Vol.2013-CSEC-61 No.8
Vol.2013-IOT-21 No.8
2013/5/9

Man-in-the-Browser 攻撃を行うマルウェアの安全な動的解析手法

瀬川達也[†1]　神薗雅紀[†2 †3]　星澤裕二[†2]　吉岡克成[†1]　松本勉[†1]

近年オンライン金融機関システム上での取引において，ユーザと金融機関システム間の正常なセッションに割り込み，個人情報の盗取や送金先の書き換え等の不正な操作，いわゆる Man-in-the-Browser(MITB)攻撃を行うマルウェアによる被害が増加している．このような MITB 攻撃を行うマルウェア検体を動的解析する場合，実際の金融機関システムに接続すると当該システムに何らかの影響を及ぼすリスクが存在する．そこで本発表では金融機関システムを模したダミーサーバを利用することで MITB 攻撃を行うマルウェアを安全に動的解析する手法を提案する．

A Safe Sandbox Analysis Method for Malware that Attempt Man-in-the-Browser Attacks

TATSUYA SEGAWA[†1]　MASAKI KAMIZONO[†2 †3]　YUJI HOSHIZAWA[†2]
KATSUNARI YOSHIOKA[†1]　TSUTOMU MATSUMOTO[†1]

There has been reported a class of malware that conduct Man-in-the-Browser (MITB) attacks, in which transactions between online banking system and an infected client host are manipulated. When conducting dynamic analysis of such malware, it is necessary to connect the malware sandbox with the online banking systems, which may cause a problem to the systems. In this paper, we propose a new method of malware sandbox analysis with dummy servers which imitate the online banking systems.

1. はじめに

オンライン上で金銭の取引を行う金融機関システムにおいて，ユーザが使用するブラウザと金融機関システムとの間のセッションを乗っ取り，ユーザの ID やパスワード等の個人情報を盗取したりユーザの意図した本来の取引内容を改ざんして不正な送金先に金銭を振り込ませたりするといった Man-in-the-Browser(以降，MITB)攻撃という詐欺の手口が流行している．このような攻撃の手口は 2009 年 10 月頃には既に確認されていた[1][2]が，その当時における MITB 攻撃の対象は主に海外のオンライン金融機関システムのユーザであった．しかし近年，日本国内のオンライン金融機関のみをターゲットにした MITB 攻撃を行うマルウェア(以降 MITB マルウェア)による被害の報告が増加してきている．

シマンテックの報告[3]によると，MITB 攻撃を行うトロイの木馬”Zeus”の亜種が 2013 年 2 月 13 日に発見されている．この MITB マルウェアは，感染ホストを使ってオンラインの金融取引を行うユーザに対し，本来の金融機関システムの Web ページでは表示されない偽の画面を表示させて個人情報の入力を促し，キーロガー機能によってそれを盗取して攻撃者に送るといった不正を行う．この MITB マルウェアの持つ設定ファイル中には日本国内の大手銀行 5 行のオンラインバンクの URL が記述されており，感染ホストの分布図を見るとこのマルウェアによる被害状況は日本国内に限られている．また，同じくトレンドマイクロの報告[4]においても，2013 年 2 月 14 日に日本国内 5 行にのみ対応した同様の MITB 攻撃を行うマルウェアに関して述べられている．この報告では盗取された個人情報の送信先となる不正サイトについての調査もなされており，このサイトへ 2012 年 12 月から 2013 年 2 月までに 300 件以上のアクセスがあり，日本国内の 30 以上の IP アドレスから個人情報が送信されていたと報じている．

このような MITB マルウェアは金融機関へのログインや取引情報の送信といったユーザによる操作をトリガとして動作するため，動的解析により不正活動を観測するためには MITB マルウェア検体を動作させた解析環境においてこれらの操作を行う必要がある．しかし解析対象の検体の挙動が未知である以上，検体を動作させた状態で実際の金融機関システムへアクセスすることにはリスクが伴う．

そこで，本研究では実際のオンライン金融機関システムを模倣したダミーのサーバを構築し，このダミーサーバへ解析環境からアクセスすることでオンライン金融機関システムとの通信を再現し MITB マルウェアの動的解析を自動で行う手法を提案する．

2. MITB 攻撃

論文[5]において，MITB 攻撃は以下の 2 種類に分類されている．

2.1 ID 盗取型 MITB 攻撃

ID 盗取型 MITB 攻撃とは，ユーザがオンライン金融機関

†1 横浜国立大学
Yokohama National University
†2 （株）セキュアブレイン
SecureBrain Corporation
†3 （独）情報通信研究機構
サイバー攻撃対策総合研究センター
サイバー防御戦術研究室
National Institute of Information and Communications Technology

