Computer Security Symposium 2016
11 - 13 October 2016

HSTSによる対策を回避可能なsslstrip攻撃

瀬戸崎 喬[1]　松尾 和人[1]

概要：HTTPS 通信に対する中間者攻撃として Marlinspike によって提案された sslstrip 攻撃が知られている．Sslstrip 攻撃はユーザと攻撃者間の通信を HTTPS から HTTP に置き換えることで機密情報を盗聴する手法である．この攻撃は HTTP Strict Transport Security (HSTS) を利用することで防ぐことが可能である．本論文は HSTS による対策を回避可能な sslstrip 攻撃の拡張を提案する．提案手法により，HSTS を回避しつつ，sslstrip 攻撃と同様にユーザが入力した機密情報を盗聴することが可能となる．本論文では提案攻撃手法の対策についても考察する．

キーワード：sslstrip, HTTPS, 中間者攻撃, HSTS

An Enhanced Sslstrip Attack aginst HTTPS with HSTS

Takashi Setozaki[1]　Kazuto Matsuo[1]

Abstract: The sslstrip attack proposed by Marlinspike is known as a man-in-the-middle attack against HTTPS. The sslstrip attack eavesdrops on confidential communications by replacing HTTPS connection with HTTP connection between the user and the attacker. It is known that this attack is prevented by HTTP Strict Transport Security (HSTS). This paper proposes an enhanced sslstrip attack aginst HTTPS with HSTS. Even if the server takes countermeasures of the sslstrip attack by HSTS, the proposed attack can eavesdrop on confidential communication. This paper also discusses the countermeasures against the proposed attack.

Keywords: sslstrip, HTTPS, man-in-the-middle attacks, HSTS

1. はじめに

インターネット上のウェブサーバとクライアント間で HTTP プロトコル [1], [2], [3], [4], [5], [6] を利用して機密情報のやり取りを行うと，第三者による改ざんや盗聴，なりすましなどの危険を伴う．これらの危険性は暗号技術によって秘匿通信や認証を行うことで防ぐことができる．ウェブ通信では，暗号プロトコル SSL/TLS [7] を HTTP に適用した HTTPS [8] が利用されている．しかし，安全とされている HTTPS による秘匿通信を破る中間者攻撃が存在する．中間者攻撃は盗聴方法の一種であり，攻撃者が通信を行う二者間に割って入り両者になりすますことで，気づかれることなく通信内容の改ざんや盗聴を行う手法である．HTTPS に対する中間者攻撃も複数知られており [9], [10], [11], [12]，HTTPS の安全性を脅かす脅威となっている．

HTTPS に対する中間者攻撃の一つに，Marlinspike によって提案された sslstrip 攻撃 [10] がある．Marlinspike はこの攻撃を行うソフトウェア [13] をウェブ配布しているので，誰でも簡単にこの攻撃が可能である．この攻撃では，攻撃者は HTTPS 接続するように設定されているリンクを HTTP 接続するように書き換えて，ユーザと攻撃者間の通信を暗号化されない HTTP 接続にする．また，攻撃者は書き換えたリンクに対するユーザの HTTP リクエストを HTTPS リクエストに変更してサーバに送信することで，攻撃者とサーバ間の接続を通常通りの HTTPS 接続のままにする．これにより，ユーザとサーバの両方に気づかれる

[1] 神奈川大学理学部情報科学科
Department of Information Sciences, Faculty of Science, Kanagawa University, Hiratsuka, Kanagawa 259-1293, Japan

ことなく攻撃を行い，ユーザの入力した機密情報を盗聴することを可能としている．

しかし，sslstrip 攻撃は HTTP Strict Transport Security (HSTS) [14] によって防ぐことができる．HSTS は，利用しているサーバへの接続を強制的に HTTPS 接続にするセキュリティ機能である．サーバが HSTS を利用している場合，攻撃者がリンクを書き換えたとしてもブラウザが強制的に HTTP 接続から HTTPS 接続に変更してしまうため sslstrip 攻撃が失敗する．

本論文では，HSTS による対策を回避可能な sslstrip 攻撃の拡張を提案する．これにより，攻撃者は sslstrip 攻撃よりも広範囲を攻撃対象とすることが可能となる．また，提案手法の適用範囲を検討し，提案手法の対策も考察する．

本論文では，2 節で sslstrip 攻撃を紹介し，3 節で HSTS を用いた sslstrip 攻撃の対策を紹介する．そして，4 節で HSTS による対策を回避可能な sslstrip 攻撃の拡張を提案し，5 節でその攻撃方法の適用範囲を検討する．さらに，6 節ではその攻撃方法の対策を考察する．最後に 7 節で本論文をまとめる．

2. Sslstrip 攻撃

本節では，本論文で提案する攻撃手法の元となる sslstrip 攻撃の概要を示し，後の提案手法の説明に利用するために sslstrip 攻撃の具体的な攻撃例を示す．

2.1 Sslstrip 攻撃の概要

Sslstrip 攻撃 [10] は，2009 年に Marlinspike によって提案された HTTPS に対する中間者攻撃である．ログイン機能などを実装している一般的なウェブサイトは，HTTP 接続によって取得させるページ内にログインページなどの HTTPS 接続させるリンクを配置していることが多い．Sslstrip 攻撃では HTTPS 接続するように設定されているリンク（`https://`で始まるリンク）を HTTP 接続するように設定されているリンク（`http://`で始まるリンク）に書き換えて攻撃を行う．ユーザと攻撃者間の HTTPS 接続を HTTP 接続にすり替えることで，ユーザと攻撃者間の通信が暗号化されず，攻撃者は通信内容を盗聴することが可能となる．

