クライアントハニーポットにおける攻撃検知手法の実装と評価

秋山満昭　岩村誠　川古谷裕平　青木一史　伊藤光恭

NTT 情報流通プラットフォーム研究所
東京都武蔵野市緑町 3-9-11
{akiyama.mitsuaki,iwamura.makoto,kawakoya.yuhei,
aoki.kazufumi,itoh.mitsutaka}@lab.ntt.co.jp

あらまし Web ブラウザの脆弱性を標的とする drive-by-download 攻撃によりマルウェアを自動的にインストールされる事例が多発しており，このような攻撃を行う悪性 web サイトの対策が急務になっている．本論文では drive-by-download 攻撃を行う悪性 web サイトを検知するためクライアントハニーポットの攻撃検知手法を提案および実装した．提案手法は複数の観点から web ブラウザを監視し攻撃を段階的に検知することにより，既知および未知の攻撃双方に対応可能にした．またこれら手法を組み合わせることにより検知可能な攻撃パターンを増加させた．

Implementation and Evaluation of Detection Methods on Client Honeypot

Mitsuaki Akiyama　Makoto Iwamura　Yuhei Kawakoya　Kazufumi Aoki
Mitsutaka Itoh

NTT Information Sharing Platform Laboratories
Midori-Cho 3-9-11, Musashino, Tokyo 180-8585 Japan
{akiyama.mitsuaki,iwamura.makoto,kawakoya.yuhei,
aoki.kazufumi,itoh.mitsutaka}@lab.ntt.co.jp

Abstract Countermeasures against malicious web sites are urgently needed because of increasing the number of incidents that vulnerable web browsers are infected malware by drive-by-download attacks. We proposed detection methods of drive-by-download attack for client honeypot system. Proposed methods focused on the behavior of web browser in the view points of exploitation phases: 1) preparation of exploitation, 2) the moment of exploitation and 3) behavior of after exploitation. By combining proposed methods, our client honeypot improved detection coverage without increasing false-positives.

1　はじめに

コンピュータウィルスやワーム，ボットなどの悪意のあるプログラム（以下，マルウェアと呼ぶ）への感染事例はインターネット黎明期から今現在まで依然として発生し続けており，その感染手法も年々多様化してきている．パーソナル FW 機能や BB ルータ配下でのエンドユーザの ISP 接続が一般的になるに従って，感染手法も OS の脆弱性を標的とするネットワーク型感染だけでなく，E メールや Web などを悪用した感染などネットワーク上のアプリケーションを悪用した感染手法が取られるようになってきた．2009 年 4 月に発生した JSRedir-R[8] は，Web サイトに悪性なコンテンツを埋め込み Web 閲覧者のマシンに対してマルウェアを自動的にインストールさせる drive-by-download と呼ばれる攻撃の一例である．Web サイトがマルウェアの発信源になる事例 [9] は数年前から急激に増加しており，対策が急務になっている．

現在ではマルウェアの発信源となる悪性 Web サイトに対抗するため，対策に必要となる悪性 Web サイトの情報収集を効果的に行うクライアントハニーポットの研究が行われている．我々の従来研究 [1][14][15][16] では，クライアントハニーポットを設計および実装し，実態調査を通して悪性 Web サイトや Web 感染型マルウェアの基本的性質を明らかにした．

一方で現在もなお攻撃に利用される脆弱性が

発見され続けており，攻撃手法の多様化に伴い攻撃検知率の低下が懸念される．そこで本論文では攻撃の多様化に対応した検知手法の提案と実装を行い，検知精度を評価する．また，検知手法を組み合わせることで検知範囲の拡大を試みる．

2 関連研究

クライアントハニーポットは web ブラウザを利用する high interaction 型と，web ブラウザを模擬したエミュレータを利用する low interaction 型に分けられ，攻撃の検知手法もそれぞれの方式で大きく異なる．前者の Capture-HPC[11] や HoneyClient[5] はシステムが乗っ取られた後に発生するファイルやレジストリの変化やプロセス生成から不正な挙動を検知する．後者は HoneyC[12]，SpyBye[7] が提案されている．HoneyC および SpyBye は snort やアンチウィルスソフトによるコンテンツのシグネチャマッチングにより exploit コードやマルウェア検体を検知する．

