6X-06

XHR2を用いたWebサイト閲覧者の属するネットワーク構成情報の採取法の提案

細井 理央† 磯 侑斗‡ 桐生 直輝‡ 塚本 耕司‡ 高須 航‡
山田 智隆‡ 武居 直樹‡ 石川 貴之† 齋藤 孝道†
明治大学† 明治大学大学院‡

1. はじめに

JavaScript，Flash 及び HTTP ヘッダから採取できる UserAgent や画面解像度などの，Web サイト閲覧者の端末に関する情報（以下，特徴点と呼ぶ）を利用することで，Web サイト閲覧者の端末の識別が可能であることが示された[1]．近年では WebRTC の登場や Web ブラウザの発展に伴い，Web ブラウザを通して特徴点の一つである Web サイト閲覧者のプライベート IP アドレスの採取や採取したプライベート IP アドレスを用いることで Web サイト閲覧者の端末が属するネットワークセグメント内に存在するホストの検出が可能となった．たとえば，ネットワークセグメント内に存在するホストを検出する既存の手法として，HTML の img 要素を用いる手法[2]や JavaScript の XHR2（XMLHttpRequest2）を用いる手法[3]がある．しかし，これらの既存手法では Windows OS のホストを検出することができない．そこで本論文では，XHR2 を用いた既存手法を改良することで，Windows OS のホストも検出可能なホストの検出法を提案する．また，提案手法により，検出したホストの OS が Windows であるのか，その他の OS（Linux，Mac OS X，Android，iOS のいずれか）であるのかの推測も可能であることを示す．本論文において，ネットワークセグメント内に存在するホストのプライベート IP アドレスと OS の種類に関する情報をネットワーク構成情報と呼ぶ．

採取したネットワーク構成情報を特徴点の一つであるグローバル IP アドレスと組み合わせることで，たとえば，企業組織の運営するネットワークの規模や構成を推測できるので，さらなる攻撃や侵入への足がかりになると考えられる．

A Proposal of Method to Obtain Website Visitors' Network Environment with XHR2
†Rio HOSOI ‡Yuto ISO ‡Naoki KIRYU ‡Koji TSUKAMOTO ‡Ko TAKASU ‡Tomotaka YAMADA ‡Naoki TAKEI
†Takayuki ISHIKAWA †Takamichi SAITO
†Meiji University ‡Graduate School of Meiji University

2. 既存のホストの検出法

本節では，既存手法[3]について説明する．これを改良し，本論文での提案手法とする．

2.1. 概要

既存手法[3]では，JavaScript を利用し Web サイト閲覧者の端末が属するネットワークセグメント内の全ての送信先 IP アドレスに HTTP リクエスト（以下，リクエストと呼ぶ）を送信することで，そのリクエストに対するエラー応答の有無によりネットワークセグメント内に存在するホストの検出を行う．エラー応答が返される場合，リクエスト先ホストが存在すると判断し，エラー応答が返されない場合，リクエスト先ホストが存在しないと判断する．

Web サイト閲覧者のプライベート IP アドレスは，WebRTC API を利用することで採取することができる[4]．リクエストの送信先 IP アドレスは，一般的に家庭内で使われる IP アドレスのネットマスクが 255.255.255.0 の場合が多いので，Web サイト閲覧者から採取したプライベート IP アドレスの第 4 オクテットを 1 から 254 までの数値におきかえた IP アドレスを使用する．よって，たとえば，Web サイト閲覧者から WebRTC API を用いて採取した IP アドレスが 192.168.10.5 の場合，192.168.10.1 から 192.168.10.254 までの 254 個を送信先 IP アドレスとして用いる．

既存手法で用いるリクエストを送信するソースコードを図 1 に示す．

```
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://" + target + ":" +
port + "/"+Math.random());
xhr.send();
```