以下では sslstrip 攻撃の概要を説明する．攻撃者はあらかじめ中間者としてユーザとサーバの間に存在するものとする．

まず，攻撃者はユーザとサーバ間の HTTP 通信を中継し，サーバから送信された html ファイル内のリンクに含まれる “`https://`” を “`http://`” に書き換える．そして，書き換え後の URL とユーザの IP アドレスのペアを全て事前に準備したリスト（書き換えリスト）に保存する．次に，攻撃者はユーザからサーバへ送信される全てのリクエストを監視し，リクエストを送信したユーザの IP アドレス

```
<html>
<head>
</head>
<body>
WELCOME PAGE!
<a href="https://login.example.com/login.html">login!</a>
</body>
</html>
```

図 1 サーバのトップページの例 : example.com/index.html のソースコード

Fig. 1 Example of server's top page: source code of example.com/index.html.

```
<html>
<head>
</head>
<body>
Login Page!
<form action="https://login.example.com/user.php" method
="post">
ID <input type="text" name="id" value=""><br>
PASSWORD <input type="password" name="password" value=
""><br>
 <input type="submit" value="send"><br>
</form>
</body>
</html>
```

図 2 サーバのログインページの例 : login.example.com/login.html のソースコード

Fig. 2 Example of server's login page: Source code of login.example.com/login.html.

```
<html>
<head>
</head>
<body>
<?php
   $id = $_POST['id'];
   printf("id:$id's page<br>");
?>
</body>
</html>
```

図 3 サーバのユーザページの例 : login.example.com/user.php のソースコード

Fig. 3 Example of server's user page: Source code of login.example.com/user.php.

とリクエスト先の URL が書き換えリスト内に存在する場合，その HTTP リクエストを HTTPS リクエストに変更しサーバに送信する．また，そのリクエストに対するレスポンスを HTTPS 接続でサーバから受け取った攻撃者は，それを HTTP 接続でユーザに中継する．こうして，ユーザと攻撃者間が暗号化されない HTTP 接続，攻撃者とサーバ間が暗号化される HTTPS 接続になる．この状態を維持しつつ攻撃者は両者になりすますことで，ユーザとサーバに気づかれることなくユーザの入力した機密情報などを盗聴することができる．

