情報処理学会論文誌 Vol. 49 No. 7 2643–2655 (July 2008)

Anti OS Fingerprintingシステムの提案と実装

三村 守[†1] 中村康弘[†2]

TCP/IP の規約は RFC（Request for Comments）に示されているが，各 OS（Operating System）における TCP/IP スタックの実装には差異がある．この差異による OS の挙動は，OS Fingerprint と呼ばれる．OS Fingerprint を分析することにより，遠隔でネットワークに接続するホストの OS を推定することができる．また，多くのホストから IP ヘッダの情報を収集して分析することにより，ネットワーク内部のトポロジを推定したり，稼動ホスト台数を検出したりすることができる．同様に，悪意ある第三者もネットワークを監視し，ネットワークに接続するホストの OS やネットワークに関する情報を収集し，攻撃に役立てることができる．とくに，OS Fingerprint の収集が受動的に行われた場合には検知が難しく，ネットワーク管理者にとっては脅威である．本論文では OS Fingerprint を収集して分析する行為に対し，ネットワークの経路上で OS Fingerprint を消去する Anti OS Fingerprinting システムを提案する．さらに Anti OS Fingerprinting システムを実装し，検証実験によりその有効性を示す．

A Proposal and the Implimentation of the Anti OS Fingerprinting System

Mamoru Mimura[†1] and Yasuhiro Nakamura[†2]

RFCs (Request for Comments) show protocols of TCP/IP. However, the implementations of the TCP/IP stack are different in individual OS (Operating System). The behavior caused by different implementations is called OS Fingerprint. By analyzing OS Fingerprint, the OS of a host connecting to the network at a remote station can be guessed. We can also collect and analyze information in the IP header from many hosts, and can estimate the network topology and a number of hosts in action. In a similar way, a malicious third person can also watch the network, and collect information on OS and network for the purpose of attacks. In particular, it is difficult to detect a malicious third person if the collection of OS Fingerprint is made passively. It is a threat for network administrators. In this paper, we propose an Anti OS Fingerprinting System in which OS Fingerprint is removed on route when a malicious third person tries to collect and analyze OS Fingerprint. Further, we show its effectiveness by verification experiments based on the proposed Anti OS Fingerprinting System.

1. はじめに

近年，LAN（Local Area Network）を経由してインターネットに接続するホストは急激に増加している．これにともない，ネットワークに接続するホストの脆弱性が指摘されることが多くなってきている．インターネットに直接接続するサーバだけでなく，LAN を経由するクライアントにおいてもバッファオーバフロー等の脆弱性が多数報告されており，問題となっている．ソフトウェア等の脆弱性関連情報に関する届出は年々増加しており，潜在していた脆弱性が顕在化してきているものと考えられている[1)]．ネットワークに接続するホストの脆弱性は，OS（Operating System）に密接に関係している．OS のセキュリティ問題は毎日のように報告されており，修正プログラムの配布も追いつかない場合がある．また，まだ報告されていない脆弱性を利用した攻撃も多く発生するようになってきている．このように，OS はつねに修正すべき問題をかかえていると考えられる．すなわち，LAN 内部で稼動する OS の種類やバージョンに関する情報が外部に漏洩すれば，その情報によって LAN 内部で稼動するホストの脆弱性が外部に知られることになる．

悪意ある第三者はネットワークを監視し，TCP/IP スタックの挙動を観察することで，遠隔でネットワークに接続するホストの OS を推定することができる[2)]．これは，TCP/IP の規約は RFC（Request for Comments）に示されているにもかかわらず，各 OS の TCP/IP スタックの実装に差異があるためである．この問題は，TCP/IP に係る既知の脆弱性に関する調査報告書[3)] では，通常でないパケットへの応答によって OS の種類が特定できる問題として扱われている．悪意ある第三者はネットワークを監視して取得した挙動を，既知の OS 固有の挙動を集めたデータベースと比較することにより遠隔で OS を推定する．本論文ではこのような OS を推定する行為を OS Fingerprinting と呼び，推定のために利用される OS 固有の挙動を OS Fingerprint と呼ぶ．OS Fingerprinting により，悪意ある第三者はネットワークに接続するホストの脆弱性を知り，攻撃に役立てることができる．そこで本研