Vol.2013-CSEC-61 No.8
Vol.2013-IOT-21 No.8
2013/5/9

にログインする際にユーザの個人情報がマルウェアにより盗取される攻撃である．具体的には，ユーザが金融機関システムにログインする際にユーザの使用する端末にあらかじめ感染した MITB マルウェアが本来のログイン画面には出現しない画面やポップアップなどを表示させる方法がある．ユーザがポップアップ画面にユーザ ID やログインパスワードといった情報を入力すると攻撃者の管理するサーバに送信される仕組みとなっている．ID 盗取型 MITB 攻撃の流れを示す．

1. 攻撃者はユーザ PC に MITB 攻撃を行うマルウェアを感染させ，以降 MITB マルウェアがユーザ PC 上で動作するブラウザのイベントを監視する
2. ユーザが MITB マルウェアに感染した PC でオンライン金融機関システムのログインページにアクセスする
3. 金融機関システムログインページへのユーザのアクセスを MITB マルウェアが検知し，ユーザに個人情報を入力させるためのポップアップ画面を出現させるように本来のログインページに改ざんを施す
4. ユーザが正規のログイン画面でユーザ ID やパスワードを入力しログインしようとすると，MITB マルウェアにより追加された偽の画面が表示され個人情報の入力をユーザに促す
5. ユーザは偽の画面に個人情報を入力しログインする
6. MITB マルウェアは入力された個人情報を攻撃者の管理するサーバに送信する
7. 攻撃者は盗取したユーザの個人情報を利用してログインし不正な取引を行う

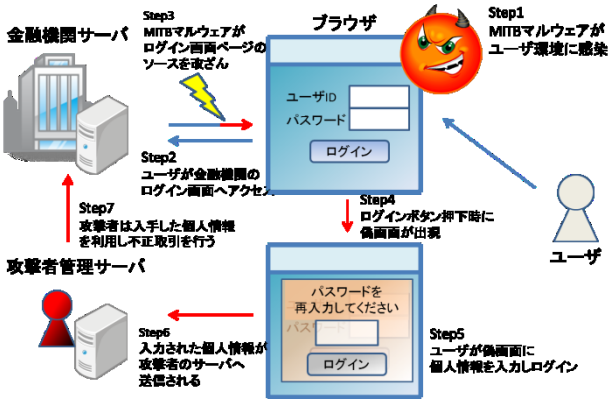

図 1　ID 盗取型 MITB 攻撃の流れ

2.2 取引内容改ざん型 MITB 攻撃

取引内容改ざん型 MITB 攻撃とは，ユーザと金融機関の間で行われる取引の内容をマルウェアがリアルタイムで改ざんする攻撃である．この攻撃を行うマルウェアは，ユーザの意図した取引内容をユーザのブラウザ上に表示しつつ実際の金融機関システムに対し改ざんした取引内容を送るという 2 方向への改ざんを行っており，ID 盗取型 MITB 攻撃より高度な攻撃と言える．取引内容改ざん型 MITB 攻撃の流れは以下のとおりである．

1. 攻撃者はユーザの PC に MITB 攻撃を行うマルウェアを感染させ，以降 MITB マルウェアがユーザ PC 上で動作するブラウザのイベントを監視する
2. ユーザがオンライン金融機関に ID とパスワードの送信など，正規の手続きを経てログインする
3. ユーザが取引内容をブラウザに入力する
4. MITB マルウェアがユーザの入力した取引内容を改ざんした上でオンライン金融機関システムへと送信する
5. オンライン金融機関システムが，受信した取引内容を確認のためにユーザ側に送信する
6. MITB マルウェアは，オンライン金融機関システムから受信した取引内容をユーザが本来意図した取引内容に改ざんしてユーザのブラウザ上に表示する
7. ユーザは，表示された取引内容を確認し,取引確定の情報をオンライン金融機関システムに送信する
8. オンライン金融機関システムが確定情報を受信し，段階 5.で MITB マルウェアにより改ざんされた取引内容をデータベースに反映させ，ユーザ側に取引完了の情報を送信する
9. ユーザ側が取引完了の情報を受信し取引が終了する

