HTTP Proxy を使った Cookie 挿入による不正通信の検出

加藤 雅彦 †‡　小出 洋 ††　金岡 晃 †††　松川 博英 ‡‡　前田 典彦 ‡‡‡
岡本 栄司 ‡

† 株式会社インターネットイニシアティブ
101-0051 東京都千代田区神田神保町 1-105 神保町三井ビルディング
masa@iij.ad.jp

†† 九州工業大学
820-8502 福岡県飯塚市川津 680-4

††† 東邦大学
274-8510 千葉県船橋市三山 2-2-1

‡ 筑波大学
305-8577 茨城県つくば市天王台 1-1-1

‡‡ トレンドマイクロ株式会社
151-0053 東京都渋谷区代々木 2-1-1　新宿マインズタワー

‡‡‡ 株式会社カスペルスキー
101-0021 東京都千代田区外神田 3-12-8 住友不動産秋葉原ビル 7F

あらまし　近年，組織の機密情報窃取を目的とした標的型攻撃が増加している．攻撃者は標的型メール等を使い，組織内 PC で RAT 等のバックドアプログラムを動作させ，遠隔操作によって情報窃取を行う．遠隔操作は Web 閲覧等に利用されている HTTP 通信を利用するため，正当な目的の通信との識別が困難である．そこで本論文では，組織内で HTTP 通信を中継する proxy で HTTP Cookie を自動挿入し，クライアントの応答を確認することで，不正プログラムによる HTTP 通信か，通常のブラウザによる通信かを判別する方法を考案した．評価実験を行い，結果として，一部の独自実装された不正プログラムによる HTTP 通信が本方式により検出されることを確認した．

Detection of suspicious HTTP communication based on Cookie insertion by HTTP proxy

Masahiko Kato†‡　Hiroshi Koide††　Akira Kanaoka†††　Bakuei Matsukawa‡‡
Norihiko Maeda‡‡‡　Eiji Okamoto‡

†Internet Initiative Japan Inc.
Jimbocho Mitsui Bldg., 1-105 Kanda Jimbo-cho, Chiyoda-ku, Tokyo 101-0051, Japan
††Kyushu Institute of Technology　†††Toho University　‡University of Tsukuba
‡‡Trend Micro Inc.　‡‡‡Kaspersky Labs Japan Inc.

Abstract　Recently, advanced persistent threats aiming at confidential information theft of an organization are increasing. An attacker operates RAT in infected PC and sends information from the PC to the attacker using HTTP. Since HTTP is normally used and allowed by FW, it is difficult to detect such suspicious communication. In this paper, we develop a proxy server to insert HTTP Cookie automatically. Whether the communication is send by a browser or by RAT is examined by checking reactions of HTTP clients to inserted Cookies. As a result, we show that the proposed system can detect such suspicious communication.

1　はじめに

近年，様々な組織の機密情報窃取を目的としていると考えられる，標的型攻撃が世界的に増加している．日本も例外ではなく，具体的な事例として，2011 年には三菱重工業が情報窃取を目的とした標的型攻撃をうけたことが明らかとなった [1]．同時期に衆議院や宇宙航空研究開発機構も同様の攻撃を受けている [2][3]．また，2013 年に入っても，農林水産省や外務省といった政府省庁が標的型攻撃を受け，情報窃取が行われるなど [4][5]，未だに攻撃は継続している状況にあり，対策が急務となっている．これら組織のネットワークやシステムはファイアウォールなどのセキュリティ装置で保護されており，インターネットから直接組織内ネットワークに接続し，内部の情報を窃取することは困難である．そのため，標的型攻撃では，直接外部から侵入しないような攻撃手法が用いられる．情報処理推進機構の資料では，攻撃の手法が記載されている [6]．一例として図 1 に攻撃の流れの概要を示す．