3 攻撃手法詳細

Drive-by-download 攻撃は Web ブラウザの脆弱性を標的としており，標的ホストに対してマルウェアを自動的にダウンロードおよびインストールさせる攻撃である (図 1)．Web コンテンツの中に Web ブラウザの脆弱性を攻撃する exploit コードが挿入されており，該当コンテンツを読み込むことで web ブラウザの制御が奪われる．脆弱性の多くは buffer-overflow などのオーバーフロー系である．リターンアドレスをメモリ上に予め配置した命令コード（以下，shellcode と呼ぶ）のアドレスに設定することでブラウザのプロセスを乗っ取る．Shellcode とは脆弱性を攻撃した後に実行させるための短い命令（機械語）コードであり，主にマルウェア本体のダウンロードと実行などを行う．

攻撃を安定して成功させるために HeapSpray という手法が脆弱性に対する攻撃と併用されている（図 2）．HeapSpray は JavaScript や VBscript を利用し，動的に shellcode を文字列として生成して web ブラウザのヒープ領域に格納する手法である．NOP などのスライディングコードと shellcode から構成される文字列ブロックを連続して生成し，web ブラウザのヒープ領域を数十 MB から数百 MB 程度確保する．スクリプトエンジンによって確保されるヒープ領域のアドレス帯はおおよそ予測出来るため，リターンアドレスを該当アドレス帯付近に設定することで，高確率で攻撃を成功させることが出来る．

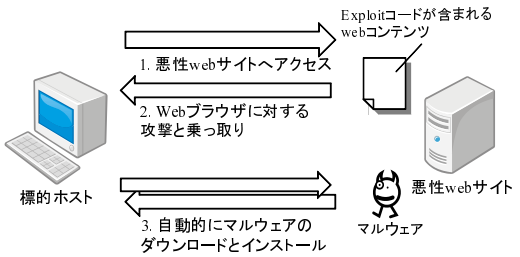

図 1: Drive-by-download 攻撃

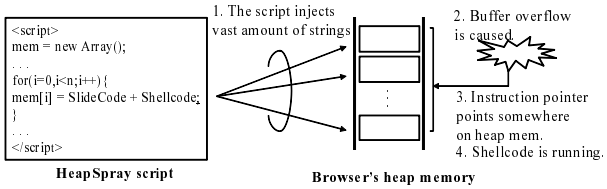

図 2: HeapSpray を利用した exploit 手法

一方，上記のオーバーフロー系の攻撃に見られる shellcode を実行させてシステムを乗っ取る手法とは異なり，MDAC(Microsoft Data Access Componet) の脆弱性 (MS06-014) は実行権限チェックのミスによって本来権限の無いシェルの起動やダウンロードファイルの実行などを行う．この脆弱性は HeapSrapy および shellcode を必要とせず，ブラウザ本来の挙動としてマルウェアをダウンロードおよびインストールさせるのが特徴である．

4 攻撃検知手法

JavaScript や VBscript は自由度の高い web コンテンツを実現できる利便性の高いスクリプト実行環境であるが，一方でその利便性の高さから攻撃にもしばしば利用されている．例えば，前述した HeapSpray，web ブラウザの種類やプラグインに合わせた攻撃手法の切り替え，複数の脆弱性に対する連続攻撃などである．また，web ブラウザおよびそのプラグインの脆弱性は今現在も発見され続けており，攻撃の標的となる脆弱性の種類が多岐にわたっている．

このような攻撃手法の多様化により，単一の手法での攻撃検知には限界があるといえる．そこで，我々の提案する検知手法では攻撃を段階的に分け，それぞれの段階における攻撃の特徴から検知する．またこれら検知を組み合わせることで攻撃手法の多様化に対応し，検知精度を向上させる．以下は攻撃を各段階に分けた時の提案検知手法である．