†1 海上自衛隊
Japan Maritime Self-Defense Force
†2 防衛大学校情報工学科
Department of Computer Science, National Defense Academy

究の目的を，OS Fingerprinting を無力化し，ネットワークの安全性を高めることとする．

以下，2 章では OS Fingerprinting の脅威，手法および対策について述べ，関連研究から問題点を明らかにする．3 章では提案手法である Anti OS Fingerprinting システム（AOFS）の動作を説明し，4 章では検証実験によりその効果を確認する．5 章では検証実験の結果をまとめ，AOFS の有効範囲について考察し，最後に今後の課題を述べる．

2. OS Fingerprinting

この章では OS Fingerprinting の脅威について述べ，その手法と対策をまとめ，関連研究から問題点を明らかにする．

2.1 OS Fingerprinting の脅威

OS Fingerprinting により，悪意ある第三者は遠隔でネットワークに接続するホストの OS を推定することができる．しかも，OS Fingerprinting は nmap[4] や p0f[5] のような誰でも無償で入手できるツールにより，専門知識がない者でも容易に実施することができる．nmap は多数の OS Fingerprint に関するデータベースを持っており，OS のバージョンまでも推定することが可能である．悪意ある第三者は攻撃対象とするホストの OS を推定すると，その OS の脆弱性に関する情報を調査する．近年では OS のセキュリティ問題は毎日のように報告されており，OS の脆弱性に関する情報を見つけることは容易である．遠隔でバッファオーバフローを引き起こす脆弱性はとくに危険であり，任意のコマンドが実行されてしまう危険性がある．任意のコマンドを実行することにより，そのホストを破壊したり，さらなる攻撃の踏み台として利用したりすることが可能となる．このような脆弱性を利用し，任意のコマンドを実行するためにはある程度の専門知識が必要である．しかし，専門知識がない者でもインターネットで公開されている実証コードを利用することにより，容易に脆弱性を利用することができる．また，Metasploit Framework[6] のような脆弱性検査のためのツールを悪用し，遠隔でバッファオーバフローを引き起こし，任意のコマンドを実行することも可能である．このように，OS Fingerprinting は本格的な攻撃の前兆となることが多く，ネットワークに接続するホストのセキュリティに深刻な影響を与える．ネットワークに接続するホストのセキュリティを確保するためには，OS Fingerprinting の対策を検討する必要がある．

2.2 OS Fingerprinting の手法

OS を推定する手法には，Application Banner を用いる手法と OS Fingerprint を用いる手法がある．本論文では OS Fingerprint を用いる手法に着目し，その対策手法を検討する．OS Fingerprint を用いる手法はさらに Active 方式と Passive 方式に分類される．

2.2.1 Active 方式

Active 方式は能動的に検査パケットを送信し，OS Fingerprint を収集する手法である．OS Fingerprint にはパケットに含まれるヘッダ情報だけではなく，TCP/IP スタックの挙動も含まれる．たとえば，あるパケットを送信した場合に反応を返す OS と返さない OS があれば，両者を区別することができる．また，応答のパケットの内容やタイミングが異なる場合にも，それらを区別することができる．文献 2) や文献 7) には主として Active 方式の具体的な OS Fingerprinting 手法が示されている．Active 方式では，対象ホストでパケットフィルタリングを実施している場合には，検査パケットに対する応答が得られず，OS の推定に失敗する場合もある．また，能動的に検査パケットを送信するため，ネットワーク上で監視対象とするホストのパケットを傍受できる位置で実施する必要がない．代表的な Active 方式のアプリケーションには，nmap[4] 等がある．

