で行われる．

1. 攻撃者は攻撃対象サイトから正規の手順でセッション ID を得る．
2. 攻撃者は後述する手法等で被害者に対しセッション ID を強要する．
3. 被害者は強要されたセッション ID を使い，攻撃対象サイトにログインする．
4. 攻撃者は被害者がログインした頃を見計らって固定化したセッション ID を使って被害者ユーザに成りすまし操作を行う．

ここでいう強要とは，セッション ID をクッキーに書き込み，それを利用させる事である．具体的な手法としては，クッキーの仕様を利用し，該当サイトまたはそのサブドメイン上のウェブアプリケーションにあるクロスサイトスクリプティング等の脆弱性を利用する方法や，それらのサイトからのレスポンスを改竄する方法，2.2 節で示したクッキーモンスターバグを利用する方法などが考えられる．

3.3 対策手法

前小節で述べたセッション ID の強要は，ウェブアプリケーション管理者だけでは防ぐのが難しい．そのため，森川氏らの論文 [4] でも示されている通りログイン後にセッション ID を再生成することでログイン前後でセッション ID を一致させない事が最も有力な対策となる．主要なプログラミング言語やフレームワークでは関数やメソッドとして実装されており，認証後にそれらを呼び出す事でセッション固定化攻撃を防げる．仕様上の理由などで再生成出来ない場合は，徳丸氏 [5] が示しているように，認証後にセッション ID とは別にランダムなクッキーを発行し，それとセッション ID を結びつけ，逐次それらが正しく送信されているかを確認することでも対策となる．他にも，Martin 氏 [6] らが示しているように，リバースプロキシを用いてセッション管理を透過的に行う手法もある．また，重要な処理の前には再度認証を行う事やセッション ID と IP アドレスを関連付ける事等も多重の防御メカニズムとなる．

4 提案手法

セッション固定化攻撃を行うにおいて，最も重要なのは前述の通りログイン前後でのセッション ID の変化である．これを確認することで大半の脆弱性は検知することが可能である．一方，徳丸氏の対策のように単純に変化の有無だけでは発見出来ない場合もある．よって，まずはトークンとセッション ID の生成に関するパターンについて考察し，その上で検知手法を示す．

4.1 分類

本節ではトークンとセッション ID の生成タイミングによって分類し，セッション固定化の脆弱性となりうるかを示す．また，例外的にセッション固定化が不可能な認証情報からセッション ID を生成しているパターンに関しても示す．

4.1.1 ログイン前に割り振る

ログインフォーム画面でセッション ID を割り振り，クッキー中にセッション ID があればそれを利用し，ログイン処理後もそのままそのセッション ID を利用するパターンがある．これは典型的なセッション固定化の脆弱性である．

4.1.2 ログイン後に割り振る

ログイン処理を終え正規ユーザとして認証された後に，セッション ID を割り振り，クッキー中にセッション ID があればそれを利用するパ

ターンがある．この場合，攻撃者が正規のユーザとして攻撃者自身のセッション ID を取得出来る環境であれば攻撃できる．

また，セッションアダプションを利用することで，ユーザでなくとも攻撃できる．セッションアダプションとは，サーバ側が割り振っていないセッション ID を利用出来てしまうバグである．攻撃手順は一般のセッション固定化攻撃と同様で，強要するセッション ID を攻撃者が自由に作れるため攻撃が容易になる．

4.1.3 ログイン前にセッション強制再生成

ログインフォーム画面で，既存のクッキーに関係なくセッション ID を割り振るパターンがある．この場合，ログインフォーム画面を開くまでにセッション ID を強要しても新たに生成されてしまうため，セッション ID を固定化出来ない．しかし，ログインフォーム画面を開いてからログイン処理を行う画面に認証情報をポストするまでの間にセッション ID を強要出来ればセッション ID を固定化する事が出来る．よってこれは不十分な対策であり，セッション ID の固定化が可能な脆弱性となる．

4.1.4 ログイン後にセッション強制再生成

ログイン処理を終え正規ユーザとして認証された後に，既存のクッキーに関係なくセッション ID を割り振るパターンがある．この場合，ログイン前後でセッション ID が必ず別の物となるため，セッション固定化は不可能である．