1. 攻撃前の準備段階
   スクリプトエンジンの挙動を監視することで HeapSpray を検知

2. 脆弱箇所に対する攻撃時
   脆弱箇所を監視することで脆弱性が攻撃されたかどうかを検知
3. 攻撃後の異常動作
   プロセスの挙動を監視し脆弱性を攻撃されたことに起因する動作かどうかを検知

本検知手法の実装は Windows マシン上で行った．また本検知手法はクライアントハニーポット上で利用されることを想定しているため，ベースとなるシステムはセキュリティパッチ未適用の Windows XP SP2 およびその上で動作する Internet Explorer 6.0 を利用する．このバージョンの Internet Explorer は，攻撃ツールの MPack[10] が標的としている脆弱性をほぼ全て包含しておりハニーポットに適しているため採用した．また，脆弱性があるバージョンのプラグインである WinZip 10.0，QuicTime 6.5.2，Acrobat Reader 8.1，Flash Player 9.45 も同時にインストールしている．

4.1 スクリプトエンジンの異常検知

前述の通り HeapSpray は脆弱性を攻撃するコードと組み合わせて行われる攻撃の準備動作である．本手法はスクリプトエンジンの動作を監視することで HeapSpray 特有の挙動である”巨大な文字列ブロックを大量に確保する動作”を検出する．

JavaScript および VBscript のスクリプトエンジンである jscript.dll および vbscript.dll がヒープを確保する場合は，oleaut32.dll の *SysAllocStringByteLen()* API が利用される．よって該当 API を API hook により監視することで，jscript.dll および vbscript.dll から該当 API が呼ばれた場合に引数として与えられる文字列サイズを取得する．取得した文字列サイズを集計し，HeapSpray の特徴であるヒープ確保量の総和および最大ヒープブロックサイズから HeapSpray の挙動を検知することを考える．なお，今回は HeapSpray の挙動に着目して特定の API を監視しているが，スクリプトエンジンから呼び出される他の API も同様に監視可能である．

4.2 脆弱性に対する攻撃検知

脆弱性が攻撃された瞬間を検知するため，特定の関数に対するデータフローを監視することにより脆弱性が突かれたかどうかを判別する．本検知手法および検知機構を HoneyPatch[17] と呼ぶ．HoneyPatch は脆弱性が存在するプログラムの該当箇所に対して入力される関数の引数や戻り値を監視する．例えばバッファオーバーフローの場合は，脆弱箇所に入力される文字列がバファサイズを超過しているかどうかを検知基準とする．脆弱性が突かれた瞬間に検知するため，その後何らかの理由により攻撃が途中で失敗した場合（例えばリターンアドレスの設定ミスによる shellcode の起動失敗など）においても攻撃の検知が可能である．

表 1: 実装した HoneyPatch の対応脆弱性

	脆弱性名	種類
Web ブラウザ (Internet Explorer)	MS06-001	WMF
	MS06-014	MDAC
	MS06-055	VML
	MS06-057	WVFIcon
	MS07-004	VML
	MS07-017	ANI
	CVE-2008-0015	Video Contorl
プラグイン	CVE-2006-5198	WinZip
	CVE-2007-0015	QuickTime
	CVE-2007-3456	Flash Player
	CVE-2007-5659	Acrobat Reader
	CVE-2008-2992	Acrobat Reader
	CVE-2009-0658	Acrobat Reader
	CVE-2009-0927	Acrobat Reader

Web ブラウザ自体の脆弱性や広く普及しているプラグインの脆弱性は標的になりやすい傾向がある．MPack には Internet Explorer およびプラグインの複数の脆弱性を攻撃する exploit コードが搭載されている．また，Acrobat Reader の脆弱性を標的とする攻撃も増加してきている [8]．このような状況を踏まえて，Internet Explorer 6.0 およびプラグインの脆弱性，合計 15 種類の HoneyPatch を実装している（表 1）．脆弱性情報は公開されている 3rd パーティパッチや脆弱性解析レポートを参考にできる [3][6][13]．

