クローキング広告の置き換えによるマルウェア対策手法の提案

前田 浩[†]　並木 孝博[††]　宇田 隆哉[†]

近年，クローキングによる Web ページの隠蔽や偽装が多く行われている．この手法を用いて Web サイトにマルウェアを仕掛けることや，広告のクローキングを装ってマルウェアを送り込むことで，ユーザに被害を与えるなどといった攻撃が問題となっている．このようなマルウェアの被害を防ぐために，様々な研究が行われている．しかし，既存の手法では第三者のサービスを利用するためにプライバシーの問題が挙げられる．また，クローキング部分のコンテンツを表示させないといった対策により，ページレイアウトに問題がでてしまうことが挙げられている．そこで，本稿では HTML で「広告のクローキング」が行われている部分を検出し，ページレイアウトに影響を与えないように該当コンテンツを置き換える手法について提案する．それにより，クローキング広告を装ったマルウェアの対策を行う．

Proposal of method against malware with replacing advertisement on cloaking

HIROSHI MAEDA[†]　TAKAHIRO NAMIKI[††]
RYUYA UDA[†]

In recent years, a lot of Web pages conceal or disguise its contents using cloaking technology. Some of them try to infect user PC with malware via masqueraded contents or Internet advertisement. This kind of attack causes serious damage to the user PC and is known as a large problem. On the other hand, there are several researches and services to prevent the attack. However, since these methods use third-party services, privacy issues tend to arise. Moreover, only detecting and hiding cloaked contents will destroy the layout of the original Web pages. Therefore, in this paper, we propose a new method for detecting and replacing cloaked advertisement without destroying layout of pages. Our method can protect user PC from malicious malware.

† 東京工科大学 コンピュータサイエンス学部, Tokyo University of Technology School of Computer Science.
†† 株式会社 ピーシーデポコーポレーション, PC DEPOT CORPORATION.

1. はじめに

近年，クローキングによる Web ページの隠蔽や偽装が多く行われている．クローキングとは，サーバがユーザごとのリクエストにより配信するコンテンツの内容を動的に変更する手法である．本来，クローキングは検索エンジン最適化のために用いられるが，悪用することでユーザごとに悪質なコンテンツを振り分けて配信することも可能である．この手法を用いて Web サイトにマルウェアを仕掛けることや，広告のクローキングを装ってマルウェアを送り込むことで，ユーザに被害を与えるサイバー攻撃が問題となっている．このような仕掛けが施された Web サイトは，ページを閲覧するだけでマルウェアに感染してしまう．

こういった悪意のある Web サイトに対して，キッズ Goo[1]のようなホワイトリストを用いたアクセス制御サービスなどで対策が行われるようになった．しかし，クローキングの技術を利用することで，ホワイトリストを作成するサイトからのアクセスに対しては安全なコンテンツを見せつつ，一般のユーザのアクセスに対しては攻撃を仕掛けることが可能となってしまう．そのため，ホワイトリストに載っているサイトであっても一概に信頼することはできない．他にも，マルウェアの対策として aguse Gateway[2]というサービスが存在する．このサービスを利用し，Web サイトの安全性が確認できても，閲覧先とやり取りするデータは全て aguse Gateway のサーバが中継する形となるため，個人情報をサーバがキャッシュとして残していた場合，プライバシーに問題がある[3]．第三者を介さないものとしては，同じページのクローキング部分の差異を取得し，該当部分を差し替えることで対策を行う方法がある．しかし，インターネットにある広告などの画像はクローキングされているものが多く，この部分を置き換えてしてしまうと全体のページレイアウトが崩れてしまう．そこで，本稿では HTML の「広告のクローキング」が行われている部分を検出し，ページレイアウトに影響を与えないように該当コンテンツを置き換える手法について提案する．それにより，クローキング広告を装ったマルウェアの対策を行う．

2. 関連技術と関連研究

2.1 関連技術

ユーザごとのリクエストによって動的にコンテンツの内容を変更する技術は，クローキング手法といわれている．これらの多くは，SEO（Search Engine Optimize）対策として用いられている．SEO 対策とは，Web サイトの管理者が自分のサイトを検索エンジンでその順位を優位に働かせるために，検索エンジンとユーザで異なるコンテンツを配信する手法である．しかし，クローキングの技術を悪用するとユーザごとに悪質なコンテンツを見せることが可能なため，今日では様々な問題となっている[4].