4.1.5 ログイン前にトークンを割り振る

クロスサイトリクエストフォージェリ（CSRF）の対策として，hidden フィールドとセッションにそれぞれ同じランダムな値を持たせ，認証情報と共にポストされた値とそれらを比較し，一致していれば処理を行うというパターンがある．この場合，ログインフォーム画面を開いてからログイン処理を行うまでの遷移が保護されるため，その間はセッション ID を強要出来ない．しかし，ログインフォーム画面を開く前にセッション ID を強要した場合，その ID にあわせてトークンが生成されるため，セッション ID の固定化が可能な脆弱性となる．トークンをクッキーに割り振った場合は遷移が保護されないため，実質的にトークンが無いものとしてセッション固定化攻撃が可能である．

4.1.6 ログイン後にトークンを割り振る

徳丸氏が示す対策のように，ログイン後にクッキーとしてトークンを割り振って認証後のどのページにおいても読み込み時に確認し，トークンとセッションが対になっていなかった場合はエラーとするパターンがある．この場合，ログイン後に割り振られるトークンが機密情報となり，攻撃者はこのトークンを奪取出来ない．よって他の固定化が可能な条件を満たしていたとしても脆弱性ではない．

4.1.7 認証情報からセッション ID を生成

認証情報を利用してセッション ID を生成し，セッション管理を行っているパターンがある．この場合，他のユーザの認証情報を攻撃者は知らないためセッション ID を生成出来ない．よって他の固定化が可能な条件を満たしていたとしても脆弱性ではない．

4.2 検知手法

高松氏 [7] が示すように，ログイン前後のセッション ID を比較し変化していれば 3.2 節で示した手順で攻撃を仕掛ける事で脆弱性の有無を確認出来る．しかし氏の提案手法では，被害者のログイン情報を組み立てる際にログインフォーム画面を攻撃者のセッション ID で取得しており，4.1.3 節で述べたパターンの場合に検知漏れを起こす．また，トークンの検査を行っていないため，4.1.5 節及び 4.1.6 節で述べたパターンに関しても検知漏れや誤検知を起こす．これを改良し，図 1 ような手順で検知を行う．

1 ログインフォーム画面を開き，ログイン情報を取得する．存在すればセッション ID も記録する．

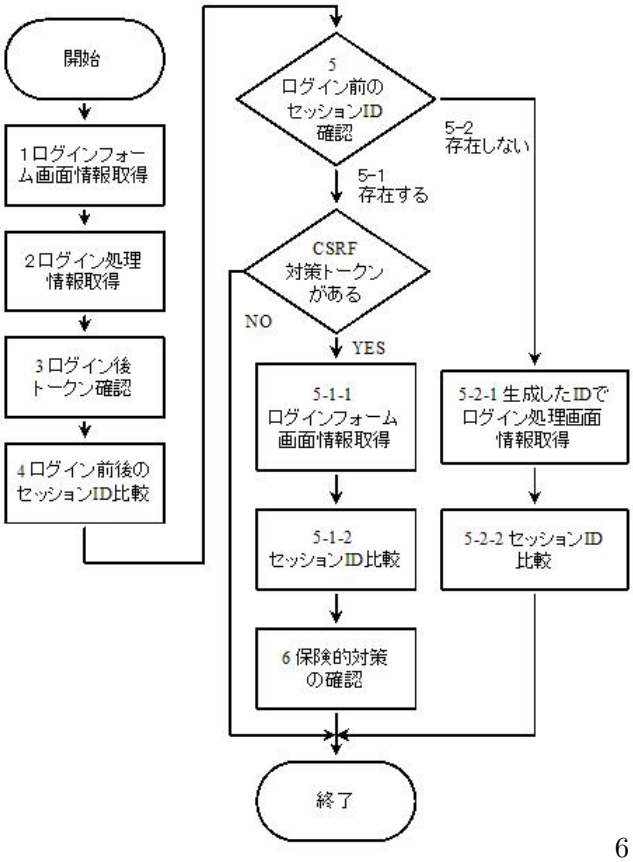

図 1: 検知手法

2 ログイン情報をログイン処理画面に送信し，ログイン後の画面とクッキー情報及びセッション ID を記録する．

3 ログイン後に割り振られるトークンの有無を確認する．あれば脆弱性は無いと判断し，終了する．

4 ログイン前後のセッション ID を比較する．変わっていれば脆弱性は無いと判断し，終了する．ログイン前のセッション ID が無かった場合は変わっていないと判断する．

5 ログイン前にセッション ID が割り振られているかチェックする．

5-1 ログイン前にセッション ID が割り振られていた場合

5-1-1 CSRF 対策トークンがログイン情報中にあった場合，1〜3 で使っていたセッション ID を使って再度ログインフォーム画面を取得し，セッション ID を記録する．