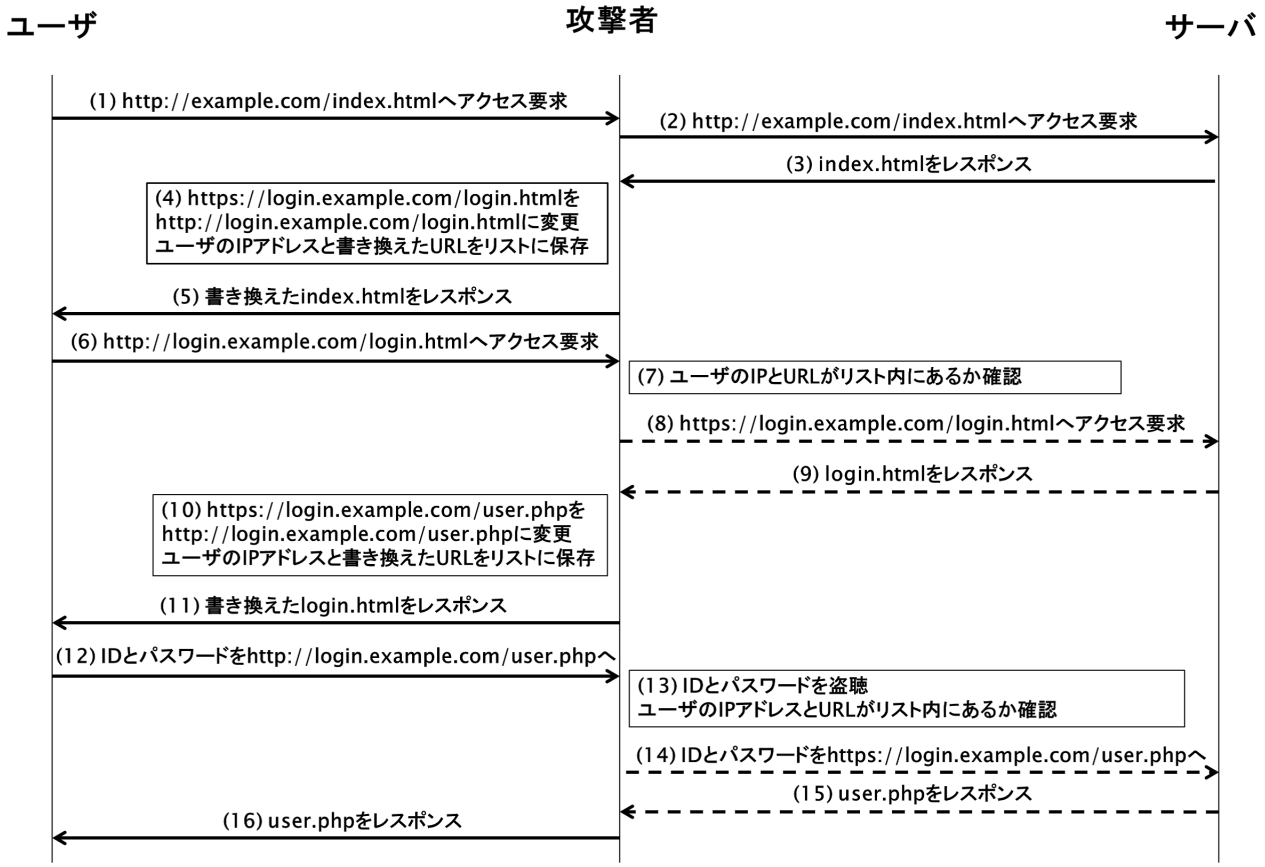

図 4　Sslstrip 攻撃の流れ（実線矢印：HTTP 接続, 破線矢印：HTTPS 接続）

Fig. 4　Flow of sslstrip attack. (Solid arrows: HTTP connection, Dashed arrows: HTTPS connection)

2.2　Sslstrip 攻撃による攻撃例

ここでは，後の提案手法の説明に利用するために，sslstrip 攻撃による具体的な攻撃例を示す．

サーバは，HTTP 接続でアクセスさせるページのドメインとして `example.com`，HTTPS 接続でアクセスさせるページのドメインとして `login.example.com` を利用しているとする．ユーザが，ログインページにアクセスするときは，まず `http://example.com` にアクセスし，そのページ内にある `https://login.example.com` へのリンクを利用することでログインページに遷移する．また，サーバのウェブページが `index.html`, `login.html`, `user.php` の 3 ファイルで構成されているものとする．ここで，ファイル `index.html` は，HTTP 接続で取得され，`login.html` への HTTPS 接続のリンクを持つページである．このページのソースを**図 1** に示す．ファイル `login.html` は，HTTPS 接続で取得され，ユーザが ID とパスワードを入力するフォームを持ち，入力された情報を `user.php` へ送信するページである．このページのソースを**図 2** に示す．ファイル `user.php` は，HTTPS 接続で取得され，ユーザが入力した ID を表示するページである．このページのソースを**図 3** に示す．

まず，ユーザが図 1 の `http://example.com/index.html` にアクセスし，ID とパスワードを入力するために “`login!`” ボタンをクリックし，図 2 の `https://login.example.com/login.html` にアクセスしたとする．次に，ユーザがファイル `login.html` 内にある入力フォームに ID とパスワードを入力し，図 3 のファイル `user.php` に送信すると，入力した情報は HTTPS 接続よって暗号化された状態でサーバに送信される．したがって，第三者によるデータの盗聴や改ざんは困難である．ユーザが入力した ID とパスワード付きのリクエストを受け取ったサーバは，ファイル `user.php` をレスポンスする．ファイル `user.php` も HTTPS 接続よって暗号化されユーザに送信される．この状況における sslstrip 攻撃を**図 4** に示す．図中の実線矢印は HTTP 接続，破線矢印は HTTPS 接続を示す．ここで，攻撃者はあらかじめ中間者としてユーザとサーバの間に存在するとする．また，攻撃者は書き換えリストを事前に準備しているものとする．

以下では，図 4 に示した攻撃の手順を説明する．

まず，(1) ユーザが `http://example.com/index.html` にアクセスするためにリクエストをサーバに送信すると，(2) 攻撃者はそれをサーバに中継する．(3) サーバがリク

エストに対するレスポンスとしてファイル index.html を送信し，(4) そのレスポンスを受け取った攻撃者は，ファイル index.html 内にある “https://” で始まる URL である “https://login.example.com/login.html” を “http://login.example.com/login.html” に書き換える．このとき，攻撃者はユーザの IP アドレスと書き換えた後の URL を書き換えリストに保存する．そして，(5) 攻撃者は書き換えたファイル index.html をユーザに送信する．次に，(6) レスポンスを受け取ったユーザが書き換えられた URL (http://example.com/login.html) にアクセスするためにリクエストをサーバに送信すると，(7) それを中継した攻撃者は，ユーザの IP アドレスとリクエスト先の URL である “http://login.example.com/login.html” が書き換えリスト内に存在することを確認する．この例では該当する URL が書き換えリスト内に存在するため，(8) 攻撃者はユーザのリクエストを変更し，サーバに HTTPS 接続で “https://login.example.com/login.html” へのリクエストを送信する．(9) サーバがリクエストに対するレスポンスとしてファイル login.html を送信し，(10) そのレスポンスを受け取った攻撃者は，レスポンス内の “https://” で始まる URL である “https://login.example.com/user.php” を “http://login.example.com/user.php” に書き換え，ユーザの IP アドレスと書き換えた後の URL を書き換えリストに保存する．(11) 攻撃者は書き換えたファイル login.html をユーザに送信する．そして，(12) ユーザがファイル login.html 内のフォームに ID とパスワードを入力し http://login.example.com/user.php へリクエストを送信すると，(13) この暗号化されていないリクエストを中継した攻撃者はユーザの ID とパスワードを入手する．さらに，攻撃者は，ユーザの IP アドレスとリクエスト先の URL である “http://login.example.com/user.php” が書き換えリスト内に存在することを確認する．この例では該当する URL が書き換えリスト内に存在するため，(14) 攻撃者はユーザのリクエストを書き換え，サーバに HTTPS 接続でユーザが入力した ID とパスワード付きの “https://login.example.com/user.php” へのリクエストを送信する．(15) サーバがリクエストに対するレスポンスとしてファイル user.php を送信すると，(16) レスポンスを受け取った攻撃者は，ファイル user.php 内に書き換えるものがないため，そのままユーザに中継する．以降もこの流れを繰り返していく．

