情報処理学会論文誌 Vol.59 No.9 1624–1638 (Sep. 2018)

マルチ環境解析とJavaScript解析を組み合せた悪性Webサイトのクローキング分析手法

西尾 祐哉[1] 廣友 雅徳[1,a)] 神薗 雅紀[2] 福田 洋治[3] 毛利 公美[4] 白石 善明[5]

受付日 2017年12月12日, 採録日 2018年6月8日

概要：Drive-by Download 攻撃で利用される多くの悪性 Web サイトには，クライアントの環境を識別して挙動を変えるクローキングという技術が用いられている．そのため，単一の解析環境では Web サイトの悪性な挙動を見逃してしまうことがある．また，悪性 Web サイト内の JavaScript コードを難読化することで挙動を隠蔽している．本論文では，マルチ環境解析と JavaScript 解析を組み合せることで，悪性 Web サイトのクライアント側とサーバ側で行われるクローキングに関する情報を詳細に分析する手法を提案する．提案手法では，マルチ環境解析によって挙動を変える悪性 Web サイトを検出し，JavaScript 解析によってクライアント側のクローキングを明らかにする．また，マルチ環境解析と JavaScript 解析の結果に対して突合分析を行うことで，クローキングがクライアント側とサーバ側のどちらで実行されたかを識別する．実在する悪性 Web サイトを提案手法で分析することで，サーバ側とクライアント側のクローキングを行う条件とその挙動をとらえることができ，攻撃対象となる環境を導出できることが確認できた．

キーワード：Drive-by Download 攻撃，クローキング，マルチ環境解析，JavaScript 解析

An Analysis Method for Cloaking Malicious Websites by Combining Multi-Environment Analysis and JavaScript Code Analysis

Yuya Nishio[1] Masanori Hirotomo[1,a)] Masaki Kamizono[2] Youji Fukuta[3]
Masami Mohri[4] Yoshiaki Shiraishi[5]

Received: December 12, 2017, Accepted: June 8, 2018

Abstract: The malicious websites used by drive-by download attacks change their behavior for web client environments. It is called cloaking. To analyze the behavior of malicious websites, the single-environment analysis cannot obtain sufficient information. Also, the JavaScript code obfuscation is used in malicious websites in order to avoid to be analyzed their behavior. In this paper, we propose an analysis method for cloaking malicious websites by combining multi-environment analysis and JavaScript code analysis. In the proposed method, we detect the change of behavior of malicious websites by multi-enviroment analysis, and identify cloaking techniques executing on client environments by JavaScript code analysis. Furthermore, we report case studies that the proposed method is applied to real malicious websites and identifies cloaking techniques executing on the client side and server side.

Keywords: drive-by download attack, cloaking, multi-environment analysis, JavaScript analysis

1. まえがき

近年，クライアント PC を狙った攻撃が世界中で発生しており，特に PC 内のファイルを暗号化し，復号させるために金銭を要求するランサムウェアが猛威を振るっている．トレンドマイクロが公開しているレポート [1] による

[1] 佐賀大学大学院工学系研究科
Saga University, Saga 840–8502, Japan
[2] PwC サイバーサービス合同会社
PwC Cyber Services LLC., Chiyoda, Tokyo 100–0004, Japan
[3] 近畿大学理工学部情報学科
Kindai University, Higashiosaka, Osaka 577–8502, Japan
[4] 岐阜大学工学部電気電子・情報工学科
Gifu University, Gifu 501–1193, Japan
[5] 神戸大学大学院工学研究科
Kobe University, Kobe, Hyogo 657–8501, Japan

[a)] hirotomo@cc.saga-u.ac.jp

と，2016 年の国内でのランサムウェア検出台数は約 65,400 件であった．ランサムウェアなどのマルウェアをクライアント PC に感染させるために，Web サイトを閲覧したユーザに強制的にマルウェアをダウンロードさせる Drive-by Download（DBD）攻撃が主に用いられている．

DBD 攻撃の対策として，ユーザの悪性 Web サイトへのアクセスを防ぐために，Microsoft や Google が提供している URL ブラックリストを用いたアクセスブロック機能が存在している [2], [3]．しかし，正規サイトが改ざんされて悪性 Web サイトへのリダイレクトコードが埋め込まれることや，悪性 Web サイトの多くはその URL を短期間で遷移させていることから，ブラックリストだけでは防ぎきれない．また，aguse [4] や urlQuery.net [5] など，ユーザの代わりに Web サイトにアクセスし，Web サイトのキャプチャ画像や解析結果を提供してくれる Web サイト解析サービスがある．しかし，これらのサービスは複数の解析結果を表示していないことから，単一の環境で解析していると推測される．悪性 Web サイトは端末の環境を識別して挙動を変えるクローキングという技術を利用しているため，これらの解析サービスだけでは十分な情報を得ることができず，悪性な挙動を見逃してしまう可能性がある．

本論文では，マルチ環境解析と JavaScript 解析を組み合せることで，悪性 Web サイトのクライアント側とサーバ側で行われるクローキングに関する情報を詳細に分析する手法を提案する．クローキングはクライアント側とサーバ側で行われるものに分けられ，クライアント側は主に JavaScript，サーバ側は主に PHP によって実行される．マルチ環境解析だけの場合，クローキングを検知できるが，それがクライアント側とサーバ側のどちらで実行されたか判別できない．また，JavaScript 解析だけの場合，クライアント側のクローキングに関する情報を取得できるが，サーバ側のクローキングに関する情報は取得できない．そこで提案手法では，悪性 Web サイトに対してマルチ環境解析と JavaScript 解析を行い，さらに双方の解析結果に対して突合分析を行うことで，クライアント側とサーバ側のクローキングに関する情報を取得する．提案手法によって，クローキングがクライアント側とサーバ側のどちらで実行されたかを識別することで，悪性 Web サイトの挙動を分析する際に，解析妨害技術に対応した環境を構築できるようになり，より詳細な分析結果を得ることが可能となる．

本論文の構成は次のとおりである．2 章では，Drive-by Download 攻撃と，悪性 Web サイトで行われるクローキングについて述べる．3 章では，マルチ環境解析と JavaScript 解析を組み合せた分析手法を提案する．4 章では，提案手法で悪性 Web サイトのクローキングを分析した結果を述べる．5 章では，関連研究について述べる．6 章では，本論文のまとめを述べる．

2. 悪性 Web サイトのクローキング

2.1 Drive-by Download 攻撃

Drive-by Download（DBD）攻撃とは，ユーザが悪性な Web サイトにアクセスすると，複数回のリダイレクトを経てマルウェア配布サイトへ誘導され，ブラウザやプラグインの脆弱性が悪用されて強制的にマルウェアがダウンロードされる攻撃である．従来のインターネットでの攻撃手法は，攻撃者が攻撃対象者に悪意のある情報を送る能動的攻撃であった．それに対して DBD 攻撃は，ユーザの Web サイトへのアクセスを攻撃の起点とし，攻撃者が攻撃対象者からの要求を受けて悪意のある情報を返答するため，受動的攻撃に分類される．**図 1** では DBD 攻撃の流れを示している．攻撃に関与する Web サイトは，入口サイト，中継サイト，攻撃サイト，マルウェア配布サイトの 4 つからなり，それぞれの役割は次のようになる．

［入口サイト］
攻撃の起点となるサイトである．アクセスすると，中継サイトへリダイレクトされる．攻撃者が作成したサイトとは限らず，正規サイトを改ざんして入口サイトとすることが多い．

［中継サイト］
攻撃サイトへの中継を行うサイトである．複数の中継サイトを経由する場合もある．

［攻撃サイト］
クライアント PC の OS や Web ブラウザ，Web ブラウザ上で動作するプラグインの脆弱性を悪用し，マルウェア配布サイトからマルウェアをダウンロードさせるスクリプトを実行させる．

［マルウェア配布サイト］
攻撃サイトによって乗っ取られたクライアント PC のリクエストに応えてマルウェアを送信し，強制的に実行させることでマルウェアに感染させる．

DBD 攻撃は上記の複数の Web サイトが連動して行われるが，ダウンロード画面やインストール画面，リダイレクト時の画面は表示されないため，ユーザは攻撃を受けたこ

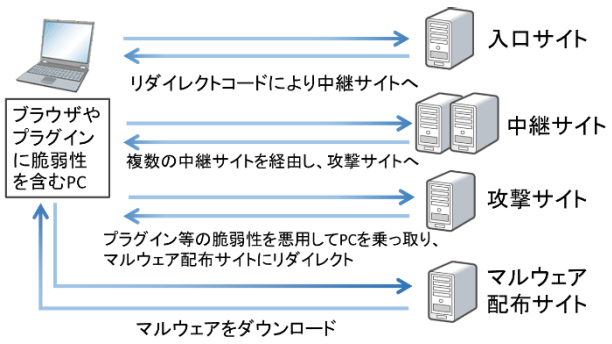

図 1 Drive-by Download 攻撃の流れ
Fig. 1 Flow of Drive-by download attack.