2.2.2 Passive 方式

Passive 方式は検査パケットは送信せず，通信中のパケットを受動的に傍受して OS Fingerprint を収集する手法である．収集した OS Fingerprint を，既知の OS Fingerprint と比較分析する手順は Active 方式と同様である．ただし，Passive 方式では受動的にネットワークを監視して得られる情報しか利用できないため，収集できる OS Fingerprint の種類は Active 方式と比べると限られる．また，Passive 方式はネットワーク上で監視対象とするホストのパケットを傍受できる位置で実施する必要がある．**表 1** に TCP 通信における，SYN パケットの OS Fingerprint の例を示す．SYN パケットは，TCP 通信においてセッション確立のため，最初の接続要求として送信されるパケットである．このように TCP 通信における最初の SYN パケットのヘッダ各部の値は，OS ごとに固有の値に定まっており，これにより OS を推定することが可能である．SYN パケットの応答として返される SYN ＋ ACK パケットのヘッダ各部の値も同様である．Windows，Linux および OpenBSD[8] の ssh 接続を，代表的な Passive 方式のアプリケーションである p0f[5] でスキャンした結果を **図 1** に示す．このように，悪意ある第三者は受動的にネットワークを監視し，ネットワークに接続するホストの OS を推定し，脆弱性を知ることができる．

2.2.3 OS Fingerprinting の応用

OS Fingerprinting を応用した様々なネットワークの分析法が存在する．たとえば，IP ヘッダの TTL を用いてネットワークトポロジを推定することができる．TTL はパケットの生存時間を意味し，ルータを経由するごとに 1 ずつ減算される．TTL の初期値が既知で

表 1　SYN パケットの OS Fingerprint の例
Table 1　Examples of SYN packet's OS Fingerprints.

Header	Field	OS					
		Linux	Free BSD	Open BSD	Solaris	AIX	Windows XP
IP	Total Length	60	64	64	52	44	48
	Type of Service	0x00	0x00	0x00	0x00	0x00	0x00
	Identification	1 ずつ増加※	1 ずつ増加	ランダム	1 ずつ増加	1 ずつ増加	1 ずつ増加
	Time to Live	64	64	64	64	60	128
TCP	Window	5840	65535	16384	49640	65535	64240

※　セッションごと

```
# ./p0f
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth0', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.
192.168.5.2:3146 - Windows XP SP1+, 2000 SP3
 -> 192.168.4.1:22 (distance 1, link: ethernet/modem)
192.168.5.3:3505 - Windows XP SP1+, 2000 SP3
 -> 192.168.4.1:22 (distance 1, link: ethernet/modem)
192.168.5.4:32777 - Linux 2.4-2.6 (up: 0 hrs)
 -> 192.168.4.1:22 (distance 1, link: ethernet/modem)
192.168.5.5:4504 - OpenBSD 3.0-3.9 (up: 2741 hrs)
 -> 192.168.4.1:22 (distance 1, link: ethernet/modem)
```

図 1　p0f のスキャン結果
Fig. 1　A result of p0f scanning.

あれば，傍受したパケットの発信元ホストまでのホップ数を算出することができる．多くのホストのホップ数を知ることができれば，ネットワークトポロジを推定することができる．また，IPid を利用した分析法もある．IPid はパケットのフラグメントが発生した場合に，元のパケットを復元するための識別子として利用されている．IPid は多くの OS で 1 ずつ増加するように実装されている．文献 9)，10) ではこの性質を利用して，IPid からホスト台数を検出する手法が提案されている．IPid の OS ごとの規則性とホスト台数の検出手法