まず，攻撃者は不正プログラムが動作するように細工された pdf などのファイルを，受信者が誤って開封しそうな内容が記載された標的型メール等に添付し，それを受信者に向けて送信する．受信者はその添付ファイルを開封することで，不正プログラムに感染する．その後，必要があれば C&C サーバより追加の不正プログラムをダウンロードするなどして，組織内部の PC を遠隔操作するためのバックドアプログラムを動作させる．バックドアプログラムは HTTP などを使って組織内部から組織外部の攻撃者との通信パス（バックドア）を確立させる．バックドアを使うことで，攻撃者は組織外部から直接侵入することなく組織内部の PC を遠隔操作し，情報窃取を行うことが可能となる．

HTTP は組織内からインターネットの Web 閲覧などに使われており，外部との通信パスとして使用できる可能性が高い．また，攻撃に HTTP を利用すれば，正当な目的の通信と遠隔操作の通信を迅速に識別，検出し，遮断することが困難となる．前述の情報処理推進機構の資料によると，実際に遠隔操作の通信の約 3 割は

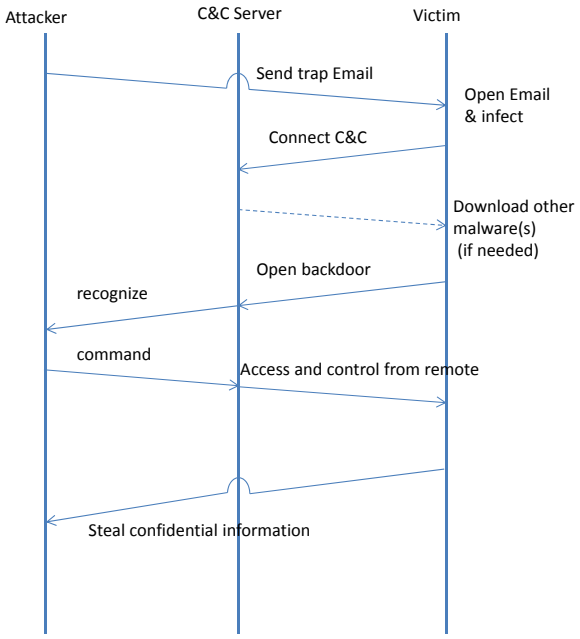

図 1: Targeted attack sequence overview

HTTP を利用していることがわかる [6]．加えて，実際の事例でも，攻撃が行われて相当の時間がたった後に，外部に情報が漏えいしている事実が判明したのち，ログ等を遡って調査することで攻撃が行われていたことが認知される事例が多々見受けられる．攻撃の検出を行うことが容易ではないことはこのような実例からも分かる．不正な通信の検出手法としては，HTTP ヘッダやコンテンツ部分のパターンマッチなども考えられるが，User-Agent の値などは容易に詐称可能なため，信頼性に問題がある．さらに，HTTP のコンテンツ部分も含めて，ネットワーク上に流れるすべてのデータを常時取得し，リアルタイムに通信内容の解析を行うといったことも考えられるが，現実的にはそのデータ量や処理能力は膨大となる．膨大なデータを迅速に処理するためには，大量のストレージや高速な解析処理能力を用意する必要があり，莫大なコストが発生する．そこで本論文では，大量のリソースを必要とせず，標的型攻撃に利用される HTTP のバックドア通信を，組織内の proxy サーバでリアルタイム検出するための一手法を考案した．以下，2 章で関連研究，3 章で提案手法，4 章で実装について記載し，5 章で評価実験とその結果を示し，6 章で考察を述べ，7 章

でまとめる．

2　関連研究

標的型攻撃そのものや，攻撃に使われるバックドア通信を検出するためにいくつかの手法が研究されている．そこで，関連研究について以下に述べる．

Guido Schwenk らは，HTTP の covert channel を検出するために，HTTP ヘッダや本文の長さ，エントロピー量，時間などを利用して，HTTP proxy 上で anomaly detection を行う，DUMONT システムを提案し，実装を行っている [7].