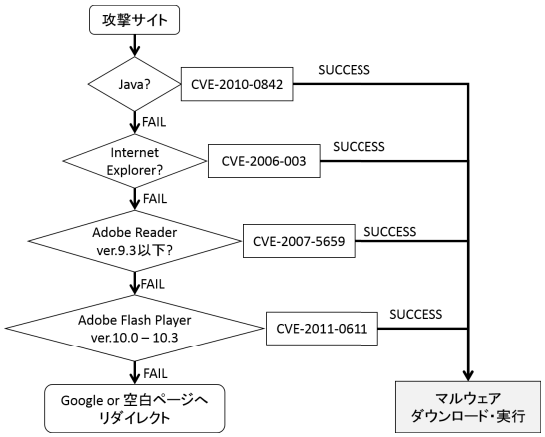

図 2 Blackhole Exploit Kit によるクローキングの一例
Fig. 2 An example of cloaking by Blackhole Exploit Kit.

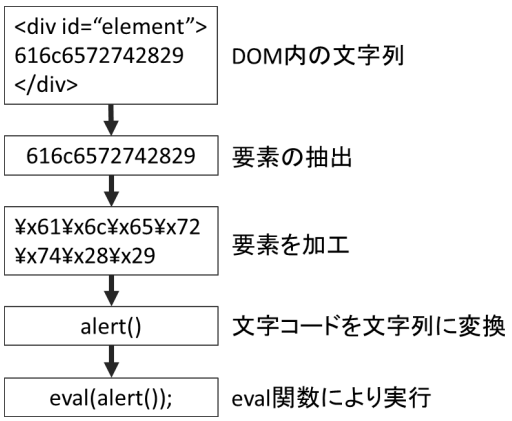

図 3 難読化 JavaScript 実行の例
Fig. 3 An example of obfuscated JavaScript.

とに気付くことができない．入口サイトは，正規サイトを改ざんして iframe を挿入し，外部の悪性サイトにアクセスさせていることが多い．この iframe のフレームサイズを小さく設定することで，表示されている悪性サイトを視覚的に見えなくしている．

2.2 クローキングによる解析妨害

クローキングとは，Web サイトにアクセスしてきたユーザ環境を識別し，環境によって応答を変える技術である．Blackhole Exploit Kit [6] のクローキングの一部を**図 2** に示す．このように，環境によって悪用する脆弱性を変化させ，悪用できる脆弱性が存在しなければ攻撃を実行しない．クローキングは解析端末による検知から逃れる目的でも使用されており，攻撃条件に一致した環境にだけ攻撃を実行する．クローキングはクライアント側とサーバ側で実行するものに分けられ，それぞれ次のような処理を行う．

［クライアント側のクローキング］

主に JavaScript によって実行される．ブラウザとプラグインの種類やバージョン，OS などの情報を取得し，ユーザ環境を識別する．ブラウザ情報は主に User-Agent の値で判断しているが，ブラウザ特有機能の可否によって，正確にブラウザの種類とバージョンを識別する場合もある．JavaScript コードは難読化されている場合が多い．

［サーバ側のクローキング］

主に PHP によって実行される．攻撃者は独自の IP アドレスのブラックリストを保有しており，セキュリティ企業などの IP アドレスによるアクセスをブロックする．また，初回アクセスの端末の IP アドレスを記録し，2 回目以降のアクセスをブロックする場合もある．入口サイトや中継サイトを経由せずに直接攻撃サイトへアクセスすることを防ぐために，HTTP リファラの値を利用してリダイレクト経路を確認する処理を行う．アクセス元の国を識別する際には，IP アドレスや Accept-Language を利用する．

クライアント側のクローキングは悪性 Web サイト内の JavaScript コードを解析することで明らかにできる．しかし，PHP コードは基本的に外部から取得することができないため，PHP コードを直接解析してサーバ側のクローキングに関する情報を取得することはできない．

2.3 クローキング以外の解析妨害

悪性 Web サイトでは，クローキング以外に次のような解析妨害手法が利用されている．

［難読化 JavaScript］

悪性 Web サイト内の JavaScript コードは，処理内容の秘匿化やアンチウイルスソフトなどの検知から逃れるために，難読化が施されている．**図 3** は難読化コードを実行する流れの一例である．DOM から要素を抽出し，Unicode のエンコード方式で文字列に変換した後に，eval 関数を用いて，文字列をコードとして実行する．生成されたコードは，実際に悪性な処理を実行する場合もあれば，さらに新しいコードを生成するなど，多重に難読化が施されている場合もある．

［短時間で変化する URL］

悪性 Web サイトの URL は数分単位で変化することが多い．この場合，Web サイト間にゲートを使用することで，URL の変化に対応したリダイレクトを実現させている．ゲートとは，Web サイト間のリダイレクトを仲介させるもので，Pseudo-Darkleech や EITest などがあげられる．URL を短時間で変化させることによって，解析者が直接アクセスすることを難しくし，さらに URL ブラックリストによる検知からも逃れている．入口サイトが改ざんサイトの場合は URL が変わらない．そのため，解析の際には入口サイトからアクセスする必要がある．

3. クローキングの分析手法

2.2 節で述べたように，クローキングはクライアント側とサーバ側で実行されるものに分けられ，それぞれ主に

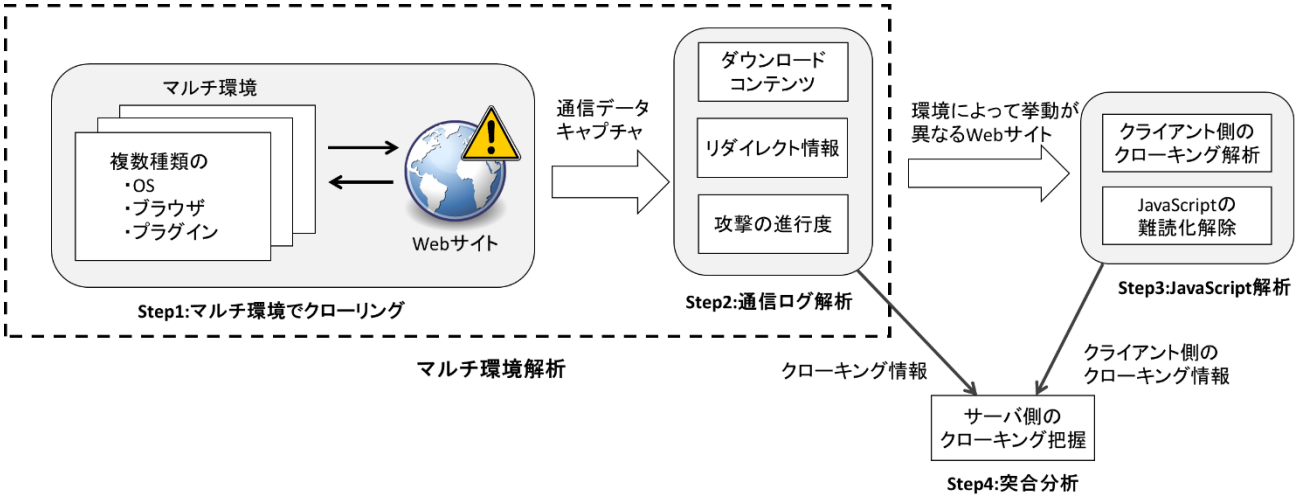

図 4 マルチ環境解析と JavaScript 解析を用いたクローキング分析の流れ
Fig. 4 Flow of cloaking analysis by multi-environment analysis and JavaScript analysis.

表 1 各解析手法で取得可能な情報
Table 1 Information that can be obtained by each analysis method.