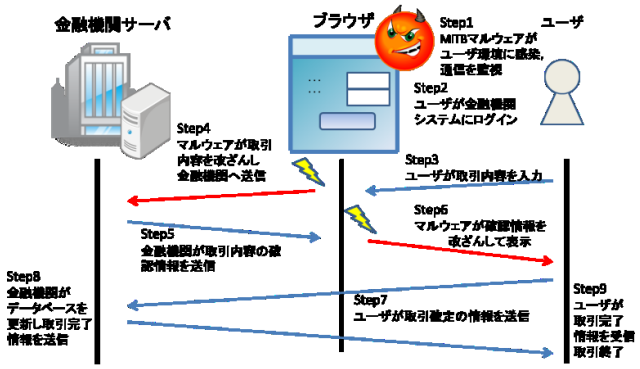

図 2　取引内容改ざん型 MITB 攻撃の流れ

3. 提案手法

3.1 概要

以上に述べたように，MITB 攻撃ではユーザの PC とオンライン金融機関システムとの間で情報の盗取や改ざんが行われる．したがって，このような MITB 攻撃を行うマルウェアの動的解析を行うには，実際にマルウェアを動作させるユーザ PC 及びその通信の宛先であるオンライン金融機関システムの両方が必要となる．しかし詳細な挙動が解明されていない MITB マルウェアの解析にあたり，実在のオ

ンライン金融機関システムとの通信を許してしまうと既存の金融機関システムに何らかの影響を及ぼす恐れがあり好ましくない．

そこで本論文では実際の金融機関システムを模したダミーサーバを構築し，これを利用して MITB マルウェアを安全に動的解析する手法を提案する．提案手法は

1. トランザクション収集フェーズ
2. マルウェア解析フェーズ

の 2 段階から成る．提案手法の全体像を図 3 に示す．

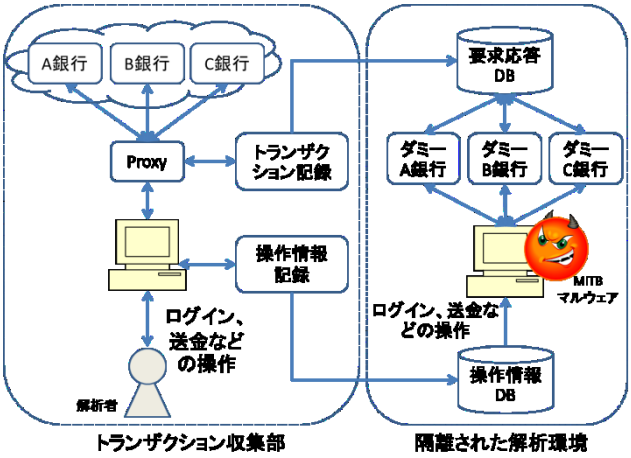

図 3　提案手法の全体像

図 3 において左側の部分が実際の金融機関システムの情報およびユーザのアクションを蓄積するトランザクション収集フェーズ，右側が蓄積された情報を利用してマルウェアの解析を行うマルウェア解析フェーズを表している．以下の節で各フェーズの詳細について解説する．

3.2 トランザクション収集フェーズ

トランザクション収集フェーズでは実際の金融機関の通信及びユーザの操作情報の記録を行う．オンライン金融機関システムがユーザ環境へと返す応答はその金融機関システムごとに異なるため，ダミーサーバを構築する際は模倣対象の金融機関システムがユーザの送信した情報に対してどのような応答を返すのかを記録し，応答を再現する必要がある．しかし，オンライン金融機関システムとの間の通信は一般的に SSL により暗号化されているため，そのままの状態では金融機関システムの応答を記録することはできない．そのため，SSL により暗号化された通信の内容を把握するために Man-in-the-Middle(MITM)という手法を用いる．

MITM 手法とは，通信を行う 2 者の間に割り込んで通信を中継し，双方向に自分を通信相手だと思わせることで気付かれることなく通信内容に介入する手法である．今回はこの MITM を実現するために 2 つのパケットリピータに SSL 暗号通信を通し，各パケットリピータでそれぞれ SSL による復号，再暗号化処理を行い，パケットリピータ間に流れる平文部分を観測することで通信の平文を取得する．MITM 手法を適用したトランザクション収集フェーズの環境を以下の図 4 に示す．