2.2 関連研究

マルウェアに対する手法として, Google インスタントプレビュー[5]と aguse Gateway という 2 つの手法が挙げられる．Google インスタントプレビューは，Google の検索結果で表示されるアイコンをクリックすると瞬時に画面にプレビューを表示する機能である．これにより，リンク先を視覚的に表示することが可能であるため，ユーザがリンク先にアクセスする前にどのようなページであるかを判別することができる．そのため, 悪意のあるクローキングによるマルウェアの被害を防ぐことができる. しかし，これは Google の検索結果に対してのみであり，検索結果以外のリンクや，ユーザが他の検索エンジンを使用した場合に対してこの機能を利用することはできない．aguse Gateway では，URL を入力するとそのサイトを一枚の画像に変換し，画像データのみを利用者に表示する技術である．こちらも Google インスタントプレビュー同様で，クローキングによるマルウェアの被害を防ぐことができ，さらに対象サーバの情報についても取得することができる．しかし，一部レイアウトが崩れてしまうページがあるため，ユーザはリンク先の Web ページの情報を正しく得ることができない．また，いずれのシステムにおいても通信の中間にサーバを置く構成となっている．送受信するデータによっては個人情報が含まれており，これらの情報をサーバがキャッシュや履歴として残している可能性がある．そのため，サーバの管理者が容易にこれらのデータを収集できてしまい，プライバシーに問題がある．

マルウェアに関しては，動的解析の研究が非常に多く行われている[6-7]．しかし，近年のマルウェアはどのような環境で実行されているかを判断し，不意な環境においては挙動を見せないように工夫されているものが多い[8]．そのため，このようなサービスからのアクセスであることを判断し挙動を変えるマルウェアに対しては，完全な対策を行うことが難しい．

3. 提案手法

関連研究で述べた Google インスタントプレビューでは, 検索エンジンに依存する技術のため，他の検索エンジンからは利用できない．したがって，ユーザの汎用性が損なわれることが問題である．また，aguse Gateway では一枚の画像に変換する際にページのレイアウトが崩れてしまう可能性がある．さらに，送受信するデータにプライバシーの対策が必要である．つまり，マルウェアの攻撃を防ぎ，同時にこれらの問題を解決する手法が求められる．

そこで本提案では，Web サーバから送られてくるコンテンツを解析することで大きさを取得し，クライアントに保存された同じ大きさの画像と置き換えることで，クローキングのマルウェアによる攻撃とページレイアウトの問題を解決する．さらに，これらの機能をブラウザのアドオンとして実装することで検索エンジンの依存を解消する．これにより，中間サーバを設ける必要がないため情報漏洩の懸念がなく，プライバシーの問題を解決することができる．本手法では，クローキングページの悪意の有無に関わらず差異の部分を置き換えることで，特にクローキング広告を装ったマルウェアに対する危険性が低減される．また，悪意のないクローキングページでは，置き換えによって変化する対象が広告のみとなるため，大きな影響を与えない．

3.1 クローキング広告の事例と対策手法

現在 Web で用いられているクローキング広告の種類は，主に以下の 3 つのものが考えられる．

（1）イメージタグで参照されている画像ファイルだけが異なる場合
（2）JavaScript で表示される広告が，Script ごと異なる場合
（3）Flash（SWF : Small Web Format）で配信される広告が異なる場合

本提案手法では，これらの広告に対して以下の方法によって対応する．

（1）ユーザが HTTP リクエストを Web サーバに対して要求した際に，サーバが HTML の構造で書かれたイメージタグに対して動的に変更を施すことで，要求する度に異なる HTML のページが返される状況である．イメージタグが変更されると，図 1 のように異なる画像が表示される．

この場合では，あらかじめ 2 つの HTML の差異を取得し，変化している部分の URL を安全なものに書き換える．具体的には，イメージタグ内の src 属性の属性値が異なる．この属性値が画像の URL である．したがって，この URL を任意のものに書き換えることで対応する．ここで画像の幅と高さを取得して，レイアウトを崩さないように置き換えを行う．width と height 属性がない場合には，タグ内から画像ファイルの幅と高さを取得することができないので, 画像のヘッダ情報から幅と高さを取得する．