	クライアント側の クローキングの有無	サーバ側の クローキングの有無	難読化の有無	短時間で変化する URL の有無
マルチ環境解析	検知可能だがサーバ側とクライアント側の どちらか判別できない※1		検知不可能	検知可能
JS 解析	検知可能※2	検知不可能	検知可能	検知不可能
提案手法	検知可能※2	検知可能※1	検知可能	検知可能

※1 誤検知と見逃しが発生する場合もある．
※2 見逃しが発生する場合もある．

JavaScript と PHP で構築されている．JavaScript コードを解析することでクライアント側のクローキング情報を取得できるが，PHP コードは基本的に取得することができないため，PHP コードを解析してサーバ側のクローキング情報を得ることはできない．そこで本論文では，クライアント側のクローキングに関する情報を取得することに加え，マルチ環境解析を利用してサーバ側のクローキングに関する情報を取得する手法を提案する．

提案手法によるクローキング分析の流れを図 4 に示す．Step1 と Step2 はマルチ環境解析の処理である．また，各 Step における処理は次のとおりである．

Step1：様々な種類の OS やブラウザ，プラグインを導入したマルチ環境で悪性 Web サイトをクローリングする．

Step2：通信ログ解析によって，ダウンロードコンテンツとリダイレクト情報を取得し，環境によって挙動を変える悪性 Web サイトを検出する．そして，環境ごとの攻撃の進行度を判定し，同じ挙動を示した環境の共通点から，挙動を変える条件とその挙動内容などのクローキング情報を推測する．また，環境によってダウンロードコンテンツなどが異なる Web サイトをクローキングサイトとして検出する．

Step3：クローキングサイトに関連する Web サイト内の JavaScript コードを解析し，主に環境情報を利用する条件分岐を探す．その分岐の条件を特定し，条件に応じた処理内容を理解することで，クライアント側のクローキングに関する情報を取得する．そして，Step3 で検知したクローキングはクライアント側で実行されたものと判断する．

Step4：マルチ環境解析と JavaScript 解析の結果に対して突合分析を行う．マルチ環境解析で判明したクローキングのうち，JavaScript 解析で判明したクローキングと一致しないものを，サーバ側のクローキングと判定する．

各解析手法で取得可能な情報を表 1 に示す．マルチ環境解析だけの場合，同じ挙動を示した環境の共通点に着目することでクローキングを検知できるが，それがクライアント側とサーバ側のどちらで実行されたかは判別できない．また，クローキングの条件に沿った挙動とは異なる挙動を

偶発的に示した場合は誤検知を起こす可能性があり，用意したマルチ環境がすべてのクローキングのパターンに対応できていない場合は挙動を見逃す可能性がある．ソースコードを解析しないマルチ環境解析では難読化 JavaScript を検知しないが，通信ログ解析によって短時間で変化する URL を検知できる．一方，JavaScript 解析だけの場合，クライアント側のクローキングに関する情報を取得できるが，サーバ側のクローキングに関する情報は取得できない．また，JavaScript コードにクローキング機能を含む難読化が施されている場合，難読化解除時に特定の環境条件でのみ生成されるコードを網羅的に捕捉しなければ挙動を見逃す可能性がある．URL に関しては，JavaScript によるリダイレクトを利用していれば URL を検出できるが，それが短時間で変化する URL であるか判定はできない．それに対して提案手法では，JavaScript 解析によってクライアント側のクローキングに関する情報を取得し，さらに JavaScript 解析とマルチ環境解析の結果に対して突合分析を行うことで，サーバ側のクローキングに関する情報を取得する．JavaScript 解析とマルチ環境解析の双方の結果を分析する提案手法では，難読化は JavaScript 解析によって，短時間で変化する URL はマルチ環境解析によって，検知できる．さらに，リダイレクト手法やリダイレクトに用いられたツールを特定できるなど，より詳細に情報を得ることが可能となる．なお，マルチ環境解析と JavaScript 解析を単体で行う場合と同様に，誤検知や見逃しが発生する場合はある．

3.1 マルチ環境によるクローリング（Step1）

3.1.1 マルチ環境解析の構成

本研究では，挙動を変える悪性 Web サイトの影響を評価するためのマルチ環境解析システムを実装した．マルチ環境解析システムの基本的な構成は，仮想端末上にそれぞれ種類やバージョンの異なるブラウザやプラグインを導入し，複数環境での解析を実現させたクライアント型ハニーポットである．このシステムでは実際のソフトウェアに存在する脆弱性を悪用させ，感染するマルウェアまでとらえることを目的としているため，高対話型ハニーポットに分類される．ゲスト OS 上で用いたソフトウェアは分析事例ごとに異なるため，4 章の各事例の冒頭で述べる．

3.1.2 クローラと IP アドレス

マルチ環境で解析対象の Web サイトにアクセスする処理は，クローラによって自動的に行う．本論文では Web アプリケーションのテストツールである Selenium Webdriver [7] を用いることで，Web ブラウザの操作とクローリングを自動化させた．また，クローリング中は Wireshark [8] によって通信データをキャプチャする．

2.2 節のサーバ側のクローキングで述べたように，同じ IP アドレスで悪性 Web サイトに複数回アクセスするとブロックされる可能性が高い．その対策として，再起動する度に IP アドレスが変わる市販のモバイル Wi-Fi ルータを使用してインターネットに接続した．各環境でアクセスする度にルータを再起動して IP アドレスを変えることにより，IP 検知を回避させる．そのため，本論文のマルチ環境解析では，全環境で同時に Web サイトへアクセスせず，1 つの環境で順次アクセスしていく．

表 2 検出する Content-Type の例
Table 2 Examples of Content-Type to be detected.

ファイル	Content-Type
PDF	application/pdf
JAR	application/java-archiver
SWF	application/x-shockwave-flash
XAP	application/x-silverlight-2
BIN	application/octet-stream
	application/x-download
	application/x-msdos-program
	application/x-msdownload

3.2 通信ログ解析（Step2）

通信ログ解析では，Step1 で通信をキャプチャした pcap ファイルを解析し，環境によって挙動を変える悪性 Web サイトを検出する．また，各環境に対する攻撃の進行度を判定する．本論文では基本的に手動で解析や良性・悪性の判断，攻撃進行度の判別を行うが，ダウンロードコンテンツの抽出処理とリダイレクト検出は自動化した．

3.2.1 ダウンロードコンテンツ

悪性 Web サイトで脆弱性を悪用する攻撃が行われる際には SWF ファイルや PDF ファイルがダウンロードされ，その後マルウェア本体である実行ファイルがダウンロードされる．通信ログ解析ではまず，HTTP レスポンス内の Content-Type の値を見て特定のファイルをダウンロードしたパケットを検出する．検出対象とする Content-Type の例を**表 2** に示す．また，コンテンツの悪性判定をオンライン解析サービスの VirusTotal [9] で行い，悪性の可能性があるコンテンツだけを検出する．検出されたコンテンツのダウンロード状況が環境によって異なる場合，そのダウンロード元 URL に対して以降の解析を進める．

3.2.2 リダイレクト情報

図 5 で示すように，3.2.1 項で検出した悪性コンテンツのダウンロード元 URL を起点とし，HTTP リファラの値を利用してリダイレクトの流れを追跡する．たとえば，中継サイトの URL は，攻撃サイトの HTTP リクエストのリファラヘッダに記述されている．リダイレクトに関連する Web サイトの分類方法は次のとおりである．

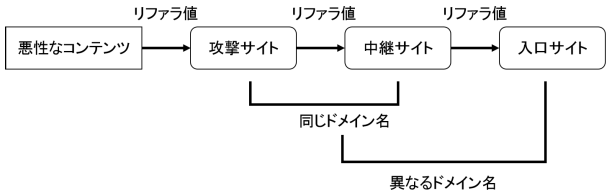

図 5 リダイレクト解析の流れ
Fig. 5 Flow of redirect analysis.