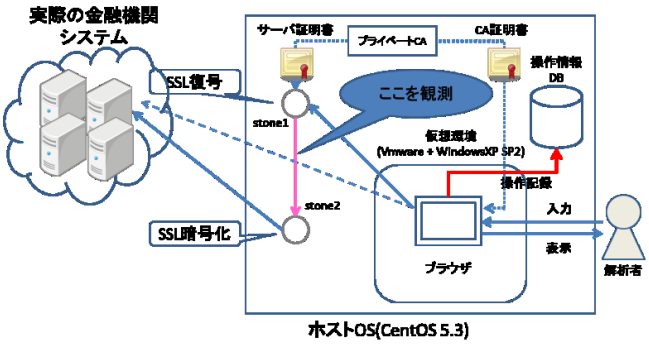

図 4　トランザクション収集フェーズ

トランザクション収集フェーズの環境はホスト OS CentOS ver5.3 上で動作する仮想マシンソフト VMware Player ver4.0.4 及びゲスト OS Windows XP Professional SP2，さらにパケットリピータ部とプライベート CA から成る．以下，環境内の構成要素について詳細に解説する．

パケットリピータ部: パケットリピータには stone[6]を用いる．ブラウザから発生した金融機関システムへの通信はまず 1 つ目の stone に転送され，ここで SSL 復号を行う．復号された平文状態の通信は 2 つ目の stone へと転送され，ここで再度 SSL 暗号化を行う．2 つ目の stone で再暗号化された通信が実際の金融機関システムへと転送される．パケットの転送は全て iptables により行い，パケットキャプチャソフト tcpdump により stone 間の平文の観測を行うことで MITM を実現する．

プライベート CA: 仮想環境内のブラウザ側から見て 1 つ目の stone を本物のオンライン金融機関システムであるように認識させるため，環境内に独自の証明書発行機関（CA)を OpenSSL[7]により構築し，この CA を用いて本物のオンライン金融機関システムのドメイン名を証明するサーバ証明書を発行する．サーバ証明書を stone に，CA 自身の証明書をゲスト OS 内のブラウザにそれぞれ適用する．

操作情報記録部:次のマルウェア解析フェーズにおけるユーザの操作を自動化するため，実際の金融機関システムへアクセスする際のユーザの一連の操作を記録する．操作情報の記録には Selenium IDE[8]を用いる．Selenium IDE はブラウザ firefox のアドオンとして提供されており，ユーザが firefox に対し行った操作を html のテーブル形式で記述され

たテストコードとして記録，及び作成されたテストコードから firefox の操作を再現することができる．Selenium IDE は firefox のアドオンであるため，firefox 以外のブラウザを使ってテストコードを作成することはできない．この自動化の環境は現在構築中であり，まだ実装はされていない．

3.3 マルウェア解析フェーズ

マルウェア解析フェーズでは，トランザクション収集フェーズで得られた実際の金融機関の応答を基にダミーサーバを構築し，これを用いて MITB マルウェアの動的解析を行う．環境はトランザクション収集フェーズで用いたホスト OS CentOS ver5.3，ゲスト OS Windows XP Professional SP2 とし，MITB マルウェアに感染した状態のゲスト OS 内からホスト OS 上で動作する金融機関システムのダミーサーバにアクセスし MITB 攻撃を再現する．マルウェア解析フェーズにおける環境を以下の図 5 に示す．

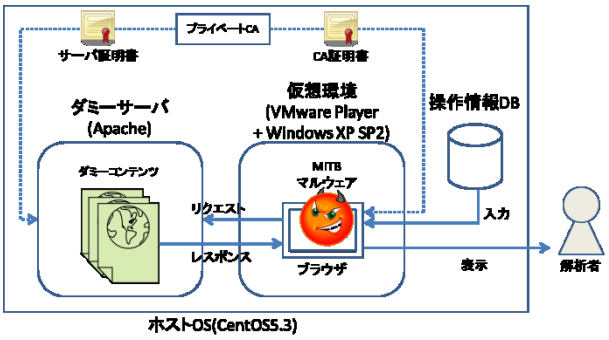

図 5　マルウェア解析フェーズ

マルウェア解析フェーズの環境の基本部分はトランザクション収集フェーズと同様にホスト OS CentOS ver5.3 上で動作する仮想マシンソフト VMware Player ver4.0.4 及びゲスト OS Windows XP Professional SP2 で構成され，さらにプライベート CA，ダミーサーバから成る．以下，環境内の構成要素について詳細に解説する．

プライベート CA: トランザクション収集フェーズで構築したプライベート CA 及びサーバ証明書をマルウェア解析フェーズでも利用する．プライベート CA によって発行した金融機関システムのサーバ証明書とサーバ秘密鍵をダミーサーバに適用する．詳細は以下ダミーサーバの項で解説する．