以上で示した例のように，攻撃者はユーザとサーバ間の HTTP 通信を中継しつつ，レスポンス内のリンクの “https://” を全て “http://” に書き換え，ユーザの IP アドレスと書き換えた URL を書き換えリストとして保存する．そして，ユーザが送信したリクエストのユーザの IP アドレスとリクエスト先 URL が書き換えリスト内に存在する場合は，そのリクエストを HTTPS リクエストに変更してサーバに送信する．これにより，ユーザと攻撃者間が HTTP 接続，攻撃者とサーバ間が HTTPS 接続となる．この結果，攻撃者はユーザとサーバ間の HTTP 通信を中継することで気づかれずにユーザの ID とパスワードなどの機密情報を盗聴することができる．

3. HSTS による sslstrip 攻撃対策

2 節で説明した sslstrip 攻撃は，HTTPS 接続するように設定されているリンクを HTTP 接続するように書き換え，ユーザと攻撃者間の HTTPS 接続を HTTP 接続にすり替える攻撃であった．この攻撃の対策として HTTP Strict Transport Security (HSTS) が知られている．本節では，HSTS [14] について説明し，それを用いた sslstrip 攻撃の対策例を示す．

3.1 HTTP Strict Transport Security (HSTS)

HSTS [14] は，ForceHTTPS [15] を基に Hodges らによって提案された，特定のドメインへの接続を強制的に HTTPS 接続にする機能である．サーバが，保護したいドメインに対する HTTPS リクエストのレスポンスに Strict-Transport-Security (STS) ヘッダを追加すると，これを受け取ったブラウザは，ブラウザ内にある HSTS 用のドメインリスト（HSTS リスト）に接続先のサーバのドメイン名を追加する．以降は，ユーザがそのドメインへ HTTP 接続しようとすると，ブラウザが強制的に HTTPS 接続に切り替える．

サーバが HSTS を利用している場合，攻撃者が HTTPS 接続するように設定されているリンクを HTTP 接続するように書き換えたとしても，ブラウザが強制的に HTTPS 接続に変更するため sslstrip 攻撃が失敗する．

3.2 対策例

ここでは，2.2 節で示した sslstrip 攻撃の例に HSTS を適用し，攻撃を無効化する例を示す．

2.2 節の攻撃に HSTS を適用した例を**図 5** に示す．HSTS で保護するドメインは，login.example.com である．そこで，ユーザのブラウザ内に HSTS リストが用意され，既にその中にドメイン login.example.com が保存されているものとする．

図 5 の (1)〜(5) は図 4 と同様のステップを踏んだとする．HSTS による対策を行うと，(6) ユーザが攻撃者によって書き換えられたリンク http://login.example.com/login.html へのリクエストを送信しようとしたときに，ユーザのブラウザの HSTS リストにドメイン login.example.com が存在するため，(7) ブラウザが強制的に HTTPS 接続に切り替えて，“https://login.example.com/login.html” へのリクエストを送信する．(8) このリクエストを受け取った攻

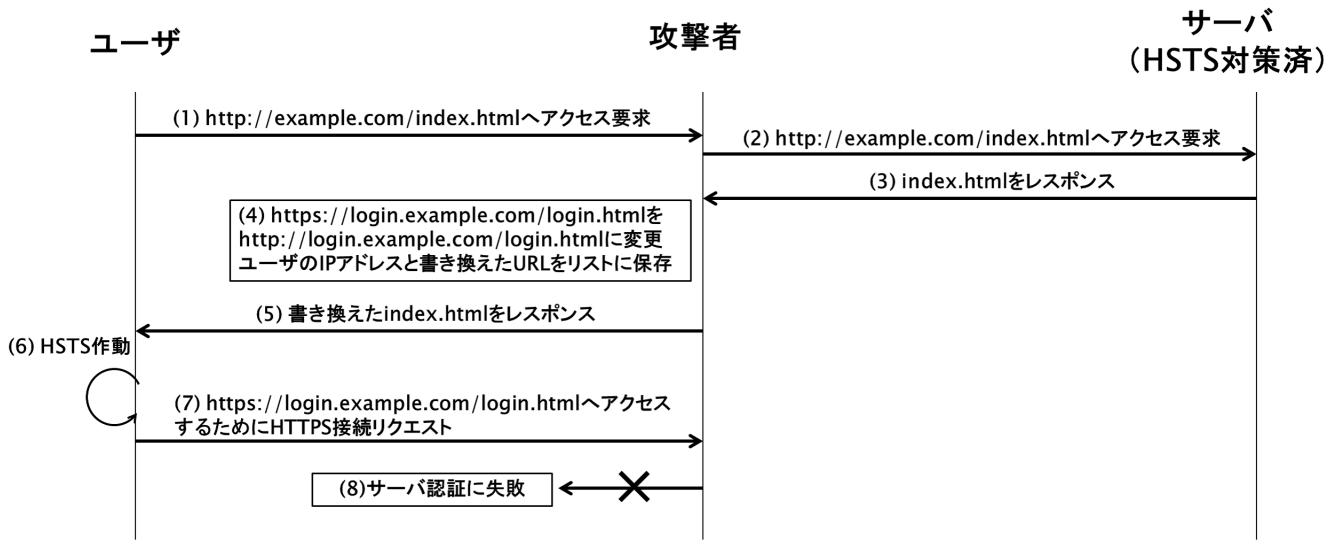

図 5　HSTS を用いた sslstrip 攻撃の対策
Fig. 5　Countermeasures of sslstrip attack by HSTS.