図 1　XHR2 でのリクエストの送信

図 1 の *target* にはネットワークセグメント内に存在すると予想する全ての送信先 IP アドレスを順に入れ，*port* にはランダムなポート番号を指定してリクエストを送信する．

3. 提案手法

本節では，2.1 節で示した XHR2 を用いた既存手法を改良したネットワーク構成情報の採取法について説明する．

3.1. ホストの検出法

XHR2 を用いた既存手法では，ランダムなポート番号を指定しリクエストを送信していたが，Windows ではデフォルトの設定により，ほとんどのポートで受信したパケットを破棄しエラー応答を返さないので，OS が Windows であるホストを検出することができない．

提案手法では，ポート番号を Windows においてダイレクトホスティング SMB サービス[5]に使われるポート番号である 445 番に変更した．このポート番号は Windows において，デフォルトで外部からの接続を待ち受けておりパケットが破棄されることはなく，想定されていないメッセージを受信した場合にはコネクションを切断しエラー応答が返される．また，445 番ポートはその他の OS において，デフォルトで外部からの接続を待ち受けていないので，エラー応答が返される．そのため，送信先ホストが存在する場合，Windows とその他の OS では 445 番に送信したリクエストは，破棄されずエラー応答が返される．一方，送信先ホストが存在しない場合，エラー応答が返されない．これらの事実より，既存手法では検出できないことがあった Windows OS のホストについても検出することが可能となる．

3.2. OS の推定

前述のとおり，デフォルトでは，Windows とその他の OS で接続を待ち受けていないポートへのリクエストを受信した際の挙動が異なる．この性質を利用することにより，リクエストの送信先ホストが Windows であるのか，その他の OS であるのかを推定する．

OS を推定するためのリクエストは，Windows でエラー応答が返されず，その他の OS でエラー応答が返されるようなポートを指定すればよいので，今回はそのうちの 1 つである 446 番を使用した．3.1 節に示した方法によって存在を確認したホストの 446 番ポートに対しリクエストを送信した場合，Windows OS において，デフォルトで外部からの接続を待ち受けていないので，パケットは破棄されエラー応答は返されない．また，3.1 節で述べたことと同様にその他の OS ではリクエストに対しエラー応答が返される．これらの事実より，Windows OS を除くその他の OS であるホストのみ検出することが可能である．

3.1 節と本節で示した 2 つのリクエストに対するエラー応答とエラーの種類を表 1 に示す．OS が Windows OS の場合，445 番へのリクエストに対しては，RST/ACK というエラー応答が返され，446 番へのリクエストに対しては，エラー応答は返されない（表 1 では N/A と表記）．その他の OS の場合，445，446 番へのリクエストに対しては，Destination Unreachable というエラー応答が返される．ホストが存在しない場合，エラー応答は返されない．

表 1　各 OS とポート番号にリクエストを送信した際のエラーの応答と種類

OS	ポート	エラー応答	エラーの種類
Windows	445	有	RST/ACK
	446	無	N/A
その他	445, 446	有	Destination Unreachable
無	445, 446	無	N/A

この表より，Windows OS とその他の OS（Linux，Mac OS X，Android，iOS のいずれか）の区別が可能である．

4. まとめ

本論文では，Web サイト閲覧者が属するネットワークセグメント内に存在する OS が Windows とその他の OS であるホストの存否の検出法と，検出したホストの OS が Windows であるのか，その他の OS であるのかの推定法について提案した．これら得られた情報は，Web ブラウザを通して，サーバサイドで採取できるので，Web browser Fingerprinting における新たな特徴であると言える．

5. 参考文献

[1] P. Eckersley, How Unique is Your Web Browser? In Proc. Privacy Enhancing Technologies Symposium (2010), LNCS vol.6205
[2] https://www.browserleaks.com/
[3] Wade Alcorn, Christian Frichot, Michele Orru, The Browser Hacker's Handbook, WILEY, 2014
[4] http://net.ipcalf.com/
[5] http://support.microsoft.com/kb/204279/ja