ダミーサーバ: ダミーサーバはホスト OS 上において HTTP サーバソフト Apache[9]により実装する．Apache は本来の金融機関システムに倣い SSL 暗号化通信を行うものとし，これは Apache のモジュール mod_ssl により実装する．環境内に独自に構築したプライベート CA により作成したサーバ証明書及びサーバの秘密鍵を Apache の SSL 設定ファイルである ssl.conf 内の SSLCertificateFile と SSLCertificateKeyFile にそれぞれ指定し，待ち受けポート番号は SSL 通信で用いられるデフォルトの 443/tcp で待ち受けるように設定する．また，ゲスト OS からダミーサーバへのアクセスは iptables の PREROUTING チェインにより本来の金融機関システムのドメインへの接続をホスト OS 上のダミーサーバ宛に書き換えることで行う．

操作情報 DB: トランザクション収集フェーズで記録したブラウザ操作情報を再現し本来ユーザが行う金融機関システムへのログインや送金といった操作を自動で行うようにする．これは具体的には，Selenium IDE でテストケースとして記録した操作をコマンドライン上から実行できる Selenium RC[8]を利用し，トランザクション収集フェーズで記録したユーザの操作を再現することで実現した．

なお，ID 盗取型 MITB 攻撃により html ソースに改ざんが加えられたかどうかを検証するため，ログイン時の一連の操作情報にページの html ソースを取得する操作を加える．これには firefox アドオンの Scrapbook[10]を用いる．Scrapbook にはショートカットキーを押下することで現在開いているページの html ソースを取得できる機能があり，これを利用してトランザクション収集フェーズで記録したテストケース内のログインボタンを押した直後の部分に Scrapbook のショートカットキーを押下する記述を加えることで html ソースの取得まで含めて自動化を行う．操作情報の記録に用いた Selenium IDE は firefox のアドオンであるため他のブラウザでは利用できないが，Selenium RC はコマンドライン上から実行可能なため，firefox で作成したテストコードを他のブラウザに適用してブラウザ操作を再現することも可能である．この自動化の環境は現在構築中であり，まだ実装はされていない．

4. 検証実験 1

4.1 実験方法

ある金融機関 A と感染ホストとの通信を改ざんすることが確認されている検体に対して，3.3 章で述べたマルウェア解析フェーズの環境により ID 盗取型 MITB 攻撃が再現できることを確認する．以下に実験方法を示す．

1. ある金融機関 A のログインページの html を取得してダミーサーバに設置する
2. 金融機関 A のログインページのドメイン名を証明するサーバ証明書をプライベート CA により発行し，サーバ秘密鍵と共にダミーサーバに適用する
3. ゲスト OS 内で MITB 攻撃を行うマルウェア検体を実行する（検体情報は付録 A.1 を参照）
4. ブラウザに金融機関 A のログインページ URL を入力

することでゲスト OS 内からホスト OS 上のダミーサーバへとアクセスを行う

5. ブラウザ上に表示されたログインページの html ソースを取得する
6. 元の html ソースと比較することで MITB 攻撃による改ざんの有無を確認する

ここで 4.において，ブラウザは Internet Explorer ver.8，Firefox ver.18，Google Chrome ver.24 の 3 種類をそれぞれ使用し，ブラウザによる攻撃の差違を調査する．なお，実験に用いたログインページは 2012 年 12 月頃に取得したもので，ログインページ URL は，現在は使われていない旧 URL である．この旧 URL は現在用いられている URL と構造が非常に近いが，パラメータの部分が多少異なっている．

4.2 実験結果

3 種類のブラウザを使い一連の実験をそれぞれ行ったところ，html ソースへの改ざんが確認できたのは Internet Explorer を用いたときのみであった．この時，以下の 3 点で html ソースに変化が見られた．

1).ログインボタン部の記述の改ざん

```
441 <td rowspan="2" width="186">
442 </td>
443 <td align="center" rowspan="2" width="114">
444 <div class="margintop15">
445 <input id="LOGIN_idJsp79" name="LOGIN_idJsp79" type="submit" value="ログイン" onclick="clear_LOGIN
446 </td>
447 <td height="0" width="186">
448 </td>
449 </tr>
450 <input type="hidden" name="LOGIN_SUBMIT" value="1" /><input type="hidden" name="jsf_sequence" valu
    type="text/javascript"><!--
451 function clear_LOGIN() {
```