[攻撃サイト]
3.2.1 項でとらえた悪性なコンテンツのダウンロード元 Web サイトを攻撃サイトに分類する．

[中継サイト]
攻撃サイトのリダイレクト元 Web サイトを中継サイトに分類する．中継サイトは攻撃サイトのドメイン名と同じ場合が多く，悪性なコンテンツはダウンロードしない．また，中継サイトは存在しない場合もあれば，複数存在する場合もある．

[入口サイト]
中継サイトと攻撃サイトのリダイレクト元であり，3.1 節のマルチ環境でクローリング対象とした Web サイトを入口サイトに分類する．正規サイトを改ざんして入口サイトとした場合，入口サイトのドメイン名と中継・攻撃サイトのドメイン名は異なる．

3.2.3 攻撃の進行度

3.2.1 項と 3.2.2 項の結果をもとに，各解析環境に対する攻撃の進行度を次の 4 段階のレベルで判定する．

レベル 1：攻撃不発生
入口サイトからリダイレクトが発生していない．主に User-Agent を見て，その環境は攻撃対象ではないか，解析端末だと判定された．

レベル 2：悪性リダイレクトのみ
中継サイトか攻撃サイトまで到達したが，悪性なコンテンツをダウンロードせず，攻撃は中止された．レベル 1 よりも詳細に環境を調査したうえで，その環境は攻撃対象ではないと判定された．

レベル 3：脆弱性を狙った攻撃
攻撃サイトまで到達し，SWF ファイルなどをダウンロードして脆弱性を悪用するコードを実行したが，攻撃に失敗した．攻撃条件は満たしているが，その環境に悪用できる脆弱性が存在していなかった．

レベル 4：マルウェア感染
脆弱性を悪用した攻撃に成功し，悪性な実行ファイルのダウンロード・実行まで至った．その環境は攻撃対象となる条件をすべて満たし，悪用できる脆弱性も存在していた．

3.3 悪性 JavaScript の解析（Step3）

3.2 節の通信ログ解析で検出した悪性 Web サイト内の JavaScript コードを解析し，クライアント側のクローキング処理を明らかにする．本論文では，JavaScript コードの解析と難読化解除は基本的に手動で行った．

表 3 環境情報を取得するプロパティの例
Table 3 Examples of property to get environmental information.

取得情報	使用プロパティ
OS	navigator.platform
ブラウザ	navigator.userAgent
プラグイン	navigator.plugins
言語	navigator.language

```
var ua = navigator.userAgent;
var bName = function () {
    if (ua.search(/Edge/) > -1) return "edge";
    if ((ua.search(/MSIE/) > -1) || (ua.match(/Trident/) > -1))
       return "ie";
    if (ua.search(/Firefox/) > -1) return "firefox";
    if ((ua.search(/Opera/) > -1) || (ua.search(/OPR/) > -1))
       return "opera";
    if (ua.search(/YaBrowser/) > -1) return "yabrowser";
    if (ua.search(/Chrome/) > -1) return "chrome";
    if (ua.search(/Safari/) > -1) return "safari";
    if (ua.search(/Maxthon/) > -1) return "maxthon";
    else return "unknown";
}();
```

図 6 JavaScript によるブラウザの判別
Fig. 6 Identification of web browser by JavaScript.

3.3.1 クライアント側のクローキング解析

JavaScript 解析では，まず環境情報を取得しているコードを特定する．基本的には**表 3** で示すような navigator オブジェクトを利用している箇所を探す．しかし，ブラウザやプラグインの特有機能を確認することで正確に判別する場合もあり，たとえば IE は ActiveXObject の有無を確認することで判別できる．そのため，ブラウザやプラグインなどの特定のソフトウェアでしか利用できないオブジェクトの有無も確認する．

次に，環境情報を利用した条件分岐処理を探す．具体的には if 文や switch 文などの条件文に，navigator オブジェクトなどから取得した値を利用している箇所を特定する．その条件文の内容を理解することで，挙動を変える条件を特定できる．また，条件に応じた処理を追跡することで，各環境条件による挙動内容をとらえることができる．JavaScript によるクローキングの例を**図 6** に示す．図 6 では User-Agent の値によってブラウザを判別している．このコードが実行された後は，取得した情報によって処理内容を変化させている．JavaScript 解析では，このような環境情報を利用した条件分岐を探すことで，クライアント側のクローキング処理を明らかにする．

3.3.2 JavaScript の難読化解除

悪性 Web サイト内の JavaScript は難読化されている場合が多いため，クローキングを解析するには難読化を解除する必要がある．ただし，クローキングによって難読化コードから生成されるコードが変化する場合があるた

め，自動で難読化を解除することは難しい．本論文ではChrome のデベロッパツールを利用し，手動でコードをステップ実行することで難読化を解除する．クローキングが組み込まれた難読化コードに対しては，環境情報を入れる変数を直接編集することで対応する．

3.4 突合分析（Step4）

マルチ環境解析と JavaScript 解析の結果を突合分析することで，サーバ側のクローキングに関する情報を推定する．まずクライアント側のクローキングは，3.3 節の JavaScript 解析で明らかになっている．具体的には，環境情報を利用した条件分岐がクローキングの処理になる．

次にサーバ側のクローキングは，マルチ環境解析と JavaScript 解析の結果を突合させて考える．マルチ環境解析では，同じ挙動を示した環境の共通点から環境条件を特定し，環境ごとのダウンロードコンテンツやリダイレクト状況を確認することで，クローキングに関する情報を取得している．そのマルチ環境解析で判明したクローキングが，JavaScript 解析で判明したクローキングと一致しない場合，それはサーバ側のクローキングであると判断できる．

4. ケーススタディ

提案手法で実在する悪性 Web サイトのクローキングを分析した結果を述べる．

4.1 分析事例 1

公開ブラックリストに掲載されている悪性 Web サイトに対して，プラグインの異なる環境を準備した解析環境を用い，マルチ環境解析を行った．

4.1.1 実験用 URL

分析事例 1 では，Malware Domain List [10] に 2016 年 2 月 29 日から 2016 年 3 月 29 の期間に掲載された入口サイトの疑いのある Web サイト 29 個の URL を使用した．

4.1.2 事例 1 の実験環境

解析環境に導入したソフトウェアを**表 4** に示し，使用プラグインを**表 5** に示す．本実験では，表 5 の各プラグインのバージョンの組合せごとに，合計 27 個の解析環境を構築した．ゲスト OS は 2015 年 12 月 25 日時点でシェア率が最も高い Windows7 を利用した．Internet Explorer（IE）8.0 は OS に適応できる最古のバージョンである．プラグインの種類は，過去に脆弱性が悪用されたことの多い Java Runtime Environment（JRE），Adobe Reader，Adobe Flash Player を用意した [11], [12]．Java と Adobe Reader は OS に適応できる最古のバージョンから 3 つを使用している．Adobe Flash Player に関しては，バージョン 10 が OS に適応できる最古のもの，バージョン 20 は 2015 年 12 月 25 日時点で最新のもの，バージョン 14 は 2015 年 12 月 25 日時点で最も脆弱性の数が多いものとなっている．各ソフトウェアのシェア率は NetMarketShare [13]，脆弱性の数は CVE Details [14] で公開されている情報を参考にした．なお，全環境で Windows アップデートは適用していない．

実験は 2016 年 3 月 29 日から 31 日までの 3 日間にかけて実施した．

4.1.3 事例 1 の分析結果

マルチ環境解析を行ったところ，実験に使用した 29 個の URL のうち，edden****.com, schuh****.de, saint****.fr, market****.com，ine****.co.il の計 5 個の Web サイトで発生した悪性なリダイレクトを検知し，この 5 個の入口サイトから同じ攻撃サイトへリダイレクトしていることが分かった．しかし，いずれの環境もマルウェアには感染しておらず，悪性なファイルをダウンロードした痕跡もなかった．さらに，全環境で特に挙動の違いを確認できなかったため，マルチ環境解析だけではクローキング情報を得られなかった．

次に入口サイトと攻撃サイトの JavaScript を解析した．JavaScript 解析で得たクローキング情報は次のとおりである．

- 入口サイトでは，ブラウザが IE 9 以前であれば iframe が生成され，攻撃サイトへリダイレクトする．ブラウザが Firefox や IE 10 か IE 11 であれば，そこで JavaScript の処理を停止し，リダイレクトが発生しない．
- 攻撃サイトでは，ユーザ環境に特定の仮想化ソフト，アンチウイルスソフト，Web デバッキングツールが存在しなければ，Adobe Flash Player か Silverlight の脆弱性を突く攻撃を行う．

入口サイトの JavaScript は難読化されており，その難読化手法から Pseudo-Darkleech によって Web サイトを改ざんしていることが分かった．ブラウザの判別は，“window.sidebar” という Mozzila 製品固有のオブジェクトが利用できるかを確認するコードが 1 件あり，Firefox の識別

表 4 事例 1 で使用したソフトウェア
Table 4 Software used in case 1.

仮想マシン	VMware Workstation 12
ホスト OS	Windows 7 Professional
ゲスト OS	Windows 7 Professional
ブラウザ	Internet Explorer 8.0

表 5 事例 1 で使用したプラグイン
Table 5 Plug-in used in case 1.