撃者は，ドメイン `login.example.com` のサーバ証明書を持っていないため，HTTPS 接続するためのサーバ認証をユーザに対して行うことができず，この時点で攻撃が失敗する．以上のように，HSTS によって sslstrip 攻撃を防ぐことが可能である．

4. 提案攻撃手法

2 節で示した sslstrip 攻撃は，“`https://`” を “`http://`” に書き換え，ユーザと攻撃者間の HTTPS 接続を HTTP 接続にすり替えて攻撃を行う手法であった．これにより，ユーザと攻撃者間を HTTP 接続，攻撃者とサーバ間を HTTPS 接続にし，ユーザとサーバに気づかれずに攻撃者はユーザの ID とパスワードのような機密情報を盗聴することができた．しかし，3 節で示したように HSTS を利用するとブラウザが指定されたドメインに対して HTTP 接続を HTTPS 接続に強制的に切り替えるため，sslstrip 攻撃を防ぐことができた．本節では，この HSTS による対策を回避可能な sslstrip 攻撃の拡張を提案する．また，その具体的な攻撃例を示す．

4.1 提案攻撃手法の概要

HSTS による対策を回避するためには，ユーザがアクセスしようとしているドメインがユーザのブラウザの HSTS リストに存在しないことが必要である．そこで，提案手法ではこれまでの sslstrip 攻撃と同様の書き換えに加えて，サーバから送られてきた html ファイル内の “`https://`” から始まるリンクのドメイン名を HTTPS 接続する直前に HTTP 接続していたドメイン名に書き換えることで攻撃を行う．この書き換えにより，書き換えられた html ファイル内のリンクが HSTS で保護されていないドメインになるため，攻撃者は HSTS による対策を回避することが可能となる．また，直前に接続していたドメイン名を利用しているため、ユーザが攻撃に気づきづらいことが考えられる．一方で，攻撃者がユーザのリクエストを中継する際に，そのリクエスト先が書き換えた URL であった場合は，それを元の URL に戻してサーバに送信しなければならない．ところが，URL スキームとドメイン名を書き換えただけでは，攻撃者はユーザのリクエスト先が書き換えた URL であることの判断ができない．そこで，ドメインの書き換えと同時にリンクを書き換えた目印としてインデックスを含めた架空のディレクトリ名をリンクの URL に書き加え，書き換え前の URL をインデックスと共にリスト (URL リスト) に保存しておく．ユーザが書き換えられたリンクを利用してリクエストを送信した場合，それを中継した攻撃者は架空のディレクトリ名と URL リストを基に本来の URL への HTTPS リクエストに書き換え，サーバにそのリクエストを送信する．これにより，HSTS を回避しつつ，sslstrip 攻撃と同様にユーザと攻撃者間を HTTP 接続，攻撃者とサーバ間を HTTPS 接続にし，攻撃者はユーザとサーバに気づかれずにユーザの ID とパスワードのような機密情報を盗聴することが可能となる．

4.2 提案攻撃手法の適用例

本節では，提案攻撃手法の具体的な適用例を示す．

2.2 節で示した sslstrip 攻撃に対して 3.2 節で sslstrip 攻撃を防ぐ HSTS による対策の具体例を示した．ここでは，3.2 節で示した HSTS による対策を行っているサーバに対して，4.1 節の攻撃法を適用し対策を回避して攻撃を行う手法を説明する．