Yao-jun DING らはパケットの長さや時間間隔といった特徴量を，C4.5 アルゴリズムを使用して分類することによって，HTTP を使ったトンネル通信の検出を試みている [8].

Paul Giura らは標的型攻撃が複数の段階にわたり攻撃手法が組み合わされていることに着目し，攻撃の全体像を定義している．その全体像から，複数のフィルタを効果的に組み合わせることで攻撃検出の精度を上げる手法を提案している [9].

このように，標的型攻撃や HTTP を使った不正な通信の検出は複数の手法が提案されているが，これらの方法では正常な通信と不正な通信の閾値の設定が統計量に依存するため誤差の発生が避けられない．そのため，検出精度の向上に課題を残している．そこで本論文では，何らかの統計量で不正な通信の可能性があると判断された場合に，本当に不正なのかを自動的かつ迅速に確認するための手法を考案した．

3　提案手法

前述のとおり，バックドアプログラムが行う HTTP は，ブラウザを使用した一般ユーザによる Web 閲覧とプロトコル上の違いはなく，プロトコル上の異常検出によって不正な通信かどうか判断することは難しい．一方，バックドアプログラムが提供する機能や通信処理に着目した場合，バックドアプログラムはブラウザではないため，ブラウザで実装されているすべての機能をバックドアプログラムで実現する必要性はないと考えられる．ブラウザで実装されている機能としては，HTTP 通信のヘッダ処理やコンテンツ処理が考えられるが，今回は解析の負荷や取得データ量を考慮して，HTTP のヘッダ情報に着目し，ブラウザとバックドアプログラムの HTTP ヘッダ処理の違いから，不正な通信を検出する手法を考案した．本論文では，HTTP ヘッダの中でも Cookie ヘッダに着目した．Web アプリケーションのセッション管理などに用いられる Cookie ヘッダは，cookie の値を格納したり，ドメインによってヘッダをつけるかどうかの判断をしたりすることが必要で，HTTP ヘッダの中では比較的処理が多いと考えられるためである．バックドアプログラムの機能として cookie 処理を実装する必要性は少ないと思われるため，cookie にどのように応答するかどうかで，ユーザがブラウザを使っているかどうかを判別してみることとした．具体的には，組織内の proxy で本来の通信に存在しない独自の cookie を挿入し，その cookie が正しく処理されるか無視されるかといった，動作を確認できる実装を行うことを想定した．

動作の概要は以下の通りである (図 2).

1. クライアントから宛先 URL 向けの HTTP のリクエストを proxy で受ける．
2. proxy で当該クライアントが過去に当該 URL へアクセスを行ったかを調べる．
3. 過去にアクセスがあった場合は，proxy で挿入した cookie が存在するか，値が正しいかを確認する．正しい場合は正規のアクセス，正しくない場合は不審なアクセスとして検知する．過去にアクセスがない場合は，次回のアクセスから cookie をつけて通信を行わせるために，proxy で独自に Set-Cookie ヘッダを付加してレスポンスを返す．

また，ブラウザではない，HTTP を利用するアップデートプログラムや，ユーザや社内で独自に開発した HTTP アプリケーション等は，利用者が都度 URL を入力しないために通信の宛