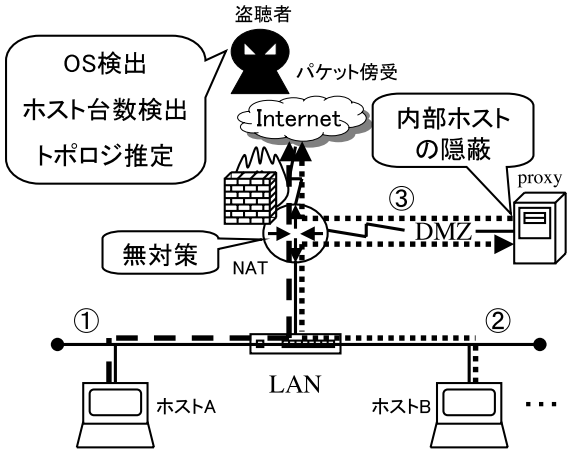


図 2　proxy の機能
Fig. 2　A function of a proxy.

の詳細については 3.2.2 項で述べる．

2.3　OS Fingerprinting の対策

LAN 内のホストの代わりに外部にアクセスする proxy や SOCKS[11] を用いることで，LAN 内の情報を隠蔽することができる．この仕組みを**図 2** を用いて説明する．ホスト B は DMZ（DeMilitarized Zone）に設置された proxy を経由して外部と通信を行う．proxy を経由した場合，LAN 内のクライアントからの接続（②）は，proxy からの接続（③）に置き換えられる．この通信は，外部の盗聴者には proxy からの接続に見えるため，LAN 内部を隠蔽することができる．このため，LAN 内のホストの OS Fingerprint は流出しない．しかし，proxy や SOCKS には対応していないアプリケーションもある．そのようなアプリケーションの外部への接続は，ホスト A のように直接行われる（①）．このため，外部の盗聴者に OS Fingerprint が流出してしまう．proxy を利用せずに LAN 内の構造を隠蔽する手法として，経路上のルータで NAT（Network Address Translator）や NAPT[12] を実施することが考えられる．しかし，NAT や NAPT では送信元 IP アドレスや送信元ポートは変換されるが，OS Fingerprint はそのまま流出してしまう．

2.3.1　Active 方式

Active 方式では様々な検査パケットを送信するため，IDS（Intrusion Detection System）

等で比較的容易に検知することが可能である．また，ファイアウォールを用いて不要なパケットを遮断し，検査パケットに対する応答を返さないようにすることで，OS 推定行為をある程度は防ぐことができる．ファイアウォールを用いたパケットの遮断は，個々のホストにおいても経路上のルータにおいても実施することができる．このように，Active 方式に対してはある程度の対策を講じることができる．

2.3.2 Passive 方式

Passive 方式はパケットを送信しないため，IDS 等で検知することは難しい．一般に Passive 方式の OS Fingerprinting は検知することが難しく，悪意ある第三者に多くの情報を与えてしまう．また，Passive 方式は受動的に行われるため，Active 方式のようにファイアウォールを用いて対策することも難しい．個々のホストにおける OS Fingerprinting 対策手法としては，IP Personality [14] のようなツールを利用し，発信するパケットのヘッダの値を変更することが対策として考えられる．

2.4 関 連 研 究

文献 13) では本来の OS とは異なる OS Fingerprint を返答する機構が提案されている．この方式では個々のホストごとに対策を実施する必要があり，対策を施すべきホスト数が多い場合には実現は難しい．また，カーネルの TCP/IP スタックを書き換える必要があるため，ソースが公開されていない商用の OS や，ネットワーク専用機器等への適用は困難である．このような個々のホストごとの対策手法は，ネットワークに接続するあらゆるホストに適用できる手法ではない．仮にすべてのホストに対策を適用したとしても，TTL や IPid によりネットワークの情報が分析される可能性がある．

文献 15)，16) では Active 方式の OS Fingerprinting に対する対策が検討され，ネットワークの経路上でパケットを検査し，変換する Protocol Scrubber が提案されている．しかしながら，文献 15)，16) では対策効果の確認を nmap でのみ実施しており，Passive 方式の OS Fingerprinting に対する効果は明らかにされていない．また，パフォーマンスへの影響を考慮し，初期ウィンドウサイズや TCP オプションへの対策は不十分なものとなっている．