図 6　改ざん前の html ソース

```
441 <td rowspan="2" width="186">
442 </td>
443 <td align="center" rowspan="2" width="114">
444 <div class="margintop15"><input type="submit" value="ログイン" onclick="Check();return false;" disabled="
445 <input id="LOGIN_idJsp79" name="LOGIN_idJsp79" type="submit" value="ログイン" onclick="clear_LOGIN(
446 </td>
447 <td height="0" width="186">
448 </td>
449 </tr>
450 <input type="hidden" name="LOGIN_SUBMIT" value="1" /><input type="hidden" name="jsf_sequence" value
    type="text/javascript"><!--
451 function clear_LOGIN() {
```

挿入

図 7　MITB 攻撃による改ざん後の html ソース

図 6 は MITB 攻撃発動前の元の html ソースの一部である．青枠部分内にユーザがログインする際に押すログインボタンに関する記述があるが，図 7 に示す MITB 攻撃後の html ソースと比較すると元のソースには無かった記述が追加されていることが分かる．これはユーザがログインボタンを押下した際に画面にポップアップを表示させるための記述である．

2).ユーザへの注意喚起の記述の削除

```
532 <table border="0" cellpadding="0" cellspacing="0" width="191">
533 <tr>
534 <td background="/rb/fes/img/mainservice/Security/LoginAuthentication/Login/table02_middule
535 <table border="0" cellpadding="0" cellspacing="0" width="160" class="margintop10">
536 <tr valign="top">
537 <td width="1%" class="txtline01">
538 ・</td>
539 <td width="99%" class="txtline01 bold">
540 当行のログイン画面には<br>
541
542 <span class="c01 smediumbold">暗証番号</span>を入力するものはありません。</td>
543 </tr>
544 </table>
545 <table border="0" cellpadding="0" cellspacing="0" width="160" class="margintop10">
546 <tr valign="top">
547 <td width="1%" class="txtline01">
548 ・</td>
549 <td width="99%" class="txtline01">
550 ログインパスワードと<br>
551
552 <span class="c01 smediumbold">暗証番号</span>を同時に入力する画面はありません。</td>
553 </tr>
554 </table>
```

図 8　改ざん前の html ソース

```
532 <table border="0" cellpadding="0" cellspacing="0" width="191">
533 <tr>
534 <td background="/rb/fes/img/mainservice/Security/LoginAuthentication/Login/table02_mi
535
536 <table border="0" cellpadding="0" cellspacing="0" width="160" class="margintop10">
537 <tr valign="top">
538 <td width="1%" class="txtline01">
539 ・</td>
540 <td width="99%" class="txtline01">
541 ユーザID、ログインパスワード、PCには保存しないでください。</td>
542 </tr>
543 </table>
544 <table border="0" cellpadding="0" cellspacing="0" width="160" class="margintop10">
545 <tr valign="top">
546 <td width="1%" class="txtline01">
547 ・</td>
548 <td width="99%" class="txtline01">
549 生年月日、電話番号など推測されやすい文字列を使用しないでください。</td>
550 </tr>
551 </table>
552 </td>
553 </tr>
554 </table>
```

図 9　MITB 攻撃による改ざん後の html ソース

図 8 はログイン時に不正な画面に個人情報を書き込まないように金融機関 A が注意をユーザに呼びかける記述である．このような記述は金融機関 A のログイン画面に常に表示されており，MITB 攻撃により出現する偽画面の特徴（ポップアップ表示される，個人情報の再入力を求める等）を示してユーザへの注意喚起を促す目的で金融機関側が用意したものである．図 9 は図 8 と同じ行数部分の MITB 攻撃後の html ソースであるが，2 つを比較するとユーザへの注意書きを促す記述が MITB マルウェアにより一部削除されていることが分かる．

3).スクリプトの挿入

```
654     var s_code = s.t();
655     if (s_code) document.write(s_code);
656 //-->
657 </SCRIPT>
658 <!-- Omniture -->
659
660
661
662
663
664 </CENTER>
665
666
667
668
669
670
671 <!-- MYFACES JAVASCRIPT -->
672
673 </BODY>
674 </HTML>
675
```