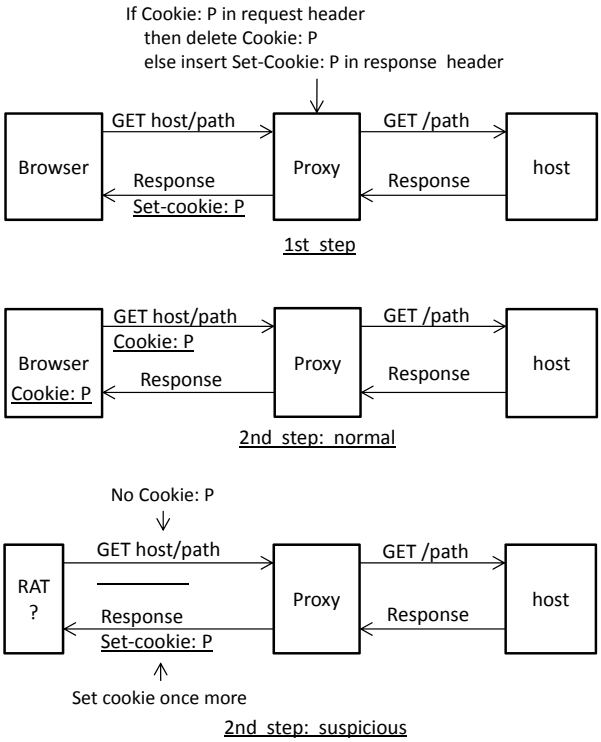

図 2: Detection method of suspicious access

先 URL が固定されていると考えられる．よって，そのようなアクセスはホワイトリスト化して誤検知を回避することを想定している．

4 実装

本 proxy サーバはリアルタイムに HTTP ヘッダの追加，削除，書き換えが必要なこと，および，cookie の発行情報管理が必要なことなどから，Squid[10] 等の設定のみで実装を行うことは困難と考えられる．そこで，HTTP 通信を中間で操作することに特化した proxy サーバである，ProxPy[11] を利用し，テスト実装を行うこととした (表 1)．ProxPy は python で記述されており，plugin 形式で proxy に追加処理を加えることが可能である．ただし，ProxPy の plugin インターフェースは，クライアントの IP アドレスを plugin プログラムに渡すことができない．そのため今回は，ProxPy の plugin インターフェースを修正し，IP アドレスの受け渡しが可能となるように修正を行っている．その他，ヘッダ追加処理に存在するバグの修正も行っている．また，cookie の発行状況，クライアント IP と宛先 URL の管理などは，扱う情報としてシンプルであり，Key Value Store 型のデータベース利用が適当であると判断し，オープンソースの KVS DB である，redis[12] を用いることとした．

表 1: SOFTWARE CONFIGURATION

Function	Software	Version
OS	Ubuntu	12.04
Proxy	ProxPy	r27
DB	redis	2.0.3
Language	Python	2.7

以下に実装した検出手法の概略を記載する．実際の処理の流れを図 3 で示す．cookie の発行状態は暫定的に，表 2 のものを定義した．なお，初期状態は DB に何も格納されていないとする．

1. HTTP リクエストがあると，クライアントの IP アドレスと宛先 URL をキーとして，DB に対して cookie の発行状態を確認する
2. DB に当該キーが存在しない場合は，初回の接続として，上記のキーに，1 回目の接続（set 状態）であることと，セットした回数を hash 値として格納する．
3. DB にキーが存在する場合は，過去にそのクライアントから URL への通信が行われていたと判断する．その場合は proxy から発行した cookie があるかないか，ある場合は値が正しいかをチェックする．
4. 正しい cookie がある場合は，その cookie を削除して宛先 URL にリクエストを中継する．
5. 2 回目以降の通信にもかかわらずクライアントから cookie が返ってこなければ，HTTP クライアントの実装に問題があるというカウントを行う．