文献 16)，17) では文献 15) の手法を基に，悪意ある第三者により IDS の検出を回避される問題への対策を検討しており，異常なパケットやプロトコル仕様に従わないトラフィックを正規化する手法が提案されている．トラフィックを正規化する目的は，IDS にトラフィックを正しく評価させることにある．たとえば，悪意ある第三者は TTL を非常に短く設定したパケットを利用して IDS の通過を試みる．TTL はルータを経由するごとに 1 ずつ減算される．TTL が 0 になると，悪意ある第三者とパケットの宛先の間にあるルータはこれらのパケットを破棄する．悪意ある第三者は TTL を長く設定したパケットを，破棄されるパケットの再送に見せかけて送信する．すると，IDS で評価されるトラフィックの内容と，宛先に到達するトラフィックの内容を異なるものにすることが可能となり，IDS の検出を回避することができる．この問題は，先に述べた Application Banner を用いる OS Fingerprinting 手法の対策を検討する場合には解決する必要があるが，本論文の検討範囲を越える．

文献 18) も同様にトラフィックの正規化を目的としており，Passive 方式の OS Fingerprinting 手法を利用し，パケットの送信元ホストの OS 推定結果をデータベースに蓄積するシステムが提案されている．Passive 方式では，機械学習を OS の推定に応用する試みもある [19]．送信元ホストの OS 推定結果は，フラグメントパケットの再構築や重複する TCP セグメントを除去するための手がかりとなる．また，文献 18) では TCP オプションを OS の推定に利用しており，異なる OS から 223 種類ものパターンが得られたことが示されている．これは，TCP オプションが OS を推定するための非常に有力な情報となることを示している．

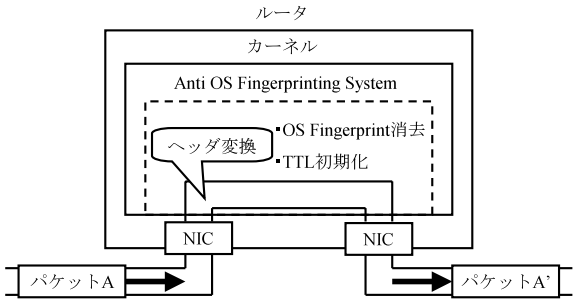
文献 20) では OS Fingerprinting に対するセキュリティ製品の開発に向け，ファイアウォール等の防御機器に対する OS Fingerprinting の効果が定量的に評価されている．文献 20) では初期ウィンドウサイズや TCP オプションから OS を推定するための最も有力な情報が得られることが示されているが，検証実験における PF [21] を用いた防御機器（以下 Network Scrubber）では十分な対策を実施していない．文献 22) では検査パケットの送信を最小限にしつつ，正確な OS Fingerprinting を行う手法が検討されている．ここでも同様に，初期ウィンドウサイズと TCP オプションが OS を推定するための最も有力な情報となることが示されている．したがって，Protocol Scrubber [15),16)] や Network Scrubber [20] では，OS Fingerprinting 対策は十分ではないものと考えられる．

3. Anti OS Fingerprinting システム（AOFS）

この章ではネットワークの経路上で OS Fingerprint を消去する Anti OS Fingerprinting システム（AOFS）を提案し，実装した AOFS の動作を説明する．

3.1 提 案 手 法

OS Fingerprinting を無力化するために，経路上でパケットの OS Fingerprint を消去し，TTL を初期化する AOFS を提案する．AOFS により悪意ある第三者に与える情報を少しでも減らすことで，ネットワークの安全性を高めることができると考えられる．AOFS の



NIC:　Network Interface Card

図 3　Anti OS Fingerprinting システムの概要
Fig. 3　An outline of the Anti OS Fingerprinting System.