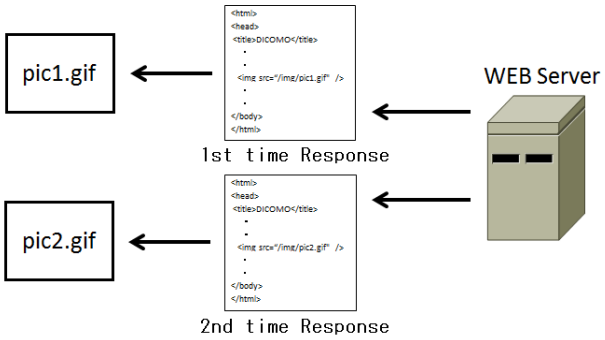

図 1 イメージタグの HTML とクローキングの例

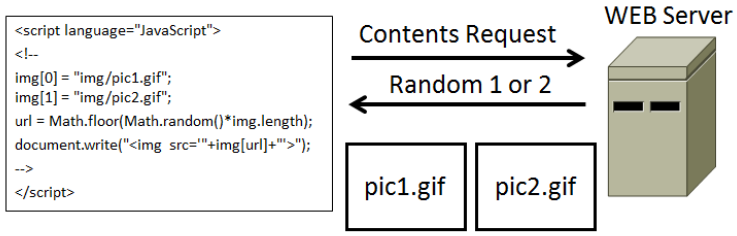

図 2 JavaScript 設置の HTML とクローキングの例

各画像フォーマットのヘッダ構造が異なるため，それぞれに対応する必要がある．主に Web で用いられる画像フォーマットの解析手法については，3.3 節で述べる．

（2）図 2 は，JavaScript がクライアントのブラウザ上で実行されることで異なるページが返される状況を示したコードである．通常，サーバ側で生成された実行コードの含まれていない HTML はクライアント側で変化することはない．

この場合では，JavaScript はブラウザで実行されることで初めて HTML コードに展開されるため，実行前の HTML の差異を比較しただけでは広告を検出することができない．しかし，コードが実行された際に広告画像を取得しに行くためには，外部とのアクセスを行う．検出時には 2 回同じコードを実行した上でそのレスポンスを監視し，サーバから返されてきたそれぞれのコンテンツのハッシュ値を比較することで検出を行うことができる．検出を行ったコンテンツは，ローカルに保存されている画像などと置き換えてからブラウザへレスポンスとして返す．この際，置き換える画像に関しては（1）の解析処理にて取得した幅と高さに変更する．これにより，ブラウザで表示されたときのレイアウト崩れを防ぎ，クローキングされた画像に対応することができる．

（3）図 3 は，Flash ファイルを画面に表示するためのタグを記述したものである．Flash は再生ファイルのみがクライアント側で実行され，提供される画像はサーバで生成される．そのため，再生の度に異なる広告が表示される場合がある．

これについても，（2）と同様にサーバから返されてきた各コンテンツのハッシュ値を比較することで検出を行うことができる．ただし，複数のコンテンツを取得しに行く場合は，「ハッシュリスト」にて対応を行う．片方の Flash ファイルが取得したコンテンツをハッシュリストに書き込んでおき，もう一方の Flash がそのリストにないハッシュ値のコンテンツを取得した場合は，その Flash はクローキングが行われたものとして対応する．

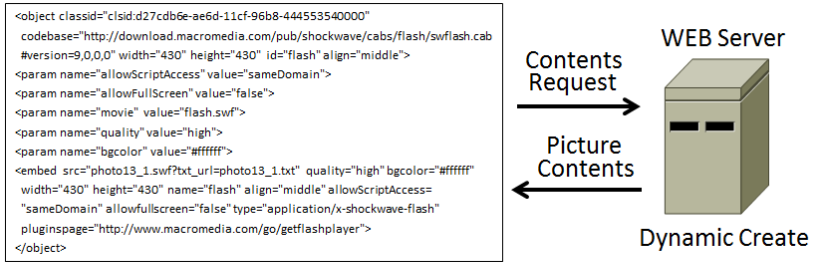

図 3 Flash 設置の HTML とクローキングの例

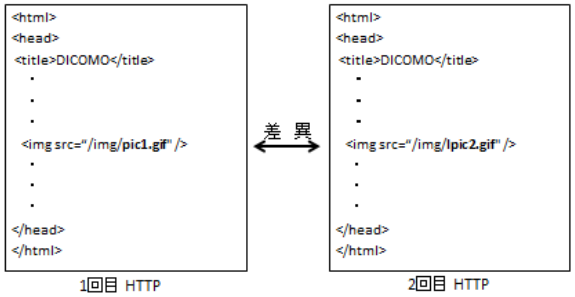

図 4 HTTP データの差異比較

3.2 クローキング広告の検出

（1）HTML 取得ステップ

ユーザがブラウザから Web ページの要求を行った際に，対象 Web サーバに対してクライアントからクエリ情報なども含めた全く同じリクエストヘッダを 2 回送信し，それぞれで取得した HTML データを保持しておく．この際，先に取得した HTML データをブラウザに表示させるものとして扱う．次に，取得したそれぞれの HTML に対して図 4 のようにデータ内容の比較を行い，差異を検出する．その差異の部分に画像，もしくは SWF コンテンツの取得先 URL が含まれていた場合，それはクローキングコンテンツであるものとして，対象リクエスト先のコンテンツを「解析処理対象リスト」に加える．解析処理対象リストとは，3.3 節で解析処理を行うコンテンツのリストである．