以上をリクエスト時に処理する．リクエストの状態を DB に保持し，以下のレスポンス処理に利用する．

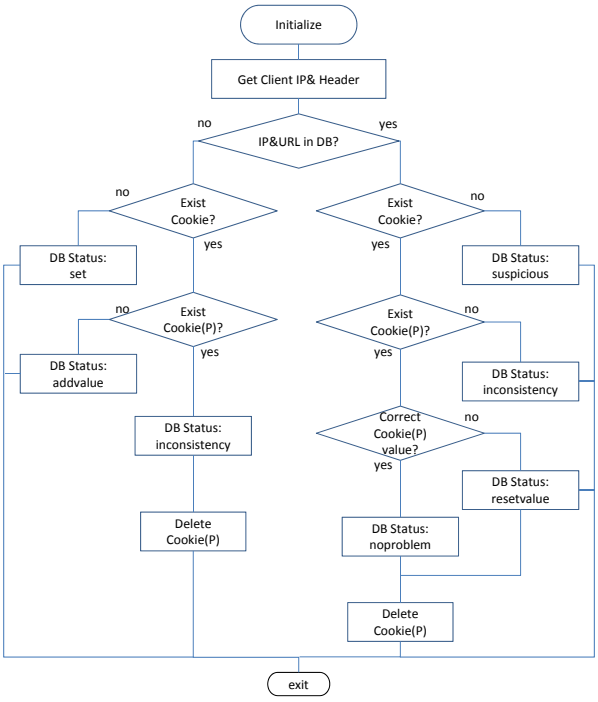

図 3: Detection flowchart

6. これらの条件判断の結果をすべて DB に格納し，サーバからのレスポンス時に DB の内容を確認する．

7. 初回の接続であれば proxy で独自の Set-Cookie を挿入する，正しいリクエストであればそのまま中継するなど，リクエスト時にセットされたステータスに従って cookie の発行処理等を行う．

表 2: COOKIE STATUS

set	cookie ヘッダそのものと proxy の値を挿入
addvalue	cookie ヘッダに proxy の値を追加
resetvalue	cookie をリセットする
neverset	cookie を挿入しない
suspicious	proxy の cookie がない
inconsistency	proxy の cookie の値が DB と一致しない

5 評価実験

本方式によって，実際にバックドアが検出されるか，また，通常の通信をおこなって誤検知が発生するかどうかの実験および確認を行った．バックドアプログラムは，HTTP を使うもの，かつ，C&C サーバへ接続を行おうとするもの，かつ，proxy サーバを使用するものという条件で抽出を行った．条件にあうマルウェアとして，2 検体をテストに使用することとした (表 3)．通常利用を想定したブラウザの通信も同時に行うことで，誤検知が発生するかどうかも同様にテストを行った．また，本テストを行うに当たり，事前調査として，九州工業大学の小出研究室で利用している proxy サーバで 3 か月にわたりヘッダ情報を収集し，cookie の利用状況を調査した．その結果，3 割以上の通信で Cookie が利用されていることを確認している．現在の Web サイトの多くは Cookie を使ってセッション管理等を行っているため，一般利用者はブラウザ設定で各種のヘッダ情報を無視する設定にしていないと仮定し，実験を行うこととした．

表 3: SAMPLE MALWARES

検体 1	
名称	BKDR_MALEX.RG
MD5	e42a5aea485bc97c584cbc22b41dd36a
検体 2	
名称	BKDR_DEMTRANC.R
MD5	6b3095d8452e7cd609545635ec5b7bfb

実験の結果，検体 1 は想定されたように cookie を処理することができず，不正として検出を行うことが可能であった．検体 1 は起動時に導通確認として，Google や AOL に HTTP 接続し，感染環境が通信可能かどうかを判断している．本方式では，その通信も不審な通信として検出した．同じ接続元クライアント IP でも，ブラウザを利用して Google や AOL の Web サイトを閲覧した場合は不正と判断せず，false positive を起こさなかった．

以下はテストプログラムによる検出時の動作例である．

初回アクセス時の cookie の挿入．ProxySessionID という cookie が proxy により挿入された cookie となる．

```
10.2.1.97:proxy.pcanywhere.net ProxySessionID=1;
expires=Mon, 30-Jul-2014 23:59:59 GMT; path=/;
host=proxy.pcanywhere.net
```

cookie を受け付けない状態の出力． IP アドレスと宛先 URL の組み合わせで，Cookie is not accepted というメッセージを出力している．

```
10.2.1.97:proxy.pcanywhere.net : Cookie is not accepted
REQ #14 method: GET ; host: ('proxy.pcanywhere.net', 80) ;
path: /AES225694521O28513.jsp?2rlcSHzCOgI0UPDbdp=L0NY9d+
BoRrlCQpGLQhnk=+9ViojKTrbkOalsSf/k=+LB=hSk=v9LQhGt0
p790vBbSqIJShGci7AA ; proto: HTTP/1.1 ; len(body): 0
Host: proxy.pcanywhere.NET
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
```