5-1-2 1〜3 で使っていたセッション ID と比較し，違っていた場合はログイン前の強制再生成と CSRF 対策トークンによるセッション ID の固定化対策とみなし，脆弱性は無いと判断する．

5-2 ログイン前にセッション ID が割り振られていなかった場合

5-2-1 任意のセッション ID を生成または本検証とは別の正規ユーザとしてログインした場合のセッション ID を取得し，2 で使ったログイン情報と共にログイン処理画面に送信し，セッション ID を記録する．

5-2-2 送信したセッション ID と記録したセッション ID を比較し，違っていれば脆弱性は無いと判断する．

6 ここまでの脆弱性情報を元に攻撃を仕掛け，成功した場合は脆弱性があると判断する．この攻撃は保険的対策の有無を確認するために，異なる IP アドレスから実際の攻撃と同様の手順で行う必要がある．

手順 3 までの一連の処理で 4.1.6 節のパターンに該当するかチェック出来る．ログイン後にトークンがあった場合，以降の処理の結果に関係なくセッション固定化攻撃は出来ないと言える．

手順 5.1 までの一連の処理で 4.1.4 節のパターンに該当するかチェック出来る．ログイン前にセッション ID を割り振られており，かつログイン後に別のセッション ID が割り振られた場合，セッション固定化の脆弱性は無いと言える．

手順 5-1-2 までの一連の処理で 4.1.5 節と 4.1.3 節のパターンに該当するかチェック出来る．4.1.5 節のパターンはログインフォーム画面からログイン処理までのセッションを保護する機能を持っている．また 4.1.3 節のパターンはログインフォーム画面までの遷移を保護する機能を持っている．これら二つを組み合わせて利用した場合，セッション ID を固定化出来るタイミングが無いため，セッション固定化攻撃は出来ないと言える．

手順 5-2-2 までの一連の処理で 4.1.4 節のパターンに該当するかチェック出来る．この段階

でのチェックは，セッションアダプションを利用した攻撃及び攻撃者が正規ユーザとしてセッション ID を取得出来た場合のセッション固定化攻撃が可能かをチェックしている．もし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 が違っていた場合，以降の処理の結果に関係なくセッション固定化攻撃は出来ないと言える．

5 実験

5.1 ツール実装

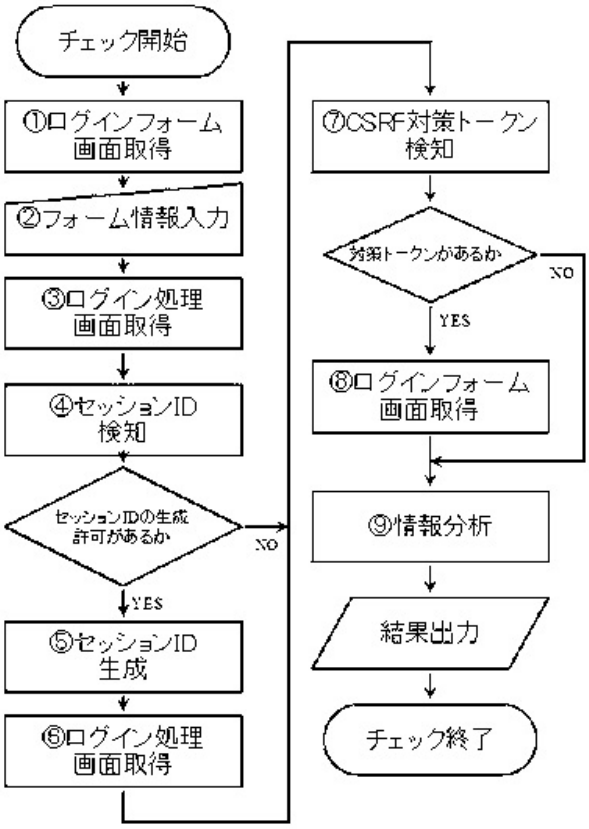

図 2: ツール検知処理

提案手法の有用性を確認するためのツールを実装した．このツールは脆弱性発見を支援するだけでなく，脆弱性情報を取り扱う仲裁機関やウェブアプリケーション運営者への報告文作成機能をつけた．これにより円滑に報告を行う事が出来るだけでなく，セッション固定化に関する専門知識の無いユーザでも検査し，知識のある機関や人物に確認することで脆弱性発見を支援することが出来る．