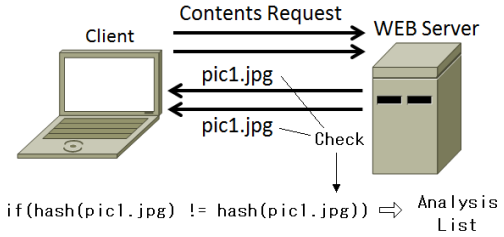

図 5 追加コンテンツのリクエストと比較

（2）コンテンツ取得ステップ

(1) のステップで取得したそれぞれの HTML データを読み込み, 図 5 のように Web サーバに対して追加のコンテンツのリクエストを行う．また，その際に取得したコンテンツデータのレスポンスヘッダを解析し，画像もしくは SWF であった場合に互いのハッシュ値を求め比較を行う．ハッシュ値が異なる場合は，クローキングコンテンツであるとして「解析処理対象リスト」に加える．

3.3 コンテンツの解析

ページレイアウトを崩さずにコンテンツを置き換えるためには, 置き換え元 Web ページのコンテンツの大きさを得る必要がある．そこで，3.2 節の検出で置き換えの対象となった「解析処理対象リスト」に含まれるコンテンツデータの種類によって，それぞれ以下のようにファイルのヘッダ情報をバイナリデータの単位で解析し，大きさの情報を取得する．ヘッダ情報を解析する対象のコンテンツは，3.2 節の（1）HTML 取得ステップで「最初に取得した HTML」からリクエストされたコンテンツである．

（1）BMP（Bitmap Image）形式

BMP 形式は，ヘッダの先頭 2byte が「0x42，0x4d」から始まっており，画像の幅や高さの情報は 14byte 以降の BITMAPINFO_HEADER というチャンクに格納されている．図 6(a)に示すように，それぞれ幅は 0x12～0x15 の 4byte，高さは 0x16～0x19 の 4byte を参照すると得ることができる．参照する際はリトルエンディアンとなる．

（2）PNG（Portable Network Graphics）形式

PNG 形式は，ヘッダの先頭 8byte が「0x89，0x50，0x4e，0x47，0x0d，0x0a，0x1a，0x0a」から始まっており，画像の幅や高さの情報は IHDR という 13byte で構成されたチャンクに格納されている. 図 6(b)に示すように, それぞれ幅は 0x10～0x13 の 4byte, 高さは 0x14～0x17 の 4byte を参照すると得ることができる．参照する際はビッグエンディアンとなる．

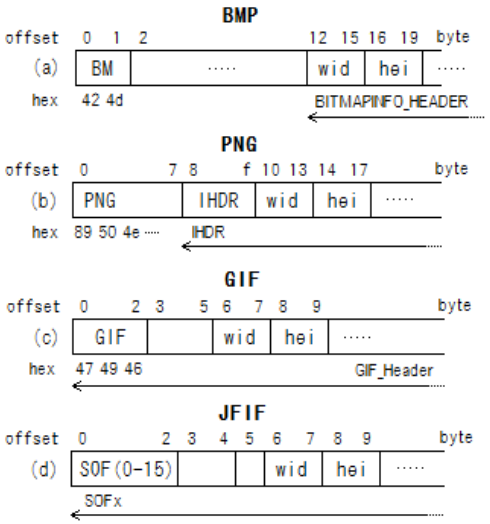

図 6 各画像形式における幅と高さの格納位置

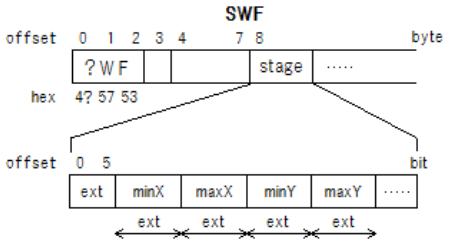

図 7 SWF 形式における幅と高さの格納位置

（3）GIF（Graphics Interchange Format）形式

GIF 形式は，ヘッダの先頭 6byte が「0x47，0x49，0x46，0x38，0x39（0x37），0x61」から始まっており, 画像の幅や高さの情報は先頭から始まる GIF_Header というチャンクに格納されている．図 6(c)に示すように，それぞれ幅は 0x06，0x07 の 2byte，高さは 0x08，0x09 の 2byte を参照すると得ることができる．参照する際はリトルエンディアンとなる．