4.3 プロセスの挙動監視

本手法はファイルアクセス，レジストリアクセスおよびプロセス操作系 API を監視することで正常時とは異なる挙動を検出する．異常動作の検出は，正常状態の挙動および悪性な挙動を記述したルールファイルを用いてブラウザの挙動と比較して行う．例えば，Internet Explorer がキャッシュフォルダにファイルを作成する場合は正規の挙動とし，*C:\\WINDOWS\SYSTEM32* フォルダにファイルを作成しようとした場合は異常な挙動として検知する．

本手法はファイルアクセスやレジストリアクセス，プロセス生成を行う特定の API に対して hook を行い，該当 API の呼び出しイベントの検知および引数情報を取得しルールとの比較を行う．この手法は乗っ取られた後の挙動から検知する方法なので脆弱性の種類に依存しないことが利点である．しかしながら，下記理由により誤検知や見逃しが発生する．例えば，キャッシュ

フォルダへのファイル出力のような正常動作として登録されている同際に対して、同様の方法でマルウェアの実行ファイルを作成する場合に見逃しが発生する．また，新しくプラグインをインストールした場合は web ブラウザの挙動が変化するため，プラグイン特有の動作を予め調査しておきルールを更新する必要になる．例えば，PDF ファイルを web ブラウザで読み込むと新たに Acrobat Reader（AcroRd32.exe）のプロセスが起動される．よって，AcroRd32.exe の起動を正規の挙動として定義しておかなければ誤検知が発生する．

Capture-HPC や HoneyClient は，本手法と同様にアプリケーションの特定の動作を攻撃として検知する手法を採用している．しかしながら，上述の通り誤検知や見逃しの発生要因がある．本実装環境上では各種プラグインをインストールしているため，web ブラウザ自体の正規動作に加えてプラグインの正規動作を事前に学習し，誤検知要因を取り除く必要がある．

5　評価

本節では実験環境および実環境において提案手法の検知性能を評価する．提案検知手法をクライアントハニーポットに搭載し，実験環境および実環境で評価を行う．実験環境での評価は，MPack および web で公開されている PoC コード [4] によって構成した悪性 web サイトを巡回した結果を利用する．実環境での評価は Malware Domain List[2]（以下，MDL と呼ぶ）に登録されている 32446URL（2009 年 8 月 21 日時点で最新のもの）を対象として 2009 年 8 月 21 日から 23 日にかけて巡回した結果を利用する．なお，MDL の URL は drive-by-download 攻撃を行うものだけではなく，既に修正もしくは消滅しているものや単に不振なファイルをホスティングしているものなどを含む．実環境での評価では，MDAC を有効および無効にした 2 パターンのクライアントハニーポットを利用して攻撃検知を行うと共に実環境における攻撃パターンの比較を行う．

5.1　スクリプトエンジン異常検知の評価

図 3 は web ブラウザのスクリプトエンジンが確保する総ヒープ確保サイズと最大ヒープブロックサイズの分布である．実験環境の exploit コードではヒープの最大確保ブロックサイズが約 500KB から 4MB，総ヒープ確保サイズが約 90MB から 230MB に分布していた．また実際の web サイトを巡回した結果では図 3 のように広く分布しており，HeapSpray とは分布傾向が

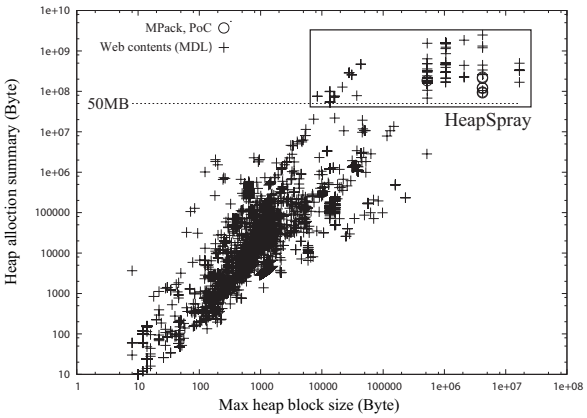

図 3: Web コンテンツのヒープレイアウト分布

異なることを確認した．Exploit コード付近に分布しているコンテンツ（総計 50MB 以上ヒープを確保するコンテンツ）については，内容を確認したところ HeapSpray を行っていた．