本ツールのユーザは，まずツールの利用法を読む旨のメッセージに同意し，ログインフォーム画面のアドレスを入力する．ツールは入力されたアドレスを用いて図 2 の手順でチェックを行う．

図 2 中の②フォーム情報入力は，①ログインフォーム画面取得において取得した情報を分析し，フォームを全て列挙する．ユーザはその中からログイン用のものを選択し，ユーザ名とパスワードを入力する．この情報を元に ③ログイン処理画面取得の処理を行う．

④セッション ID 検知は，クッキー中にある以下の条件を満たすものをセッションとして扱っている．ここでセッション ID として扱われたものは⑤セッション ID 生成において 4.2 節中の手順 5-2-2 の処理を行うためにツール側でランダムに値を生成され，⑥ログイン処理画面取得としてサーバに送信される．この④の検知機能は⑤,⑥の処理のためだけに使われ，⑨情報分析等においてはセッション ID とその他のクッキーを平等に扱っている．これはツールの目的が検査支援であり，機械的に行うと誤検知や検出漏れに繋がる恐れのある機能に関しては処理せずツール利用者にその情報を知らせる事が目的に適った挙動だからである．なお本機能は稼動するウェブアプリケーションへの検査実験中は法律に抵触する恐れがあったため使わず，現在は処理としては残っているもののデフォルトでは無効としている．

⑦CSRF 対策トークンの検知はログインフォームにおいて hidden となっている値があった場合に行っている．この処理は 4.2 節中の手順 5-1-2 の処理に該当する．そのため⑧ログインフォーム画面取得では①で得られたクッキーをサーバに送信している．

⑨では，一連の検査で得られた情報を元に分析を行う．まず①と③で得られたクッキーを比較し，ログイン前後で変化していないクッキーを探す．これは 4.2 節中の手順に該当する．変化していないクッキーがあった場合，③で得られたクッキーを格納しているテーブル中の該当

クッキーオブジェクトに対して固定化フラグを立てる．次に⑧で得られた情報と①で得られたクッキーを比較し，変化していないクッキーがあった場合，①で得られたクッキーを格納しているテーブル中の該当クッキーオブジェクトに対して固定化フラグを立てる．その後，これらの情報と各画面を構成する HTML 情報を出力する．

ユーザはこの情報を元に，IPA またはウェブアプリケーションの運営者もしくは専門家に報告を行うかどうかを選択する．ここでいう専門家は，送信されてきた情報を悪用せず，詳細な分析を行って脆弱性の有無を判断し，しかるべき人物または機関に届出て仲裁を行う人物を指す．

続いて送信する情報を選択する．デフォルトではクッキーやフォーム情報の内容は送信されないようになっているが，ツール利用者が自身の責任でそれらの情報を送信できるようになっている．送信情報を選択すると，送信される文面をチェックする画面に移る．ここでは最終的に送られる情報を確認及び編集することが出来る．自身の機密情報等が記載されていないかどうかを確認し，送信先が IPA またはウェブアプリケーションの運営者の場合は作成した文章をテキストファイルとしてローカルに出力し，専門家の場合は暗号化され専用のサーバに送信される．専門家は各ツールに使われている公開鍵に対する秘密鍵を保有しており，これを使って情報を復元する．

本ツールはチェック用のライブラリとツール固有のコードに分かれており，様々なツールに組み込める．またライブラリにはツールでは無効化した機能もついており，それらを使ったチェックも行えるようになっている．

5.2 検査用ウェブアプリケーションの検査

ツールが正しく検知出来るかを確認するため，4.1 節で示した各パターンの処理を行うウェブアプリケーションを実装し，ツールを用いて確認を行った．その結果表 1 のようになった．

4.1.2 節のパターンは前述の通り本ツールでは無効化されているものの，ライブラリとしてはチェック出来ている．

表 1: 検査用サイト対象の実験結果

検査事項	ツールの挙動	脆弱性
4.1.1 節のパターン	検知し警告を表示	有
4.1.2 節のパターン	検知せず	有
4.1.3 節のパターン	検知し警告を表示	有
4.1.4 節のパターン	検知せず	無
4.1.5 節のパターン	検知し警告を表示	有
4.1.6 節のパターン	検知し警告と注意を表示	無
4.1.7 節のパターン	検知し警告を表示	無
4.1.3 節と 4.1.3 節のパターン	検知し注意を表示	無