一方，検体 2 は検出されず，false negative となった．理由は後述する．
また，同時に通常のブラウザ (本実験では Windows 版の Firefox を使用) を使い，Yahoo 等のポータルサイト，朝日新聞 DIGITAL や YOMIURI ONLINE といったニュースサイト等，いくつかの Web 閲覧を行ったところ，不正プログラムとして検知された通信は存在しなかった．

6 考察

評価実験の結果，cookie の処理を行わないバックドアプログラムが存在することが確認できた．不正サイト，正規サイトといった URL のブラックリストや，PC のアンチウイルスソフト等に依存せず，バックドアプログラムからの通信を不正な通信としてリアルタイムに検出することが可能であった．本方式のように，通信に強制的に割り込みデータ操作を行うことで，PC に何らかのエージェントを導入することなく，また，User-Agent のような信頼性の低い情報を利用することなく，同一 PC 上のバックドアプログラムからの通信を proxy サーバでリアルタイムに識別することができる可能性があることがわかった．
一方，今回行ったテストにより，複数の課題があることも分かっている．cookie を無視するブラウザの誤検知，cookie を処理するバックドアプログラムの存在，HTTP/1.1 の持続的な接続，cookie 発行数に関してである．まず，cookie を無視するブラウザの存在についてである．cookie 処理を全く行わないと Web 閲覧に支障をきたすため，完全に cookie が無視されることはないと考え，さらに事前調査により通常利用において cookie は相当量使われているということを確認しているが，一方で cookie に応答しないためのブラウザプラグインなども存在する [13]．cookie を無視されるとバックドアプログラムとみなされるため，特定クライアント IP で false positive が多発する．よって，クライアント IP 単位でのホワイトリストなどの対応が必要となると考えられる．
次に，cookie を正常に処理するために，本方式で false negative を起こしてしまうバックドアプログラムが存在したことに関して，その原因を調査した．バックドアプログラムが Windows のシステムファイル wininet.dll 内にある HttpOpenRequest API を使っている場合，HTTP のヘッダは自動的に Windows API により処理されてしまう [14] ため，バックドアプログラムが意図的に HTTP のヘッダ処理を実装しなくても cookie が正常に処理される．よって，false negative となる．HTTP に挿入するヘッダとして，cookie 以外にも，URL リダイレクションを使う方法等も IPA により検討されている [6] が，cookie と同様にバックドアプログラムが使用している Windows API が処理してしまうものは不正な通信の識別には利用が困難と考えられる．これらの API を使用するバックドアプログラムの識別には別の方法を検討する必要があると考えられる．
HTTP/1.1 の持続的な接続による検出の遅れも検討が必要である．HTTP/1.1 はその通信仕様上，1 回の TCP 接続でまとめて HTTP GET を行うことが可能である．そのため，初期状態の接続において，cookie がついていない GET リ

クエストが同時に複数発生する．今回の方式では proxy は初回接続時，かつ cookie が無いリクエストが同時に到着すると，Set-Cookie を複数発行してしまうため，false positive につながる可能性が高い．今回の実験ではその対策として，Set-Cookie の発行が 2 回では不正とは検出しないこととして誤検知を回避している．
cookie の発行数については，本実験では特に事前のフィルタ等を使用せず，アクセスする URL 全てに cookie を発行している．検出漏れの減少は期待できるものの，ブラウザに大量の cookie が保存されるため，リソース上の無駄は多くなる．実際にどの程度性能に影響を及ぼすかについては，今後調査を行う予定である．
最後に，ProxPy の現在の仕様であるが，Transfer-Encoding: chunked の場合，複数のチャンクを 1 つにまとめて中継してしまう．ブラウザによってはここでプロトコル違反が発生する事象が見られたが，本方式の有効性検証とは問題の本質が異なるため，今回は問題としては取り上げていない．