HeapSpray に必要なヒープ確保サイズは書き換えたリターンアドレスに設定したアドレスに依存する．Shellcode を含むヒープブロックを該当アドレスに確保することで，shellcode を実行でき攻撃が成功する．連続して確保したヒープブロックが該当アドレスまで届かない場合は攻撃失敗となる．該当アドレスまで到達するために大量のヒープブロックを確保する必要がある．一方，攻撃の成功率はヒープのブロックサイズにあまり影響されないことが実験により明らかになった．先ほどの exploit コードの傾向を踏まえ，検知基準となる閾値は総ヒープ確保サイズ 50MB 付近に設定できる．ただしこの閾値を下回る値でも HeapSpray は可能であるため検知の見逃しが発生する可能性があるが，攻撃の成功率は著しく低下することが実験によってわかっている．

5.2　脆弱性に対する攻撃検知の評価

実験環境にて，表 1 に示した脆弱性に対する攻撃を web ブラウザに行ってみたところ全ての攻撃を検知できた．本検知手法は脆弱性が攻撃されたかどうかを判断しており，対応している脆弱性であれば誤検知および見逃しは発生しないためである．一方，未対応の脆弱性を攻撃するコンテンツに対しては当然検知できなかった．また，本手法は脆弱性が突かれたかどうかを判断基準として検知するため，攻撃が途中で失敗した場合であっても脆弱性に対する攻撃を検知できた．

実環境において MDL 巡回時に HoneyPatch が検知した攻撃について，脆弱性の内訳を集計した結果が表 2 である．同一巡回中に複数の異なる脆弱性に対して攻撃を検知した場合はそれぞ

表 2: 攻撃に利用される脆弱性の割合

脆弱性名	MDAC 無効	MDAC 有効
MS06-001	0 (0%)	0 (0%)
MS06-014	0 (0%)	171 (63.8%)
MS06-055	4 (3.6%)	2 (0.7%)
MS06-057	16 (14.5%)	17 (6.3%)
MS07-004	6 (5.4%)	1 (0.3%)
MS07-017	5(4.5%)	0 (0%)
CVE-2008-0015	66 (60.0%)	67 (25.0%)
CVE-2006-5198	1 (0.9%)	1 (0.3%)
CVE-2007-0015	0 (0%)	0 (0%)
CVE-2007-3456	0 (0%)	0 (0%)
CVE-2007-5659	3 (2.7%)	4 (1.4%)
CVE-2008-2992	8 (7.2%)	4 (1.4%)
CVE-2009-0658	0 (0%)	0 (0%)
CVE-2009-0927	1 (0.9%)	1 (0.3%)
合計 (攻撃検知件数)	110	268

れをカウントしている．MDAC が有効な環境では MS06-014 と CVE-2008-0015 を同時に攻撃する exploit コードが多数検知したが，MDAC が無効な環境では他の脆弱性を代わりに標的とするなど，標的の環境に合わせた攻撃の変化を観測した．また，HoneyPatch が対応していない未知の脆弱性を標的とする攻撃については見逃しが発生しており，web ブラウザやプラグインの脆弱性は未知の脆弱性が多数存在することが想定されることから，すべての脆弱性に対して HoneyPatch のみで対応することは困難である．よって未対応の脆弱性に対する攻撃については HeapSpray 検知やイベント検知で対応する．