4.1.6 節のパターン及び 4.1.3 節と 4.1.3 節を合わせたパターンは，トークンの判定を機械的に行っている関係上，間違ったものをトークンと認識し，セッション固定化の脆弱性があるのに無いと検出漏れを起こす可能性があるため，例えトークンらしきものが検出された場合でも脆弱性があるという警告を出し，注意としてトークンらしきものが見つかった旨を表示するという挙動になっている．

4.1.7 節のパターンは機械的に検知することが難しいため，特にチェックを行っていない．そのためこのパターンの場合は誤検知が起こる．

この結果より，検査支援ツールとして有用だと確認出来る．

5.3 稼動するウェブアプリケーションへの検査

セッション固定化の脆弱性は，それを確認するために該当サイトのアカウントを保持している必要があり，単独での検査や開発者が自身のサイトを検査するだけでは限界がある．そこで筆者だけでなく信頼出来る人物 4 人に協力を依頼し，本ツール及びその過程で判明した情報を利用した不正アクセス行為等を行わないことを前提に実稼動するウェブアプリケーションへの検査を行った．このツールの挙動は任意のセッション ID を生成して送信する手順を除けば通常のブラウザでのログインと何ら変わらない．そのため，ツール配布時には該当機能を利用しないように指示することで，4.1.2 節で示したパターンが検査出来ないかわりに法令に抵触しないようにした．また，念のため協力者が検査を行う対象サイトはそういった実験を行っても問題の無いサイトを選択してもらった．本実験の

表 2: 稼働中サイト対象の実験結果

脆弱性の有無	パターン	件数
無し	-	7 件
有り	4.1.1 節のパターン	4 件
	4.1.3 節と 4.1.3 節のパターン	1 件
検知不能	-	1 件
総計		13 件

結果，協力者に 4 件サイトを検査してもらい，2 件のサイトに本脆弱性があることが分かった．また，筆者自身でも脆弱性報告を推奨しているサイトなどを 9 件ほど検査し，4 件の脆弱性を発見した．打ち明けは表 2 の通りである．

実験の過程で，本ツールでは検査が行えないウェブアプリケーションを発見した．このウェブアプリケーションはログイン画面を開くまでに複数の画面遷移を要するもので，ログイン画面に行く前にセッション ID を二つ割り振ってそれらの遷移を追跡しており，ログインに関する処理のみでセッションが割り振られる事を前提としていた本ツールでは検査が出来なかった．

6 まとめ

本研究では，ウェブアプリケーションにおけるセッション管理手法の脆弱性の一つであるセッション固定化問題に着目し，これを分類分けし，それらに対応する提案手法を考案した．またその手法を利用した検査ツールを作った．このツールを用い，検査用に作ったサイトに対して検査する実験と，協力者と共に実際にサイトを検査する実験を行った．

検査用のウェブアプリケーションを検査する実験の結果，法的な問題等によって一部は検知出来なかったが，検出支援としては十分使えることが分かった．また，実際のサイトを検査する実験の結果，一部サイトでは検査を行えなかったものの，実用に耐えうることが分かった．検査出来なかったサイトに関しては，本ツールにおいては対応させない．これは実験過程で見つかったような複雑な画面遷移に対応させた場合，ツールの設定等も複雑になり運用しにくいものとなってしまう上，そういった複雑な設定は誤検知や検知漏れの原因となるからである．

今後の課題としては，ツールを開発者向けと利用者向けに分け，法的な問題を解決することや，ブラウザに組み込み設定を容易にすることなどが考えられる．

参考文献

[1] Session Fixation Vulnerability in Web-based Applications
Mitja Kolsek
Acros Security,December 2002

[2] WhiteHat Website Security Statistic Report Winter 2011,11th Edition
http://www.whitehatsec.com/resource/stats.html

[3] RFC 6265 HTTP State Management Mechanism
http://tools.ietf.org/html/rfc6265

[4] モデル検査による Web セッションのセキュリティ欠陥検出手法
森川 郁也，山岡 裕司，中山 裕子
電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス 107(4), 29-34, 2007-04-12

[5] 体系的に学ぶ 安全な Web アプリケーションの作り方
徳丸 浩
2011 年 3 月 1 日, ソフトバンククリエイティブ

[6] Reliable protection against session fixation attacks
Martin Johns，Michael Schrank，Joachim Posegga
SAC '11 Proceedings of the 2011 ACM Symposium on Applied Computing Pages 1531-1537

[7] 高松 勇輔, 河野 健二
ウェブアプリケーションの開発時におけるセッション管理の脆弱性の自動検査手法
2012 年 3 月, 慶應義塾大学大学院理工学研究科 修士論文