概要を**図 3** に示す．ルータ上に設置した AOFS は，内側からのすべてのパケットを対象にヘッダを変換して外側へ転送する．文献 15)，16) では Active 方式の OS Fingerprinting 対策を検討しており，主に外側から内側へのパケットを対象に変換を実施している．我々は Passive 方式の OS Fingerprinting に着目し，内側から外側へのパケットを変換することを考えた．我々の提案手法では外側から内側へのパケットの変換は実施しない．AOFS の設置場所は境界ルータに限定されず，経路上のどのルータにも設置することが可能である．したがって，LAN 内においても必要に応じて AOFS を経由させることにより，LAN 内の盗聴者に対する対策をとることもできる．

3.2 実　　装

ルータの TCP/IP スタックにおけるパケット転送処理部分を改造することにより，OS Fingerprint の消去および TTL の初期化を行う．実装は Linux カーネル 2.6.18[23] を書き換えることによって実施した．カーネルの変更箇所を**図 4** に示す．転送されるパケットは NIC（Network Interface Card）から読み込まれると，ip_forward 関数によって処理される．新たに header_convert 関数を作成し，これを ip_forward 関数から呼び出すことにより，IP ヘッダおよび TCP ヘッダの変換を行う．変換処理終了後，チェックサムを再計算してパケットを転送する．以下，その細部手順について述べる．

3.2.1 TOS フラグの変換

TOS（Type of Service）はパケットがルータで処理される際の優先順位を示す情報であるが，IPv4 ではほとんど利用されていない．TCP 通信における SYN パケットの TOS の値は OS ごとに異なる場合があり，推定に利用されている．よって OS ごとの差異をなくすため，TOS を 0 に変換する．

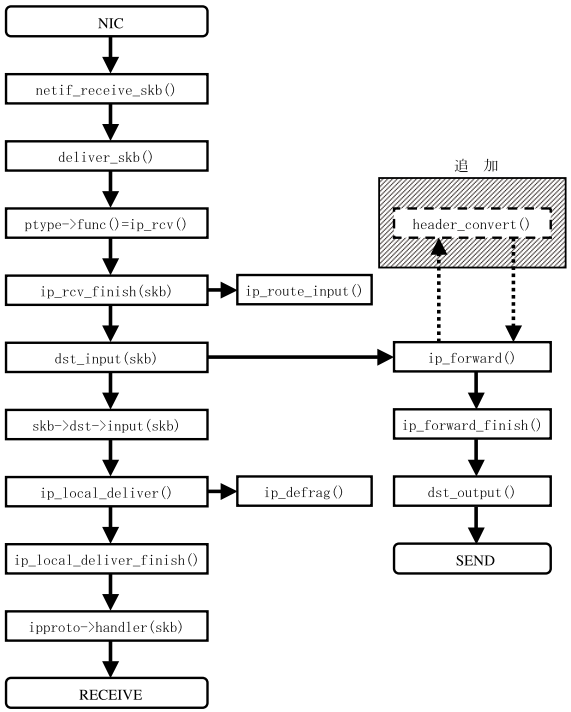


図 4　実装
Fig. 4　The Implementation.

3.2.2 IPid の変換

IPid はフラグメントパケットを再構築するための識別子として利用されている．パケットを分割すると，分割された複数のフラグメントパケット群の IPid は元のパケットと同じ値となる．IPid の値の変化は OS ごとに規則性がある．IPid は Linux ではセッションごとに 1 ずつ増加し，OpenBSD ではつねにランダムに変化する．しかし，多くの OS では IPid は 1 ずつ増加するように実装されている．そのため，同じ送信元 IP アドレスの 1 ずつ増加する IPid の値を順番にプロットして得られる直線を観測することで，NAT の内側で稼

表 2　TCP オプションの例
Table 2　Examples of TCP options.