図 10　改ざん前の html ソース

```
631 <!-- Omniture -->
632
633
634
635
636
637 </CENTER>
638
639
640
641
642
643
644 <!-- MYFACES JAVASCRIPT -->
645
646 </BODY>
647 </HTML><style type="text/css">
648
649  .initialtext{
650  color:#555;
651  font-size:15px;
652  font-family:    "Hiragino Kaku Gothic Pro", "ヒラギノ角ゴ Pro W3", "MS Pゴシック";
653  }
654
```

挿入

図 11　MITB 攻撃による改ざん後の html ソース

図 10 は改ざん前の html ソースの末尾部分, 図 11 は MITB 攻撃発動後の html ソースの末尾部分であるが，図 10 では </BODY></HTML>タグで記述が終了しているのに対し図 11 ではこれ以降にも記述が続いている．これは，改ざんされたログインボタン押下時に表示される偽画面のスクリプトが MITB マルウェアにより挿入されたものであると考えられる. なお図 10 と図 11 とで</BODY></HTML>タグのある行数が異なっているのは，図 9 で示したように MITB マルウェアによりユーザへの注意書きの記述が削除された分だけ行がずれたためである. 改ざん前の html ソース全体の行数は 675 行，MITB 攻撃後の html ソース全体の 965 行であった.

4.3 考察

今回の実験では使用するブラウザとして Google Chrome, FireFox, Internet Explorer の 3 種類を用いたが，ID 盗取型 MITB 攻撃による html ソースへの改ざんが確認できたのは Internet Explorer を用いたときのみであった．ただし MITB 攻撃が発動しなかった 2 つのブラウザは，バージョンの違いによって攻撃の成功可否が異なる可能性も考えられる.

また MITB 攻撃による html ソースの改ざんは上記の 3 点で行われていたが,ログインボタン押下時のアクションや偽画面表示のスクリプト追加といった直接的な改ざんだけではなく，偽画面に関するユーザへの注意喚起の記述を削除するといった MITB 攻撃に直接は関係のない部分に改ざんが行われていることも判明した.

5. 検証実験 2

5.1 実験方法

検証実験 1 の結果から，実験で用いた検体に対して金融機関 A のダミーサーバを用いて ID 盗取型 MITB 攻撃を再現できることが確認できた．そこで検証実験 2 ではこの検体が金融機関 A 以外に攻撃対象を有するかを調査する．以下に実験方法を示す.

1. 国内のオンライン金融機関 29 行について，検証実験 1 と同様に各金融機関についてそれぞれサーバ証明書を発行する
2. ダミーサーバに金融機関のログインページ，秘密鍵，サーバ証明書を適用する
3. ゲスト OS 内で MITB マルウェアを実行する（検体は検証実験 1 で使用したものと同様）
4. ブラウザに各金融機関のログインページ URL を入力することでゲスト OS 内からホスト OS 上のダミーサーバへとアクセスを行う
5. ブラウザ上に表示されたログインページの html ソースを取得する
6. 元の html ソースと比較することで MITB 攻撃による改ざんの有無を確認する
7. 段階 2.～6.を各金融機関について行う

なお段階 3.において，今回用いるブラウザは Internet Explorer ver8 とした．また各金融機関のログインページ及び URL は基本的には 2013 年 4 月 5 日現在のものを用いた.

5.2 実験結果・考察

実験に用いた 29 行のログインページの中で MITB 攻撃による改ざんが認められたのは 2 行であり，そのうち 1 行は検証実験 1 でマルウェアによる改ざんが確認できた金融機関 A である．改ざんが行われたもう 1 行の金融機関を金融機関 B とする.

まず金融機関 A については, 2013 年 4 月 5 日現在の URL でアクセスした場合は MITB 攻撃が確認されず，検証実験 1 で用いた旧 URL でアクセスした場合にのみ検証実験 1 と

同じ改ざんが行われた. 旧 URL でアクセスした場合はコンテンツが新しい場合でも改ざんが行われたため，この検体は金融機関 A の古いログインページの URL を攻撃対象の識別情報として利用していると考えられる.

金融機関 B への MITB 攻撃に関してはログインボタン部分の改ざんとソース末尾へのスクリプトの追加が行われた. 以下に金融機関 B へのスクリプト追加の様子を示す.

```
164 <div class="footerarea padTop10">
165     <div class="footer-normal">
166         <address>Copyright (C)
167     </div>
168 </div>
169
170
171
172
173
174 </body>
175 </html>
176
```

図 12　改ざん前の html ソース

```
164 <div class="footerarea padTop10">
165     <div class="footer-normal">
166         <address>Copyright (C)
167     </div>
168 </div>
169                                    挿入
170
171
172
173
174 </body>
175 </html><style type="text/css">
176
177  .initialtext{
178  color:#555;
179  font-size:15px;
180  font-family:     "Hiragino Kaku Gothic Pro", "qMmpS Pro W3", "lr oSVbN";
```

図 13　MITB 攻撃による改ざん後の html ソース

図 12,13 を比較すると，検証実験 1 で確認できた金融機関 A へのスクリプト挿入と同様の改ざんをしていることが分かる．全体の行数で見ると改ざん前の html ソースは 176 行，MITB 攻撃後は 520 行に増加していた.