5.3 プロセス挙動監視の評価

実験環境において，HeapSpray による shellcode 実行を必要とする攻撃については攻撃が失敗する場合もあり，その場合は見逃しが発生した．一方，MDAC の脆弱性（MS06-014）を標的とする攻撃については失敗することが無くファイル出力やプロセス生成などのイベントが必ず発生するため検知できた．

また，新たにインストールしたプラグイン独自の挙動である設定ファイルのファイル読み込みやヘルパーオブジェクトのプロセス生成などは正規の挙動として登録しておく必要があるため，事前に様々なコンテンツを実行させて挙動を学習した．これにより，想定される正規動作の誤検知を無くした．

5.4 攻撃パターンと各検知手法の検知範囲

本節では実際の web 空間において，攻撃パターンの調査と各検知手法の検知性能を評価する．表 3 は攻撃手法のパターンであり，脆弱性の既知もしくは未知，HeapSpray の有無，攻撃成功もしくは失敗によって 8 種類に分類した．なお，ここで言う既知および未知の脆弱性とは HoneyPatch 対応および未対応の脆弱性と定義する．表 4 は攻撃パターンにおける各検知手法の検知範囲である．例えば，パターン B の場合は HeapSpray を行い特定の脆弱性を攻撃するが途中で攻撃が失敗しファイル出力やプロセス生成などのイベントが発生しない．表に示した通り，各攻撃検知手法は攻撃のパターンによって検知可能な場合と検知不可能な場合が存在する．そこで各攻撃手法を組み合わせることで検知精度の向上を図る．

表 5 は MDL 巡回時における各検知手法の検知率（検知数）である．表 6 は MDL 巡回時における検知情報から得られた攻撃パターンの割合である．MDAC が無効および有効の環境において，検知できたもので攻撃に失敗しているもの (パターン B，D，F) がそれぞれ 68.4%と 22.4%，攻撃が成功かつ脆弱性が特定出来なかったもの (パターン E，G) がそれぞれ 20.2%と 11.6%であった．MDAC が有効な環境では優先的に MDAC の脆弱性（MS06-014）を標的とするため，攻撃

表 3: 攻撃手法のパターン

脆弱性	HeapSpray	攻撃	パターン
既知	Yes	成功	A
		失敗	B
	No	成功	C
		失敗	D
未知	Yes	成功	E
		失敗	F
	No	成功	G
		失敗	H

表 4: 各検知手法の検知範囲

パターン	HeapSpray 検知	HoneyPatch	イベント検知
A	√	√	√
B	√	√	-
C	-	√	√
D	-	√	-
E	√	-	√
F	√	-	-
G	-	-	√
H	-	-	-

表 5: 各検知手法の検知率（検知数）

検知手法	MDAC 無効	MDAC 有効
HeapSpray 検知	161 (77.7%)	159 (63.8%)
HoneyPatch	104 (50.2%)	179 (71.8%)
イベント検知	61 (29.4%)	198 (79.5%)
合計 (URL ユニーク)	207	249

表 6: 攻撃パターンの分布

パターン	MDAC 無効	MDAC 有効
A	17 (8.2%)	110 (44.1%)
B	79 (38.1%)	7 (2.8%)
C	6 (2.8%)	54 (21.6%)
D	2 (0.9%)	8 (3.2%)
E	4 (1.9%)	1 (0.4%)
F	61 (29.4%)	41 (16.4%)
G	38 (18.3%)	28 (11.2%)
H	-	-
合計 (URL ユニーク)	207	249

に成功しファイル出力やプロセス生成などのイベント発生により 79.5%が検知出来ていた．一方，MDAC が無効な環境では攻撃に失敗する場合が多く，イベント検知では 29.4%の検知率になった．このような攻撃失敗に起因するイベント検知の見逃しについても，HeapSpray 検知および HoneyPatch を組み合わせることで救済できる．