プラグイン	バージョン		
Java Runtime Environment	1.5	1.6	1.7
Adobe Reader	9.0	10.0	11.0
Adobe Flash Player	10	14	20

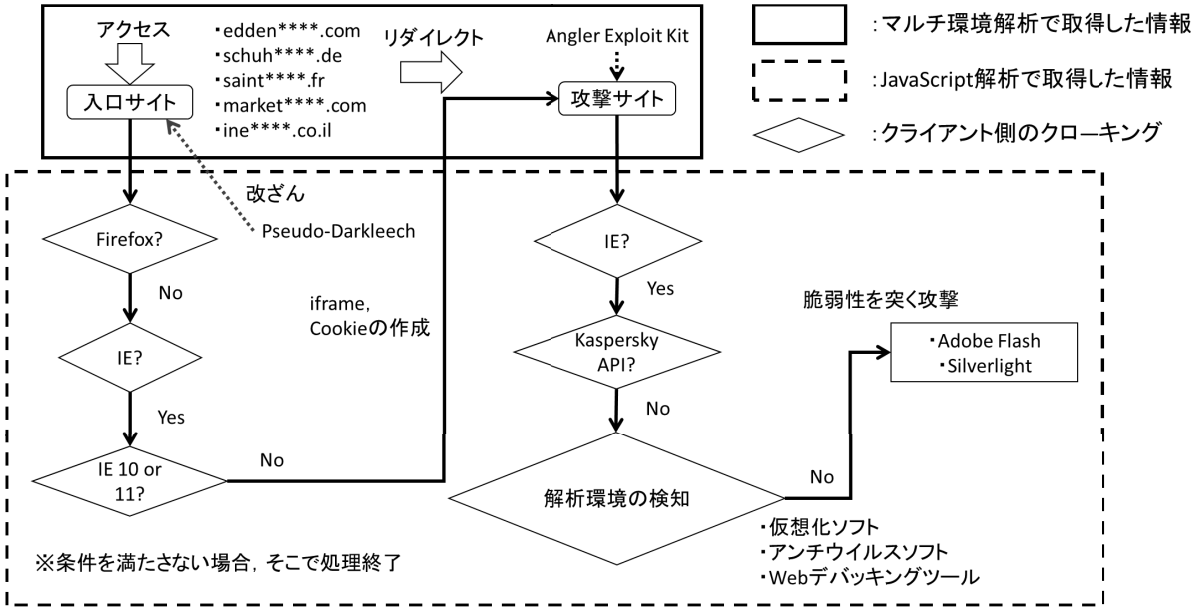

図 7 事例 1 の分析結果
Fig. 7 Analysis result of case 1.

を行っていた．また，IE に関しては User-Agent の値で識別していた．難読化コードにはクローキング機能が組み込まれており，User-Agent の値が IE9 以下を示していれば，攻撃サイトへリダイレクトさせる iframe を含むコードを生成し，自身のソースコードに新しく追加して実行する．User-Agent の値がそれ以外であれば，意味を持たない文字列を生成して自身のソースコードに追加するだけで，リダイレクトは発生しない．クローキング機能が組み込まれた難読化コードはこの 1 件のみである．iframe を作成する際に Cookie も生成しており，これは攻撃サイトへアクセスする際の許可証のようなものである．そのため，特定の Cookie がない環境では，攻撃サイトへアクセスできないと考えられる．

攻撃サイトの JavaScript は多重に難読化が施されていた．攻撃サイトでは，User-Agent の値で IE を識別していた．その後，カスペルスキー製品や Malwarebytes Anti-Malware などのアンチウイルスソフト，VMware や VirtualBox などの仮想化ソフト，さらに Web アプリケーションデバッグ用ツールの Fiddler2 が端末に存在するか調査していた．JavaScript で画像を読み込む際に “res://C:¥Program Files¥Fiddler2¥Fiddler.exe” のように，特定のファイルパスを指定することで検知しており，この検知コードを 82 件確認した．本実験で用意した全環境は，この解析端末の検知で VMware を検知されたため，脆弱性を悪用する攻撃が実行されなかったと考えられる．解析端末の検知手法から，攻撃サイトは Angler Exploit Kit によって構築されたことが分かった．

分析結果のまとめを**図 7** に示し，提案手法で取得した情報を**表 6** に示す．本事例ではマルチ環境解析によってクローキング情報を取得できなかったため，サーバ側のクローキングを推定できなかったが，JavaScript 解析によってクライアント側のクローキング情報を取得することができた．この悪性 Web サイト全体の挙動から，IE 9 以下のブラウザを使用し，特定のアンチウイルスソフトや仮想化ソフトを使用せず，特定の脆弱性が存在する Adobe Flash Player か Silverlight が存在する環境であれば，マルウェアに感染すると判断できる．

4.2 分析事例 2

分析事例 2 では，マルチモーダル分析によって取得した悪性 Web サイトの URL に対して，プラグインだけでなく OS やブラウザの環境を変更した解析環境を準備し，マルチ環境解析を行った．

4.2.1 実験用 URL

分析事例 1 では公開ブラックリストから実験用 URL を取得した．しかし，2.2 節のサーバ側のクローキングと 2.3 節の短時間で変化する URL で述べたように，掲載されている URL が中継サイトや攻撃サイトのものであれば，直接アクセスしても悪性な挙動を示すことはない．そのため，入口サイトからアクセスする必要があるが，入口サイトがブラックリストに登録された頃には，悪性 Web サイトへのリダイレクトが機能していない場合もある．そこで本実験では，Exploit Kit の解析レポートに記載されている URL を起点としたマルチモーダル分析 [15], [16] によって入口サイトの URL を取得した．

マルチモーダル分析では，Exploit Kit の解析レポートから入口サイトの IP アドレスを取得した後，その値を VirusTotal [9] で検索し，検索結果で表示される関連 URL

表 6 事例 1 で取得した情報
Table 6 Information obtained in case 1.

	クライアント側の各種クローキングの有無	サーバ側の各種クローキングの有無	難読化の有無	短時間で変化する URL の有無
マルチ環境解析	・クローキングを検知できず		検知不可能	・攻撃サイトの URL が短時間で変化 ・URL の特徴から Angler EK を使用
JS 解析	・入口サイトでブラウザを識別し，IE9 以下の場合は攻撃サイトへリダイレクトし，同時に Cookie を作成する． ・攻撃サイトでブラウザを識別後，仮想化ソフトやアンチウイルスソフト等を確認し，解析環境と判断されなければ Flash と Silverlight の脆弱性を突く．	検知不可能	・入口サイトに難読化 JS あり．難読化手法から Pseudo-Darkleech を使用 ・攻撃サイトに多重の難読化が施された JS あり．	検知不可能
マルチ環境解析と JS 解析の結果に対する突合分析	・特に新たな情報はない	・特に新たな情報はない	・入口，攻撃サイト共に，難読化コードはアクセスする度に変わるが，中身は変わらない．	・特に新たな情報はない

表 7 事例 2 の解析環境（合計 21 個）
Table 7 The analysis environment of case 2 (Total of 21).

仮想マシン	VMware Workstation 12					
ホスト OS	Windows 7 Professional					
ゲスト OS	Windows 7 Professional					Windows 10
ブラウザ	IE 8	IE 11.0.20	IE 11.0.38	Firefox 6	Chrome 47	IE 11.0.21
プラグイン	Adobe Flash Player 10, 19 Adobe Reader 9, 10 JRE 1.6, 1.7 Microsoft Silverlight 5.0	Adobe Flash Player 10, 19 Adobe Reader 9, 10 JRE 1.6, 1.7 なし	なし	なし	なし	Adobe Flash Player 18 Adobe Reader 11.0 JRE 1.7, 1.8

を調査に使用する．具体的な調査用 URL の取得手順は次のようになる．

Step1) 最新の Exploit Kit の解析レポートを取得する．
Step2) レポートに記載されている悪性 Web サイトの IP アドレスを取得する．
Step3) 取得した IP アドレスを VirusTotal で検索する．
Step4) 検索結果で表示される関連 URL を実験で使用する．

4.2.2 事例 2 の実験環境

分析に使用した解析環境を**表 7** に示す．事例 1 では Web ブラウザと OS を 1 種類ずつしか用意していなかったが，本実験では Web ブラウザを 6 種類用意し，OS に Windows 10 を追加している．また，6 通りの OS とブラウザの組合せに対し，表 7 に示しているプラグインを 1 つずつ導入している環境と，プラグインを導入していない環境を構築し，合計で 21 個の解析環境を構築した．事例 1 のように複数のプラグインを組み合せても挙動の変化にあまり影響しないことと，1 端末に 1 つのプラグインを入れた方がマルチ環境解析の際に環境の共通点を見つけやすく分析が容易になるため，本実験ではプラグインを 1 つずつ導入している．IE はバージョン 11 から User-Agent の形式が変わっており，バージョン別に挙動を比較するために IE 11.0.20 と，2016 年 12 月 25 日時点で最新の IE 11.0.38 を用意し