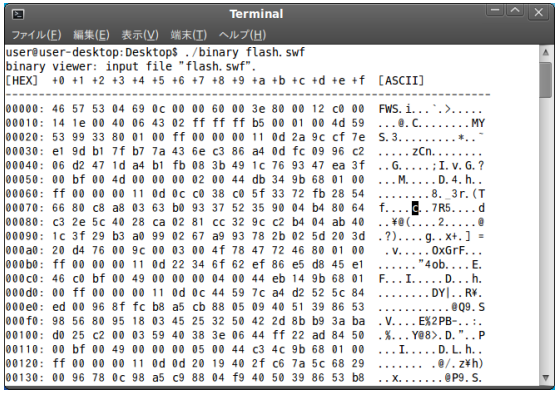

図 8 SWF 形式ファイルのバイナリデータ

（4）JPEG（Joint Photographic Experts Group）形式

JPEG 形式は，ヘッダの先頭 2byte が「0xff，0xd8」から始まっている．このファイル形式は規格名であり，データのフォーマットは JFIF（JPEG File Interchange Format）となっている．JFIF は画像情報や定義情報が含まれる各チャンクのことをセグメントと呼んでいるが，このセグメントは定まった順序を持たないため，他の画像フォーマットと比べて画像サイズなどの取得が難しいものとなっている．各セグメントの先頭を示すために，マーカーと呼ばれる 2byte のコードが用いられているため，始めにこの位置の検出を行う．フレームと呼ばれる画像情報が格納されている領域を示すマーカーコード（フレームヘッダ）は，符号化方式などのアルゴリズムの違いにより異なるが「0xffc0～0xffcf」の範囲となっており，JFIF には必ずこのうちのどれかが含まれている．フレームヘッダの位置を検出した後は，そのフレームのセグメントを読み取ることで画像情報の取得が可能となる．フレーム構造は図 6(d)のようになっており，それぞれ幅は 0x06，0x07 の 2byte，高さは 0x08，0x09 の 2byte を参照すると得ることができる．参照する際はビッグエンディアンとなる．

（5）SWF（Small Web Format）形式

SWF 形式は，ヘッダの先頭 3byte が「0x46（0x43），0x57，0x53」から始まっている．画像フォーマットとは異なり，Web で用いることを前提として可能な限り容量を小さく切り詰めるために，可変長の bit 単位で情報データが格納されている．このフォーマットのステージ（描画領域）の幅と高さが格納されている bit 幅は，0x08 の上位 5bit で判断を行うことができる．この値を 10 進数に変換した数の bit 幅が，幅と高さの情報を区切る基準となる．ただし，ヘッダが CWS の場合は 8byte 以降のデータが zlib 圧縮されているため，この限りではない．図 7 のように 0x08 の下位 3bit から後の領域を組み合わせ，先の 5bit で決められた bit 幅ずつ 4 つの領域に区切っていく．この 4 つの領域は，先頭から左上の X 座標，右下の X 座標，左上の Y 座標，右下の Y 座標という順番で表現されている．そのため，2 つ目と 4 つ目のパラメータを読み取ればステージの幅と高さを得ることができる．図 8 が実際の SWF ファイルのバイナリデータである．このファイルの場合，0x08 の上位 5bit が 0b01100 となることが分かる．つまり，以降のデータを 12bit ずつ 4 つに区切ることで，大きさを得ることができる．0x00，0x3e，0x80，0x00，0x12，0xc0，0x00 までをビット列に変換し，maxX，maxY の領域を求めると maxX が 2000，maxY が 600 になることが分かる．ただし，ここで得られるデータは Twips（Twentieth of an Inch Point）というウィンドウの設計をする際に用いられる単位であるため，注意が必要である．SWF では ActionScript などでモーションの動きを滑らかに表現するために，Pixel より細かい単位を定めている．20Pixel が 1Twips となるため，実際には取得したそれぞれの値の 20 分の 1 が幅と高さになる．よって，図 8 のファイルのステージの幅は 100Pixcel，高さは 30Pixcel となる．