2.2 節と同様に攻撃者はあらかじめ中間者としてユーザ

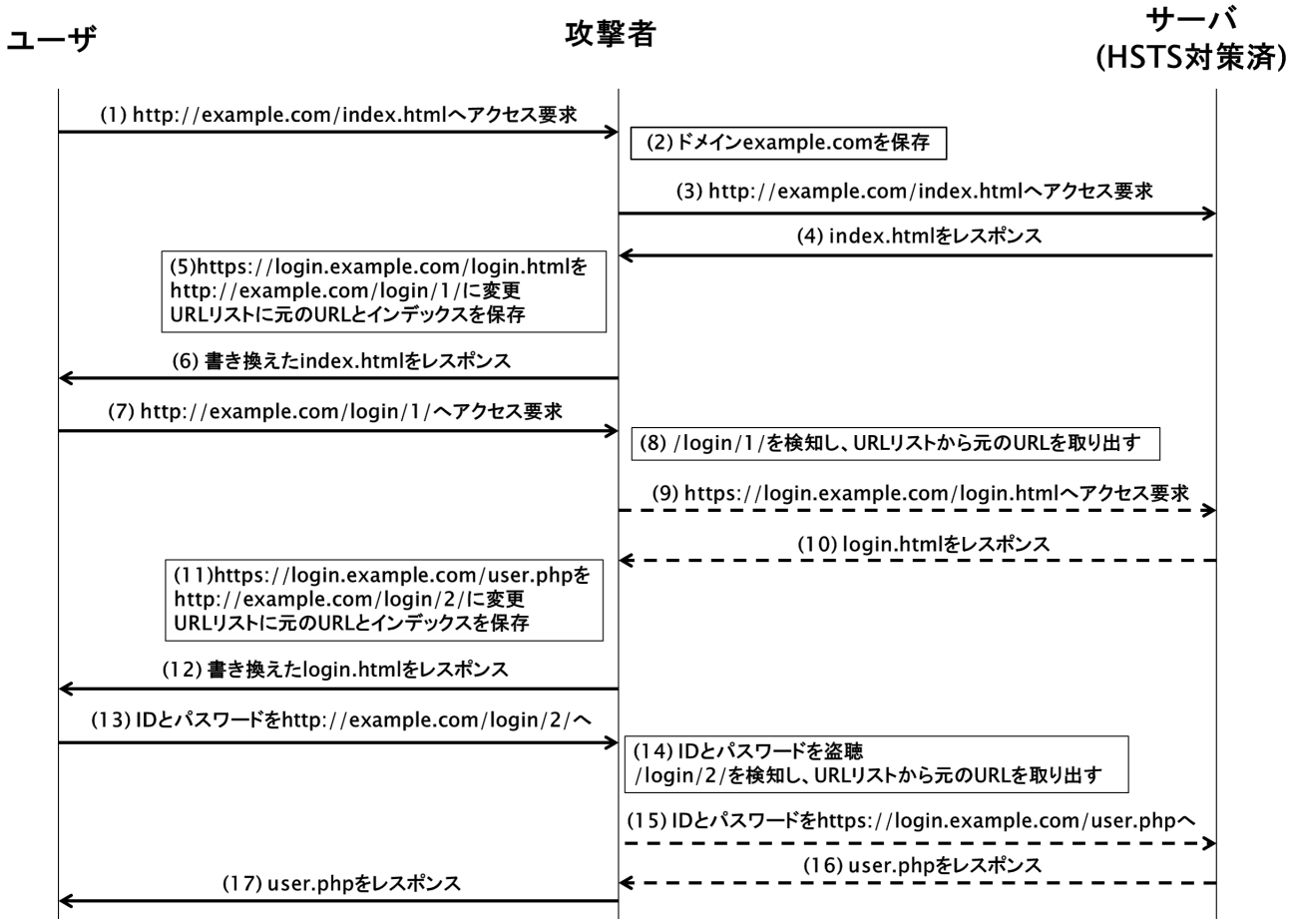

図 6 提案攻撃手法の流れ（実線矢印：HTTP 接続, 破線矢印：HTTPS 接続）
Fig. 6 Flow of proposed attack. (Solid arrows: HTTP connection, Dashed arrows: HTTPS connection)

とサーバの間に存在しているとする．

上記の設定下で，3.2 節で示した HSTS による対策を行っているサーバに対する提案手法による攻撃を**図 6** に示す．

まず，(1) は図 4 の (1) と同様であるとする．次に，図 4 とは異なり，(2) ユーザからのリクエストを中継する攻撃者は，リクエスト先のドメイン “example.com” を保存する．(3), (4) 攻撃者とサーバが図 4 の (2), (3) と同様の HTTP 通信を行う．(5) サーバからのレスポンスを受け取った攻撃者はこのレスポンス内の “https://login.example.com/login.html” を，(2) で保存したドメインと書き換えた目印としてインデックスを含むディレクトリ名 “/login/1/” を組み合わせた “http://example.com/login/1/” に書き換える [*1]．このとき，書き換え前の URL とそのインデックスを事前に準備した URL リストに保存する．(6), (7) 図 4 の (5), (6) と同様に攻撃者は書き換えたファイルをユーザに送信し，ユーザが書き換えられた URL (http://example.com/login/1/) へリクエストを送信する．この URL のドメイン “example.com” は，(1) で示したように HTTP 接続が可能であることから HSTS による保護対象外であり，ブラウザは書き換えられた URL (http://example.com/login/1/) へのリクエストを HTTP 通信のまま送信する．(8) 攻撃者がこのリクエストを受け取ると，URL に新たに追加した “/login/1/” を基に，URL リスト内のインデックスに対応する URL を確認し，本来の URL を取り出す．(9), (10) 図 4 の (8), (9) と同様に，攻撃者はユーザのリクエストを本来の URL に書き換えサーバに送信し，それに対するレスポンスを中継する．(11) 攻撃者は中継したレスポンス内の “https://login.example.com/user.php” を “http://example.com/login/2/” に書き換える．このとき，書き換え前の URL とそのインデックスを URL リストに保存する．(12), (13) 図 4 の (11), (12) と同様に攻撃者は書き換えたファイルをユーザに送信し，そのページ内にユーザが ID とパスワードを入力すると，HTTP 接続でサーバへのリクエストが送信される．(14) このリクエストを中継する攻撃者はユーザの入力した ID とパスワー

[*1] 便宜上ディレクトリ名を “login” としているが，実際のファイル名と一致しないようにランダム文字列などを加えたものが望ましい．

ドを入手し，URL に新たに追加した “/login/2/” をもとに URL リスト内のインデックスに対応する URL を確認し，本来の URL を取り出す．(15), (16), (17) では，図 4 の (14), (15), (16) と同様に攻撃者はユーザと HTTP 通信，サーバと HTTPS 通信を行う．以降もこの流れを繰り返していく．