た．Chrome と Firefox は IE の次にシェア率が高いブラウザで，Firefox は OS に適応できる最古のものを利用した．プラグインの Silverlight は分析事例 1 で攻撃対象となっていたため，ブラウザに適応できる最古のバージョンを追加した．ただし，IE 11 では古いバージョンの Silverlight は自動ブロックされるため，IE 8 にだけ導入している．Windows 10 の Adobe Flash Player は，手動でインストールやアンインストールができない仕様になっているため，全環境にデフォルトバージョンのものが導入されている．つまり，Windows 10 の Adobe Flash Player 18 と表記している環境は，新たに何もプラグインを導入していない環境である．Adobe Reader 11.0 と JRE 1.7 は Windows 10 に適用できる中で最古のバージョンである．JRE 1.8 は Windows 10 がサポートしている中で最古のバージョンになっている．なお，Windows アップデートは基本的に適用していないが，IE 11.0.38 の環境だけは最新の状態にするため適用している．

マルチモーダル分析の起点となる解析レポートは，2017 年 1 月 13 日に Malware-Traffic-Analysis.net [17] に投稿された Rig Exploit Kit の解析レポートを使用し，2017 年 1 月 17 日に実験を行った．この解析レポートに記載されている入口サイト（activ****.com）の IP アドレスを VirusTotal で検索したところ，23 個の URL を取得した．この 23 個の URL に対して提案手法のクローキング分析を実施する．

4.2.3 事例 2 の分析結果

マルチ環境解析を行ったところ，実験に使用した 23 個の URL のうち，sdain****.com，video****.com，encin****.com では環境によって挙動を変えており，悪性な SWF ファイルなどをダウンロードしていた．この 3 個の悪性クローキングサイトを解析対象とする．3 個の入口サイトから同じ中継サイトへリダイレクトし，その後攻撃サイトへリダイレクトしていた．マルチ環境解析だけで得たクローキング情報は次のとおりである．

- ブラウザが IE の場合，入口サイトから中継サイトへリダイレクトする．それ以外はリダイレクトなし．
- 同じ IP によるアクセスが 2 回目以上の場合，中継サイトから空の Web サイトへリダイレクトする．初回アクセスであれば攻撃サイトへリダイレクトする．
- プラグインに Flash がある場合だけ，攻撃サイトから悪性の SWF ファイルをダウンロードする．
- OS とプラグインに関係なく，IE 8，IE 11.0.20，IE 11.0.21 の環境はすべてマルウェアに感染した．

本実験では，一度マルウェアに感染した環境と同じ環境を起動させ，同じ IP アドレスから再び入口サイトにアクセスすることで，同じ IP アドレスによる挙動の変化を確認した．さらに，マルチ環境解析では，ブラウザの種類によって攻撃の進行度を**表 8** のように分けることができた．Firefox と Chrome の環境は入口サイトからリダイレクトが発生せず，悪性な挙動をいっさい見せなかった．IE 11.0.38 の環境は攻撃サイトまでリダイレクトしていたが，マルウェアに感染しなかった．それ以外の IE の環境はすべて CERBER ランサムウェアに感染した．IE 11.0.38 はバージョンが新しいため，攻撃対象の脆弱性が存在せず，攻撃に失敗したと考えられる．攻撃サイトでダウンロードする SWF ファイルを VirusTotal で分析したところ，54 個中 26 個のアンチウイルスソフトが悪性と判定した．中継サイトや攻撃サイトの URL の特徴から，この攻撃には Rig Exploit Kit が用いられたことが分かった．

表 8 ブラウザ別の攻撃の進行度
Table 8 Progress of attack by web browser.

ブラウザ環境	攻撃の進行度
Firefox，Chrome	レベル 1
IE 11.0.38	レベル 3
IE 8，IE 11.0.20 & 21	レベル 4

次に入口サイト，中継サイト，攻撃サイトの JavaScript コードを解析した．JavaScript 解析だけで得たクローキング情報は次のとおりである．

- 入口サイトでは，ブラウザが IE の場合だけ iframe タグを挿入し，中継サイトへリダイレクトさせる．
- 中継サイトではブラウザ情報を詳細に調査し，ブラウザが IE の場合だけ iframe で攻撃サイトへリダイレクトさせる．IE 以外の環境か，User-Agent を偽装した場合は「404 Not Found」を表示する．
- 攻撃サイトでは，プラグイン状況によって悪用する脆弱性を変える．Flash がある場合は SWF ファイルをダウンロードさせて Flash の脆弱性を突く．それ以外の場合は IE の脆弱性を突く．

入口サイトの改ざんによって iframe タグだけが挿入され，JavaScript は挿入されていなかった．また，iframe タグの特徴から，Pseudo-Darkleech による改ざんと考えられる．中継サイトの JavaScript は難読化されていなかったが，攻撃サイトの JavaScript は多重に難読化が施されていた．中継サイトでは，User-Agent の値でブラウザを識別した後に，各ブラウザ固有のオブジェクトの有無によって正確にブラウザの種類を識別し，User-Agent を偽装していないか確認していた．各ブラウザ固有のオブジェクトを検知するコードは 5 件確認した．攻撃サイトでは，Flash コンテンツの利用の可否によって，悪用する脆弱性を変化させていた．

次にマルチ環境解析と JavaScript 解析の結果から，より詳しくクローキングを分析する．各 Web サイトで行われたクローキングは次のとおりである．

［入口サイト］

サーバ側のクローキングによって Web ブラウザを判別

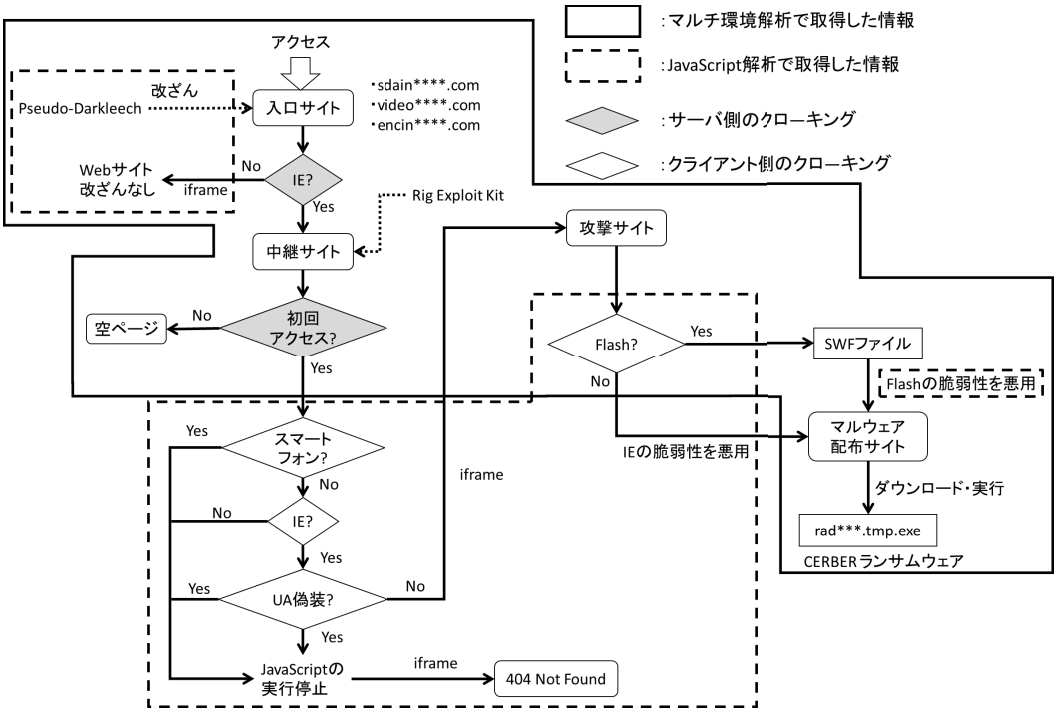

図 8 事例 2 の分析結果
Fig. 8 Analysis result of case 2.

表 9 事例 2 で取得した情報
Table 9 Information obtained in case 2.