3.4 クローキング広告の置き換え

図 9 に示すように，あらかじめユーザが登録しておいた画像リストの中から，「ヘッダ情報の解析によって得られたコンテンツ」の幅や高さに近い値に拡大縮小した画像を，3.2 節での HTTP リクエストに対するレスポンスとしてブラウザに返す．ブラウザはリクエストしたコンテンツが正常に返ってきたものとしてその画像を画面に表示するため，クローキングコンテンツを画像リストのものに置き換えることができる．画像リストからの選出は，表示領域の大きさに一番近いものを選択すると違和感がなく表示できる．置き換え対象コンテンツの幅と高さを解析してあるため，基本的にどのような画像であってもその大きさに自由に合わせることができる．

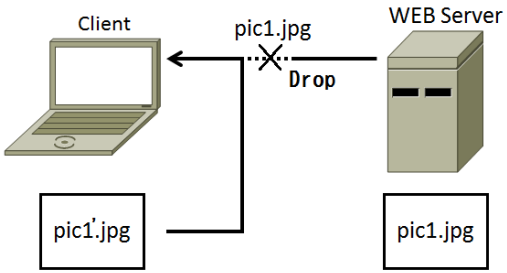

図 9 レスポンスコンテンツとの置き換え

4. おわりに

今日では，マルウェアが埋め込まれた悪意のあるクローキングサイトが多く存在している．このような Web サイトは，セキュリティサービスを提供するサーバからのアクセスと一般ユーザからのアクセスで異なるページを表示させるため，ホワイトリストやブラックリストなどを用いて対策することができない．そのため，このようなサービスを利用しても悪質なサイトを完全に検出することができず，マルウェアに感染する危険性があった．

そこで，Google インスタントプレビューや aguse Gateway といった既存の対策手法では，Web サイト側とユーザ側の中間にサーバを置き，ユーザが直接閲覧先のページにアクセスする状況を回避させることで，この問題を解決した．しかし，既存技術には検索エンジンに依存する点や, ページのレイアウトが崩れてしまう可能性のある点, プライバシーの問題点などが存在する．

本提案では，クローキングされたページのレスポンスのコンテンツに対して画像の大きさを解析し，あらかじめユーザが登録していた画像のリサイズを行った上でそれと置き換えることにより，ページレイアウトを崩すことなく信頼できるコンテンツへの置き換えを可能とした．さらに，これらの機能をブラウザのアドオンとして実装することで，検索エンジンの依存とプライバシーの問題を解決した．

Web サイト上で埋め込まれているマルウェアに対して本提案を実装することで，Script のレベルでは対策が可能である．一方，2.2 節で述べたような，内容は同一であるが環境によって振る舞いを変えるマルウェアであった場合，現在の段階では対応することができないなどの問題点がある．

今後の課題として，本提案を実装し評価実験を行ったうえで，多種多様なページに対してレイアウトを崩さずに置き換えができるか，また，様々な種類のマルウェアに対してどの程度の対応が可能であるかなどの検討を行っていく．

参考文献

[1] キッズ goo, < http://kids.goo.ne.jp/>(参照 2011-05-06).

[2] aguse Gateway, <http://gw.aguse.jp/>(参照 2011-04-26).

[3] Ryuya Uda, “Protocol and Method for Preventing Attacks from the Web”, World Academy of Science, Engineering and Technology 76 2011, pp.456-460, 2011.

[4] Mcafee:Do you know cloaking?, Mcafee Blog(online), <http://blogs.mcafee.com/mcafee-labs/do-you-know-cloaking>(参照 2011-04-10).

[5] Google インスタントプレビュー, <http://www.google.co.jp/landing/instantpreviews/>(参照 2011-04-26).

[6] U.Bayer, C.Kruegel, and E.Kirda, “TTAnalyze: A Tool for Analyzing Malware”, 15th Annual Conference of the European Institute for Computer Antivirus Research(EICAR), 2006.

[7] D.Inoue, K.Yoshioka, M.Eto, Y.Hoshizawa, and K.Nalao, “Automated Malware Analysis System and its Sandbox for Revealing Malware’s Intemal and Extemal Activities”, IEICE Trans. Vol.E92D, No.5, pp.945-954, 2009.

[8] 笠間 貴弘, 織井 達憲, 吉岡 克成, 松本 勉, “マルウェア動的解析オンラインサービスの脆弱性(その 2)”, 情報処理学会 コンピューターセキュリティシンポジウム(CSS)2010, pp.303-308, 2010.