パターン F ではインストールしていないプラグインの脆弱性を標的とした exploit コードが含まれている場合があった．まず HeapSpray を行った後，プラグイン（ActiveX Control）の起動を試みるがインストールされていないため，そこで攻撃が終了していた．また，同様にブラウザやプラグインのバージョンチェックを行い標的のバージョンかどうかを事前に確認するコンテンツも存在した．現在の環境では主要なプラグインの特定バージョンのみインストールしているため，マイナーなプラグインの脆弱性を標的とする攻撃に対しては見逃しが発生することが想定される．よって攻撃に利用されるプラグインの事前インストールと正規の挙動の学習が必要になる．また，同一プラグインでもバージョンにより脆弱性が排他のものが存在するため，排他関係にある脆弱性を一つのシステムに持たせることが難しい．よって，複数種類のアプリケーション（プラグイン）やバージョンで構成されたクライアントハニーポットシステムを用意する．

6 まとめ

Web ブラウザの脆弱性を標的としてマルウェアを自動的にインストールさせる drive-by-download 攻撃を検知するため，クライアントハニーポットに搭載する攻撃検知手法を提案し実装した．乗っ取られた後のシステムの動作から検知する手法は多くのハニーポットシステムで採用されている手法であるが，誤検知や見逃しが存在する．そこで，攻撃の段階に着目して各段階で検知する検知手法を実装した．攻撃の準備段階におけるスクリプトエンジンの異常検知では，攻撃に途中で失敗したとしても攻撃を検知可能にした．また，脆弱性箇所の監視により対応脆弱性に対する攻撃を確実に検知可能にした．これら検知手法を従来のイベント検知手法と組み合わせることで検知範囲を拡大させた．

参考文献

[1] M. Akiyama, Y. Kawakoya, M. Iwamura, K. Aoki, and M. Itoh. MARIONETTE: Client honeypot for Investigating and Understanding Web-based Malware infection on Implicated Websites. In *Joint Workshot on Information Security*, 2009.

[2] Malware domain List. `http://malwaredomainlist.com/`.

[3] Microsoft. Security research & defense. `http://blos.technet.com/srd/`.

[4] Milw0rm. Remote browser vuln exploitation. `http://milworm0rm`.

[5] MITRE. Honeyclient project. `http://www.honeyclient.org/`.

[6] National Institute of Standards and Technology. National vulnerability database. `http://nvd.nist.gov/`.

[7] N. Provos. Spybye. `http://www.provos.org/index.php?/categories/1-SpyBye`.

[8] Sophos. Malicious jsredire-r script found to be biggest malware threat on the web. `http://www.sophos.com/blogs/gc/`.

[9] Symantec. Global internet threat report volume xiv. `http://www.symantec.com/business/theme.jsp?themeid=threatreport`.

[10] Symantec. Mpack, packed full of badness. `http://www.symantec.com/enterprise/security_response/weblog/2007/05/mpack_packed_full_of_badness.html`.

[11] The Client Honeynet Project. Capure-HPC. `https://projects.honeynet.org/capture-hpc`.

[12] The Client Honeynet Project. HoneyC. `https://projects.honeynet.org/honeyc`.

[13] Zeroday Emergency Response Team (ZERT). Released patches. `http://www.isotf.org/zert`.

[14] 秋山満昭, 川古谷裕平, 岩村誠, and 伊藤光恭. クライアントハニーポットを用いた web 感染型マルウェアの実態調査. In **コンピュータセキュリティシンポジウム** *(CSS)*, 2008.

[15] 青木一史, 川古谷裕平, 秋山満昭, 岩村誠, 針生剛男, and 伊藤光恭. 能動的攻撃および受動的攻撃に関する調査および考察. **情報処理学会論文誌**, 50(9), 2009.

[16] 川古谷裕平, 秋山満昭, 青木一史, 伊藤光恭, and 高倉弘喜. スパムメールに起因する web 型受動攻撃の実態調査. In **情報通信システムセキュリティ研究会** *(ICSS)*, 5 2009.

[17] 川古谷裕平, 柳原忠明, 岩村誠, and 伊藤光恭. Honeypatch: Honeypot における攻撃検知手法の提案. In **電気電子情報学会** *2006* **年総合大会**, 2006.