	クライアント側の各種クローキングの有無	サーバ側の各種クローキングの有無	難読化の有無	短時間で変化するURLの有無
マルチ環境解析	・入口サイトでブラウザの種類を識別 ・中継サイトでIPアドレスを識別 ・攻撃サイトでプラグインを識別し，Flashの環境には悪性なSWFをダウンロード ・用意した環境の内，最新バージョン以外のIEはランサムウェアに感染		検知不可能	・中継サイト以降のURLが短時間で変化する ・URLの特徴からRig EKを使用
JS解析	・入口サイトに悪性なiframe ・中継サイトでブラウザを正確に識別し，User-Agentの偽装を確認．PCからIEを使用し，User-Agentを偽装していなければ攻撃サイトへリダイレクト ・攻撃サイトでプラグインを識別後，Flashがあればその脆弱性を，無ければIEの脆弱性を突く．	検知不可能	・攻撃サイトに多重の難読化が施されたJSあり．	検知不可能
マルチ環境解析とJS解析の結果に対する突合分析	・入口サイトでは，サーバ側でブラウザを識別後，改ざんによってiframeを挿入する．iframeの記述方法から，Pseudo-Darkleechが使用されている．	・入口サイトでブラウザの種類を識別 ・中継サイトでIPアドレスを識別し，初回アクセスであれば攻撃を継続	・難読化コードはアクセスする度に変わるが，中身は変わらない．	・Pseudo-DarkleechによってURLを動的に変化

し，IE の場合だけ Pseudo-Darkleech によって iframe を挿入し，中継サイトへリダイレクトさせる．IE 以外の場合は改ざんを行わない．

［中継サイト］

サーバ側のクローキングによって IP アドレスを確認し，初回アクセスであれば悪性な中継サイトを返す．2 回目以降であれば無害で中身が空の Web サイトを返す．中継サイトではクライアント側のクローキングによって，ユーザのブラウザ環境を詳細に調査する．ブラウザが IE であれば攻撃サイトへリダイレクトさせるが，それ以外のブラウザや，User-Agent を偽装している場合はそこで攻撃を停止する．

［攻撃サイト］

クライアント側のクローキングによって，環境に合わせた脆弱性を悪用する．プラグインに Adobe Flash Player があれば SWF ファイルをダウンロードさせ，Flash の脆弱性を悪用する．それ以外の場合は IE の脆弱性を悪用する．攻撃に成功すれば，CERBER ランサムウェアに感染する．

この悪性 Web サイト全体の挙動から，特定の脆弱性が存在する IE で，User-Agent を偽装せず，IP アドレスが初回アクセスの環境で入口サイトにアクセスした場合，CERBER ランサムウェアに感染すると判断できる．

分析結果のまとめを**図 8** に示し，提案手法で取得した情報を**表 9** に示す．提案手法によってクライアント側とサーバ側両方のクローキングの条件やその挙動をとらえ，攻撃対象とする環境の条件を導き出せた．また，攻撃に使用されたツールなども特定することができた．

5. 関連研究

クローキングを行う悪性 Web サイトに対して，Lu ら [18] は端末の環境を変更して解析する有効性を示しており，Wang ら [19] は複数環境のクライアント型ハニーポットによる解析技術を示している．Shiraishi ら [20] は，複数環境のクライアント型ハニーポットを用いて解析するマルチ環境解析システムを提案し，悪性 Web サイトの影響を評価する方法を与えている．Shiraishi らのマルチ環境解析では，複数環境の端末で悪性 Web サイトにアクセスし，その際の通信ログやダウンロードコンテンツを解析することで，悪性 Web サイトの挙動分岐条件の判定を試みている．Invernizzi ら [21] は悪性なクローキング Web サイトの検知システムを提案している．Invernizzi らのシステムでは，複数環境で Web サイトにアクセスし，環境ごとの通信や表示される画面の類似性を比較することでクローキングを検知する．ただし，検知が目的であるため，クローキングの挙動内容は分析していない．また，Shiraishi らや Invernizzi らのマルチ環境解析では，JavaScript を詳細に解析していないため，クローキングに関する情報は取得できるものの，それがサーバ側で実行されているか，クライアント側で実行されているか判定できない．

JavaScript のクローキングに関する研究として，Kolbitsch ら [22] は JavaScript によるクローキングを自動で解析するシステムを提案している．このシステムでは複数環境を装い，複数の実行パスを並行して探索することでクローキングに関する情報を取得する．Kapravelos ら [23] は JavaScript による解析端末の検知回避テクニックをとらえるシステムを提案している．このシステムでは，良性コードと，検知回避コードが含まれる悪性コードを比較することで検知回避テクニックを推定する．Takata ら [24] は特定の環境でしか実行されないリダイレクトコードを抽出するシステムを提案している．Kapravelos らと Takata らのシステムは JavaScript を解析することでクライアント側のクローキングを明らかにできるが，サーバ側のクローキングに関する情報は取得できない．

6. 手法の制限事項

6.1 マルチ環境解析

マルチ環境解析では，環境によって挙動に差異が見られた場合にクローキングを検知できるが，用意したすべての環境が同じ挙動を示した場合は，クローキングを検知できずに見逃してしまう．それとは逆に，すべての環境が違う挙動を示した場合や，同じ挙動を示した環境に共通点などの規則性が見られなければ，クローキングを検知できても，その挙動を変える条件を特定することができない．また，何らかのエラーなどでクローキングの条件に沿った挙動とは異なる挙動を示した場合は，クローキングの情報を誤ってとらえることがある．悪性なクローキングか否かの判断は通信ログ解析で行っているが，ブラウザの種類などによって表示画面を変え，なおかつリダイレクトが発生するような良性 Web サイトに対しても，誤って悪性なクローキングサイトと判定することがある．

6.2 JavaScript 解析

JavaScript 解析は基本的に手動で行っているため，複雑な難読化が施されたコードや，環境によって難読化解除後のコードが変化するものに対しては，クローキングを見逃すことがあり得る．JavaScript 解析を行う際には，3.3.1 項で述べたように，if 文や switch 文などの条件文に着目して解析を進める．for 文や try/catch 文などの制御文でクローキング処理が行われている場合，クローキングを見逃してしまうが，それらの制御文は文字列生成などの難読化解除の処理に使われている場合が多く，クローキング処理に使われることが少ないため，クローキングを見逃す可能性は低い．

6.3 突合分析

突合分析はマルチ環境解析の結果に依存しているため，マルチ環境解析でサーバ側のクローキングを検知できなければ，突合分析によってサーバ側のクローキングを導き出せない．また，サーバ側のクローキングは，マルチ環境解析の結果から推定して導出したものであるため，挙動を変える条件は，実際に PHP などのコードとして記述されている内容と異なっていることがあり得る．

7. むすび

本論文では，マルチ環境解析と JavaScript 解析を組み合せることによって，悪性 Web サイトのクライアント側とサーバ側で行われるクローキングに関する情報を詳細に分析する手法を提案した．マルチ環境解析だけの場合クローキングを検知できるが，それがクライアント側とサーバ側のどちらで実行されたか判別できず，JavaScript 解析だけの場合クライアント側のクローキングに関する情報しか取得できないため，サーバ側のクローキングに関する情報を取得することができなかった．提案手法では，マルチ環境解析と JavaScript 解析を行い，さらに双方の解析結果に対して突合分析を行うことで，クライアント側とサーバ側のクローキングに関する情報を取得する．提案手法で実在する悪性 Web サイトを分析したところ，サーバ側とクライアント側のクローキングを行う条件とその挙動をとらえることができた．また，攻撃条件や攻撃対象となる環境も導出できた．提案手法によって得られた挙動内容は，Exploit Kit を用いて攻撃手法の複雑化が進む DBD の攻撃基盤や CaaS（crimeware as a service）の解析に貢献すると期待できる．

今後の課題として，分析手法を自動化したシステムの構築があげられる．柴田ら [25] は JavaScript の API フックを用いて，クローキング機能を含む難読化 JavaScript を自動で解除する手法を提案している．そのため，Kolbitsch ら [22] のシステムに柴田ら [25] の手法を組み合わせることで，JavaScript のクローキングを自動で解析できると考えられる．サーバ側のクローキングに関しては，用意した環境のバリエーションに依存するため，より多くの OS やブラウザなどを利用して分析する必要がある．

参考文献

[1] トレンドマイクロ：Trend Labs 2017 年第 1 四半期 セキュリティラウンドアップ，入手先 ⟨http://www.trendmicro.co.jp/jp/security-intelligence/sr/sr-2017q1/index.html⟩（参照 2017-06-05）．

[2] Microsoft: SmartScreen フィルター機能，入手先 ⟨https://support.microsoft.com/ja-jp/help/17443/windows-internet-explorer-smartscreen-filter-faq⟩（参照 2017-05-01）．

[3] Google: Google Safe Browsing, available from ⟨https://developers.google.com/safe-browsing/⟩ (accessed 2017-05-01).