このように，sslstrip 攻撃におけるリンクの書き換えに加えてドメイン名も書き換えることによって，HSTS による対策をされた場合にも，sslstrip 攻撃と同様にユーザと攻撃者間を HTTP 接続，攻撃者とサーバ間を HTTPS 接続にしつつ攻撃を行うことができる．この結果，攻撃者はサーバに気づかれずにユーザの ID とパスワードのような機密情報を得ることができる．

5. 提案攻撃手法の適用範囲

本節では，4 節で提案した sslstrip 攻撃の拡張手法の適用範囲を検討する．

2 節で示した sslstrip 攻撃の適用範囲は，ユーザが HSTS を利用していないサーバと HTTP 接続で取得したページ内のリンクからページ遷移して HTTPS 接続を行う全ての場合である．提案手法も sslstrip 攻撃が適用可能な状況に対しては全て適用可能である．一方で，3 節で示したように HSTS 対策が行われている場合，sslstrip 攻撃は適用が困難であるが，4 節で示したように提案手法は適用可能な場合がある．以下では，サーバが HSTS などの対策を行っている場合に対して提案手法を適用可能な範囲を示す．提案手法は，サーバが HSTS 対策を行っている場合でも，4.2 節で示したようなユーザがサーバと HTTP 接続するドメインと HTTPS 接続するドメインが異なり，ユーザが HTTP 接続でサーバから取得したページ内のリンクからページ遷移して HTTPS 接続を行う全ての場合が適用範囲となる．一方で，サーバが HSTS 対策を行っており，ユーザに HTTP 接続させるドメインと HTTPS 接続させるドメインが同一である場合は，HSTS の仕様上ユーザとサーバ間の全ての通信が HTTPS 接続になるため提案手法の適用は困難である．同様にユーザとサーバ間の全ての通信が HTTPS 接続になっている場合も，提案手法の適用は困難である．

ここまでは，単一組織のサーバに対して提案手法の適用可能な範囲を検討した．以下では，より現実的に複数組織がサーバを運用している状況を考慮し，攻撃対象サーバが無関係のサーバからリンクされている状況における提案手法の適用可能性を考察する．以下では，攻撃対象サーバのリンクを HTTP 接続で取得可能なページ内に設置している無関係のサーバが存在するとき，攻撃対象サーバの通信が全て HTTPS 接続になっていても攻撃可能であることを示す．提案手法では，4.2 節で示したように，攻撃者が無関係のサーバの HTTP 接続で取得可能なページ内のリンクを攻撃者が HTTP 接続可能なドメインである攻撃対象サーバとは無関係のサーバのドメインを含めたリンクに書き換える．そして，ユーザが提案手法によって書き換えられたリンクを利用してページ遷移すると，そのリンクのドメインは HSTS の保護対象外であるため，HTTP 接続でリクエストが送信される．したがって，攻撃者はユーザと攻撃者間の通信を HTTPS 接続から HTTP 接続にすり替えることが可能となる．攻撃対象サーバとは無関係のサーバが設置している攻撃対象サーバへのリンクが，トップページのような攻撃者が盗聴したい内容を含むページではなかったとしても，4.2 節と同様にそのトップページに対しても提案手法を繰り返し適用することが可能である．このため，一度ユーザが提案手法によって攻撃を受け，そのときにサーバから取得したページを起点にリンクを辿ってページ遷移を繰り返す場合，提案手法の繰り返し適用によって，機密情報を盗聴することが可能である．一方で，インターネット上の全ての HTTP 通信が HTTPS 通信になっている場合は，提案手法による攻撃は困難である．

以上のことから，提案手法の適用範囲は，単一組織のサーバに対しては，sslstrip 攻撃が適用可能な場合に加えて HSTS 対策を行われていてもユーザがサーバと HTTP 接続するドメインと HTTPS 接続するドメインが異なる場合である．これは，サーバが HTTP 接続と HTTPS 接続を混用している多くの場合に当てはまる．また，複数組織にまたがった適用を考慮すると，攻撃対象サーバとユーザ間の全ての通信が HTTPS 接続になっている場合でも，そのサーバと無関係のサーバが HTTP 接続で取得可能なページ内に攻撃対象サーバへのリンクを設置している場合は提案手法を適用可能である．ユーザとの通信を HTTPS 接続によってのみ行っているサーバが HTTP 接続を許可している無関係のサーバにリンクされていることは珍しくなく，広範囲に提案手法が適用可能であると考えられる．

6. 提案攻撃手法の対策

本節では，提案攻撃手法の対策方法を検討する．

まず，単一組織のサーバによる対策について検討する．提案手法は，5 節で示した通りユーザとサーバ間の全ての通信が HTTPS 接続で行われる場合は攻撃が不可能である．そこで，サーバが HTTPS 接続のみを許可するという対策が考えられる．この対策を行うと，通常はユーザから HTTPS 接続でリクエストがサーバに送信されると考えられ，提案手法による攻撃を防ぐことができる．しかし，この場合においてもユーザが HTTP 接続でリクエストを送信することが可能であり，その場合は提案手法による攻撃を防ぐことができないことに注意が必要である．

次に，HSTS を利用した対策を検討する．提案手法で HSTS を回避可能なのは，HTTP 接続するドメインと HTTPS 接続するドメインが異なり，ユーザが HTTP 接続したページ内のリンクからページ遷移して HTTPS 接続を