7 まとめ

近年，組織の機密情報窃取を目的とした標的型攻撃が増加している．HTTP を使用して不正な通信を行っている場合，一般ユーザの Web 閲覧と区別がつかないため，検出が困難となっている．統計値や HTTP のパターンマッチにより攻撃を検出する方法もあるが，検出精度やコスト等に課題がある．そこで本論文では，統計値やパターンマッチに依存せずに不正な通信を検出するための手法を考案した．具体的には，HTTP proxy 上で本来の通信にはない cookie を独自に挿入し，クライアント PC がその cookie をどのように処理するかを確認することで，ブラウザによる通信かバックドアプログラムによる通信かを分類する．さらに，本手法を proxy サーバとして実装しテストを行った．実際の Web 閲覧を行いつつ，バックドアプログラムを動作させることによって，同一端末，同一 URL でもバックドアプログラムによる通信とブラウザによる閲覧をリアルタイムで分類することに成功した．
一方，cookie を挿入する手法では検出が困難なバックドアプログラムの存在も確認された．今後は，バックドアプログラムの動作を確認するために，挿入するヘッダの追加や変更，データのバリエーション増加，cookie の状態定義見直し等により，検出精度の向上を行う予定である．加えて，cookie の発行を少なくすることができるように，本手法の前段に統計量を用いたフィルタの導入を検討，実装を行う予定である．

参考文献

[1] 三菱重工業： コンピューターウイルス感染に関する調査状況について（その 1），http://www.mhi.co.jp/notice/notice_110930.html

[2] 衆議院： 衆議院へのサイバー攻撃報道に関する件，http://www.shugiin.go.jp/itdb_kaigiroku.nsf/html/kaigiroku/009017920111114005.htm

[3] 宇宙航空研究開発機構： JAXA におけるコンピュータウイルス感染の発生及び情報漏洩の可能性について，http://www.jaxa.jp/press/2012/11/20121130_security_j.html

[4] 農林水産省： 農林水産省へのサイバー攻撃に関する調査結果（中間報告）の公表について，http://www.maff.go.jp/j/press/kanbo/hisyo/130524_1.html

[5] 外務省: 外務省ネットワークから外部への情報流出，http://www.mofa.go.jp/mofaj/press/release/25/2/0205_07.html

[6] 情報処理推進機構： 「標的型メール攻撃」の対策に向けたシステム設計ガイド，http://www.ipa.go.jp/security/vuln/newattack.html

[7] Guido Schwenk and Konrad Rieck, Adaptive Detection of Covert Communication in HTTP Requests, *Computer Network*

Defense (*EC2ND*), *IEEE Seventh European Conference*, pp. 25-32 (2011).

[8] Yao-jun DING and Wan-dong CAI, A Method for HTTP-Tunnel Detection Based on Statistical Features of Traffic, *Communication Software and Networks* (*ICCSN*), *IEEE 3rd International Conference*, pp. 247-250 (2011).

[9] Paul Giura and Wei Wang, A Context-Based Detection Framework for Advanced Persistent Threats, *International Conference on Cyber Security*, pp. 69-74 (2012).

[10] Squid: `Optimising Web Delivery` , `http://www.squid-cache.org/`

[11] ProxPy: `A Python HTTP/HTTPS Proxy` , `http://code.google.com/p/ProxPy/`

[12] Redis: `http://redis.io/`

[13] Cookie Monster: `https://addons.mozilla.org/ja/firefox/addon/cookie-monster/`

[14] Microsoft Developer Network: HttpOpenRequest function, `http://msdn.microsoft.com/en-us/library/windows/desktop/aa384233%28v=vs.85%29.aspx`