[4] アグスネット：aguse.jp，入手先 ⟨https://www.aguse.jp/⟩（参照 2017-05-01）．

[5] urlQuery.net, available from ⟨http://urlquery.net/⟩ (accessed 2017-05-01).

[6] Dell SonicWALL: Blackhole Exploit Kit: Rise & Evolution, available from ⟨http://software.sonicwall.com/gav/Blackhole%20Exploit%20Kit%20-%20Rise%20&%20Evolution.pdf⟩ (accessed 2017-11-30).

[7] Open QA: Selenium – Web Browser Automation, available from ⟨http://www.seleniumhq.org/⟩ (accessed 2017-05-22).

[8] Wireshrk, available from ⟨https://www.wireshark.org/⟩ (accessed 2017-12-01).

[9] VirusTotal, available from ⟨https://www.virustotal.com/⟩ (accessed 2017-05-22).

[10] Malware Domain List, available from ⟨https://www.malwaredomainlist.com/⟩ (accessed 2017-05-22).

[11] IBM：2016 年上半期 Tokyo SOC 情報分析レポート，入手先 ⟨https://www.ibm.com/blogs/tokyo-soc/wp-content/uploads/2016/02/tokyo_soc_report2016_h1.pdf⟩（参照 2017-12-01）．

[12] ラック：CYBER GRID VIEW Vol.3，入手先 ⟨https://www.lac.co.jp/lacwatch/pdf/20170202_cgview_vol3_f001t.pdf⟩（参照 2017-12-01）．

[13] NetMarketShare, available from ⟨https://www.netmarketshare.com/⟩ (accessed 2017-12-01).

[14] CVE Details, available from ⟨http://www.cvedetails.com/⟩ (accessed 2017-12-01).

[15] 笠間貴弘，中里純二，鈴木未央ほか：多様なセンサの観測情報を用いたマルチモーダル分析，電子情報通信学会 2012 年暗号と情報セキュリティシンポジウム（SCIS2012）(2012).

[16] 笠間貴弘，衛藤将史，井上大介：マルチモーダル分析による不正通信の検出，電子情報通信学会技術研究報告，Vol.112, No.315, pp.25–30 (2012).

[17] Malware-Traffic-Analysis.net, available from ⟨http://www.malware-traffic-analysis.net/index.html⟩ (accessed 2017-05-22).

[18] Lu, L., Yegneswaran, V., Porras, P. and Lee, W.: BLADE: An attack-agnostic approach for preventing drive-by malware infections, *Proc. 17th ACM Conference* (*ACM CCS 2010*), pp.440–450 (2010).

[19] Wang, Y.M., Beck, D. and Jiang, X., et al.: Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities, *Proc. Network and Distributed Systems Security Symposium*, pp.35–49 (2006).

[20] Shiraishi, Y., Kamizono, M., Hirotomo, M. and Mohri, M.: Multi-Environment Analysis System for Evaluating the Impact of Malicious Web Sites Changing Their Behavior, *IEICE Trans. Inf. & Syst.*, Vol.E100-D, No.10, pp.2449–2457 (2017).

[21] Invernizzi, L., Thomas, K. and Kapravelos, A., et al.: Cloak of Visibility: Detecting When Machines Browse a Different Web, *Proc. 37th IEEE Symposium on Security and Privacy*, pp.743–758 (2016).

[22] Kolbitsch, C., Livshits, B., Zorn, B. and Seifert, C.: Rozzle: De-Cloaking Internet Malware, *Proc. IEEE Symposium on Security and Privacy*, pp.443–457 (2012).

[23] Kapravelos, A., Shoshitaishvili, Y and Cova, M., et al.: Revolver: An Automated Approach to the Detection of Evasive Web-based Malware, *Proc. 22nd USENIX Security Symposium*, pp.637–651 (2013).

[24] Takata, Y., Akiyama, M. and Yagi, T., et al.: Mine-

Spider: Extracting URLs from Environment-dependent Drive-by Download Attacks, *IEICE Trans. Inf. & Syst.*, Vol.E99-D, No.4, pp.860–872 (2016).
[25] 柴田龍平，羽田大樹，横山恵一：Js-Walker: JavaScript API hooking を用いた解析妨害 JavaScript コードのアナリスト向け解析フレームワーク，コンピュータセキュリティシンポジウム 2016 論文集，Vol.2016, No.2, pp.951–957 (2016).

西尾 祐哉

2016 年佐賀大学理工学部知能情報システム学科卒業，2018 年同大学大学院工学系研究科博士前期課程修了．在学中，ネットワークセキュリティの研究に従事．

廣友 雅徳

2000 年徳島大学工学部知能情報工学科卒業，2002 年同大学大学院工学研究科博士前期課程修了，2005 年同大学院工学研究科博士後期課程単位取得退学．同年同大学工学部助手．2006 年ひょうご情報教育機構専任研究員．2008 年神戸大学大学院工学研究科助教．2011 年佐賀大学総合情報基盤センター特任助教．2013 年同大学工学系研究科准教授．博士（工学）．主に符号理論，情報セキュリティ等の研究・教育に従事．電子情報通信学会会員．

神薗 雅紀

PwC サイバーサービス合同会社サイバーセキュリティ研究所所長．大学時代に国立研究開発法人情報通信研究機構（NICT）nicter システムの研究開発に従事する．NICT, JPCERT/CC, IPA，某省・民間のセキュリティに関する研究開発・コンサルティングに従事し，セキュリティに関する多数の国家プロジェクトの研究員およびプロジェクトマネージャーを経験する．同時に，マルウェアの解析作業やインシデントが発生した際のレスポンス対応を行う．また，マルウェア対策研究人材育成ワークショップでは不正な Web サイトの検知・解析および標的型攻撃に関わる論文にて 2 年連続優秀論文賞を受賞．AVAR 等の国内外のセキュリティカンファレンスにて研究発表も行っている．2015 年より PwC に入社し，PwC サイバーサービス合同会社の設立メンバとして，新たなサイバーセキュリティサービス/インテリジェンスサービスを開発．サイバーセキュリティ研究所を率いて新たなコア技術の研究開発やサイバー攻撃の分析に従事．

福田 洋治 （正会員）

2000 年徳島大学工学部卒業．2002 年同大学大学院工学研究科博士前期課程修了．2005 年同大学院工学研究科情報システム工学専攻博士後期課程単位取得退学．博士（工学）（徳島大）．2005 年愛知教育大学教育学部情報教育講座助手．2008 年同大学講師，情報処理センター兼任．2016 年近畿大学理工学部情報学科講師．情報セキュリティ，コンピュータネットワーク，ICT 教育支援等の研究・教育に従事．2002 年電子情報通信学会オフィスシステム研究賞，2008 年本会 DICOMO シンポジウム優秀論文賞．2015 年電子情報通信学会高度交通システム研究会優秀論文賞．2016 年電子情報通信学会ライフインテリジェンスとオフィス情報システム研究会功労賞．電子情報通信学会会員．

毛利 公美

1993 年愛媛大学工学部情報工学科卒業．1995 年同大学大学院工学研究科情報工学専攻博士前期課程修了．2002 年博士（工学）（徳島大）．1995 年香川短期大学助手．1998 年徳島大学工学部知能情報工学科助手，2003 年同講師．2007 年岐阜大学総合情報メディアセンター准教授，2017 年同大学工学部電気電子・情報工学科准教授．ネットワークセキュリティ，符号・暗号理論，コンピュータネットワーク等の研究・教育に従事．電子情報通信学会シニア会員．

白石 善明 （正会員）

1995 年愛媛大学工学部情報工学科卒業．1997 年同大学大学院博士前期課程修了．2000 年徳島大学大学院博士後期課程修了．博士（工学）．2002 年近畿大学理工学部情報学科講師．2006 年名古屋工業大学大学院情報工学専攻助教授．2013 年神戸大学大学院電気電子工学専攻准教授．情報セキュリティ，コンピュータネットワーク，教育支援，知識流通支援等の研究・教育に従事．2002 年電子情報通信学会オフィスシステム研究賞，2003 年暗号と情報セキュリティシンポジウム（SCIS）20 周年記念賞，2006 年 SCIS 論文賞．2007，2008，2011，2013 年 DICOMO 優秀論文賞．2012 年電子情報通信学会ライフインテリジェンスとオフィス情報システム研究会功労賞．2015 年本会高度交通システム研究会優秀論文賞．2017 年電子情報通信学会関西支部活動功労賞．2017 年より電子情報通信学会情報通信システムセキュリティ研究専門委員会委員長．電子情報通信学会シニア会員．