行う場合である．したがって，5 節で示したようにサーバが HTTP 接続するドメインと HTTPS 接続するドメインに同一のドメインを利用した上で HSTS による対策を施すことで提案手法を防ぐことができる．しかし，この場合はサーバに対する HTTP 接続が許可されないため，サーバが全ての通信を HTTPS 接続のみ許可する場合とほぼ同一であると考えられ，このような対策を行う利点は考えにくい．

その他にも，HSTS による対策を施した上でページ内のリンクの書き方を工夫する方法も考えられる．提案手法が，“http://” で始まるリンクを書き換え対象にしない点を利用して，HSTS による対策を施した上でページ内のリンクを全て “http://” から始まるリンクにすると，このリンクは提案手法では書き換えられない．ユーザがこのリンクを利用して HTTP 接続でアクセスしようとすると，HSTS によって保護されているドメインに対してはブラウザが強制的に HTTPS 接続に切り替える．これにより，図 5 の (6)〜(8) と同様の流れになり提案手法は失敗する．しかし，現在利用されている一部のブラウザのバージョンが HSTS に対応していないため，本対策では全てのユーザを提案手法から保護することができない．また，ユーザの HSTS リストを攻撃者が推測できると “http://” で始まるリンクに対してもリストに掲載されている場合はドメインを書き換えることで攻撃が可能となるため，効果は限定的である．

次に，複数組織間にわたる攻撃の対策について検討する．5 節で示したように，攻撃対象サーバの通信が全て HTTPS 接続になっていても，そのサーバとは無関係のサーバの HTTP 接続可能なページからリンクされていると，そのリンクを利用してページ遷移するユーザと攻撃対象のサーバの通信に対して提案手法による攻撃が可能となる．そのため，単一の組織での対策では攻撃を防ぐことが困難であり，インターネット上の全てのウェブ通信に対する対策が必要となる．具体的には，インターネット上の全ての HTTP 通信を HTTPS 通信に置き換えることが考えられる．これにより，前述した単一組織のサーバでの対策と異なりユーザが HTTP 接続でリクエストを送信することがないため，全てのリクエストが HTTPS 接続でサーバに送信され，完全に提案手法を防ぐことができる．

以上のことから，提案手法に対して確実な対策を行うには，インターネット上の全ての HTTP 通信を HTTPS 通信に置き換え，ユーザがサーバへ HTTP 接続でリクエストを送信することがなくなる必要があると考えられる．

7. まとめ

本論文では HSTS による対策を回避可能な sslstrip 攻撃の拡張を提案した．提案攻撃手法により，サーバが HSTS を利用して sslstrip 攻撃の対策をしていても，HTTP 接続するドメインと HTTPS 接続するドメインが異なり，ユーザが HTTP 接続したページ内のリンクからページ遷移して HTTPS 接続を行う場合には攻撃が可能となる．本論文では，提案攻撃手法の適用範囲とその攻撃の対策についても考察した．

著者等は提案攻撃手法をすでに実装済みであり，今後は様々な状況で実験を行いながら，更に提案手法の拡張を検討する予定である．また，本論文で考察した対策よりも簡便かつ効果的な対策方法も今後の課題である．

参考文献

[1] R. Fielding, J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing,” https://tools.ietf.org/html/rfc7230, RFC 7230, 2014

[2] R. Fielding, J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content,” https://tools.ietf.org/html/rfc7231, RFC 7231, 2014

[3] R. Fielding, J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests,” https://tools.ietf.org/html/rfc7232, RFC 7232, 2014

[4] R. Fielding, J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Range Requests,” https://tools.ietf.org/html/rfc7233, RFC 7233, 2014

[5] R. Fielding, J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Caching,” https://tools.ietf.org/html/rfc7234, RFC 7234, 2014

[6] R. Fielding, J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Authentication,” https://tools.ietf.org/html/rfc7235, RFC 7235, 2014

[7] T. Dierks, E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” https://tools.ietf.org/html/rfc5246, RFC 5246, 2008

[8] E. Rescorla, “HTTP Over TLS,” https://tools.ietf.org/html/rfc2818, RFC 2818, 2000

[9] F. Callegati, W. Cerroni, M. Ramilli, “Man-in-the-Middle Attack to the HTTPS Protocol,” http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4768661, IEEE Security and Privacy Magazine 03/2009

[10] M. Marlinspike,“New Tricks For Defeating SSL In Practice,” https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf, Black Hat DC 2009, 2009

[11] M. Marlinspike,“NULL Prefix Attacks Against SSL/TLS Certificates,” https://moxie.org/papers/null-prefix-attacks.pdf, Blackhat 09 and Defcon 17, 2009

[12] T. Zoller,“TLS/SSLv3 renegotiation vulnerability explained,” http://www.g-sec.lu/practicaltls.pdf, G-SEC

[13] M. Marlinspike,“Sslstrip,” https://moxie.org/software/sslstrip/, 2009

[14] J. Hodges, C. Jackson, A. Barth, “HTTP Strict Transport Security,” https://tools.ietf.org/html/rfc6797, RFC 6797, 2012

[15] C. Jackson, A. Barth, “ForceHTTPS: Protecting High-Security Web Sites from Network Attacks,” https://crypto.stanford.edu/forcehttps/, In Proceedings of the 17th International World Wide Web Conference (WWW2008) , 2008