以上より，この MITB マルウェア検体は少なくとも国内の 2 つの金融機関ログインページに対して，ログインボタン改ざんやスクリプトの挿入といった MITB 攻撃を行うということが確認できた.

6. 関連研究

本研究に類似の研究として，オンラインの正規サービスを悪用するマルウェアの動的解析手法が既に論文[11]で提案されている．この論文では解析環境内に本来のオンラインサービスを模擬するダミーサーバを設置し，マルウェア検体を実行した犠牲ホストから実オンラインサービスへの通信をダミーサーバへと転送し動的解析を行っている．またこのダミーサーバはマルウェア検体からの応答をデータベースに蓄積しており，マルウェアからの要求が既知のものであれば対応した応答を返し未知の要求であれば実サービスに送信して実サービスからの応答を記録することでマルウェアからの要求に対する応答を学習する．この手法は犠牲ホスト内のマルウェアが自発的に発生させる攻撃の解析を対象としており，金融機関システムへのログインや送金時の操作時にユーザが起こすブラウザへのアクションに対して発生する MITB 攻撃の解析を対象とする点で本研究とは異なっている.

7. おわりに

オンライン金融機関システムとユーザとのセッションの間に不正な操作を紛れ込ませ個人情報の漏えいや取引内容の改ざんをする MITB 攻撃に関して，金融機関システムを模したダミーのサーバを環境内に構築し MITB 攻撃を行うマルウェアを外部システムに影響を与えること無く安全に動的解析を行うシステムを提案し，検証実験として ID 盗取型の MITB 攻撃について，特定条件下でこの攻撃が発生することをダミーサーバと仮想環境を用いて確認した．今後はユーザのアクションを充実させ，より多様な MITB マルウェアを解析することで提案手法の有効性を示したい.

謝辞　本研究の一部は，総務省情報通信分野における研究開発委託／国際連携によるサイバー攻撃の予知技術の研究開発／サイバー攻撃情報とマルウェア実体の突合分析技術／類似判定に関する研究開発により行われた.

参考文献

1)　日立ソリューションズ情報セキュリティブログ
http://securityblog.jp/words/790.html
2)　ITmedia エンタープライズ
http://www.itmedia.co.jp/enterprise/articles/0909/29/news049.html
3)　Symantec Connect
http://www.symantec.com/connect/ja/blogs/zeus
4)　TrendLabs SECURITY BLOG,
http://blog.trendmicro.co.jp/archives/6702
5)　鈴木 雅貴, 中山 靖司, 古原 和邦, “インターネット・バンキングに対する Man-in-the-Browser 攻撃への対策「取引認証」の安全性評価” 暗号と情報セキュリティシンポジウム 2013, 2013.
6)　Simple Repeater stone
http://www.gcd.org/sengoku/stone/Welcome.ja.html
7)　OpenSSL
http://www.openssl.org/
8)　SeleniumHQ Browser Automation
http://docs.seleniumhq.org/
9)　Apache
http://httpd.apache.org/
10)　SCRAPBOOK :: Firefox Extension
http://amb.vis.ne.jp/mozilla/scrapbook/?lang=ja
11)　村上 洸介, 吉岡 克成, 松本 勉, “オンラインサービスを悪用するマルウェアに対する動的解析手法の提案“ 電子情報通信学会技術研究報告. ICSS, 情報通信システムセキュリティ, 2010.

付録 A.1 実験に用いたマルウェア検体情報

ハッシュ値 (SHA256)	0f011ce5e582cdaa620f2da93cb52bcc19a30fef8642daab1b81e0955cefddda
ハッシュ値(SHA1)	967d5689ba091eed2933eb9406ddb5107957f5e8
ハッシュ値(MD5)	e3797ea82090ed385d3ec8fc73cdac96
ファイルサイズ	385.0 KB
ファイルタイプ	Win32 EXE
分析日時	2012/10/30

.