OS	TCP Options
Linux (Fedora)	MSS, Timestamp, sackOK, wscale2, 1nop
Linux (Vine Linux)	MSS
FreeBSD	MSS, Timestamp, sackOK, wscale1, 3nop, EOL
OpenBSD	MSS, Timestamp, sackOK, wscale0, 5nop
Solaris	MSS, 3nop, wscale0, sackOK
AIX	MSS
WindowsXP	MSS, sackOK, 2nop

MSS:　Maximum Segment Size
sack:　Selective Acknowledgement
wscale:　Window Scale
nop:　No Operation
EOL:　End of List

動するホストの台数を検出することが可能である[9),10)]．そこで，この観測を無力化するために，IPid をランダムな値に変換する．IPid はフラグメントが発生しない場合には参照する必要がないため，ランダムに変化しても通信への影響はないものと考えられる．Network Scrubber[20)] でも IPid をランダムに変換する手法が用いられている．ただし，パケットが AOFS に到達する前にフラグメントが発生する場合には，同じ IPid であるフラグメントパケット群を，すべて同じ IPid に変換する必要がある．すなわち，フラグメント化されたパケットの IPid の変換にはランダムオラクルな性質が求められる．したがって，IPid の変換にはハッシュ関数を利用した．

3.2.3　TCP オプションとパケット長の変換

TCP 通信における SYN パケットに含まれる TCP オプションは OS ごとに異なり，これも OS Fingerprint として OS 推定に利用されている．**表 2** に OS ごとの TCP オプションの例を示す．

TCP オプションは必須ではないが，パフォーマンスに影響を与えるものが多い．たとえば，ウィンドウスケールオプションは TCP ヘッダのウィンドウサイズ（Window）を拡張するために利用される．ウィンドウスケールオプションに値 X を指定すると，ウィンドウサイズは 2 の X 乗倍の値と見なされる．表 2 では MSS オプションが各 OS に共通しているため，すべての OS の TCP オプションを MSS のみに変換すれば，これらの OS を区別することはできなくなるものと考えられる．しかしながら，古い OS には TCP オプションをサポートしていないものもある．これらの OS も考慮し，AOFS の内側で動作する OS の種類を隠蔽するためには，すべての TCP オプションを消去せざるをえない．したがって，すべての TCP オプションを消去する．パケット長（Total Length）やオフセット（Data Offset）についても，オプションの変換に合わせて変換する．

3.2.4　TTL の初期化

TTL は OS ごとに初期値が決まっており，これも OS Fingerprint として OS の推定に利用されている．TTL はルータを経由するごとに 1 ずつ減算される．この性質を利用して，パケットの送信元ホストまでのホップ数を算出することもできる．このため，送信元ごとのホップ数を収集することで，ネットワークトポロジを推定することができてしまう．これらを無力化するため，AOFS では TTL を一定値に初期化する．文献 16)，17) では IDS の検出を回避される問題への対策として，小さい値の TTL を増加させる手法が提案されている．TTL を一定値に初期化することで OS 推定を妨げるとともに，この問題にも対処することができる．

3.2.5　ウィンドウサイズの変換

SYN パケットに含まれる初期ウィンドウサイズが OS の推定に利用されるため，一定値に変換する必要がある．ウィンドウサイズは TCP 通信において，許容できるバッファの大きさを通知するために利用される．ウィンドウサイズを経路の途中で増加させた場合には，許容量を超えるバッファの大きさを通知することになる．この場合にはバッファから溢れたパケットが破棄され，通信に支障が生じる可能性がある．ウィンドウサイズを経路の途中で減少させた場合には，スループットは低下するものの，通信に支障はないものと考えられる．したがって，初期ウィンドウサイズを最も値が小さい OS に合わせて変換する．表 1 では Linux の初期ウィンドウサイズの値が最も小さいが，ウィンドウスケールオプションにも注意する必要がある．表 2 では Fedora[24)] はウィンドウスケールオプションが 2 であるため，ウィンドウサイズは 23,360 と見なされる．Vine Linux[25)] ではウィンドウスケールオプションは指定されていないため，ウィンドウサイズは 5,840 となる．したがって，初期ウィンドウサイズの変換値を 5,840 に設定する．

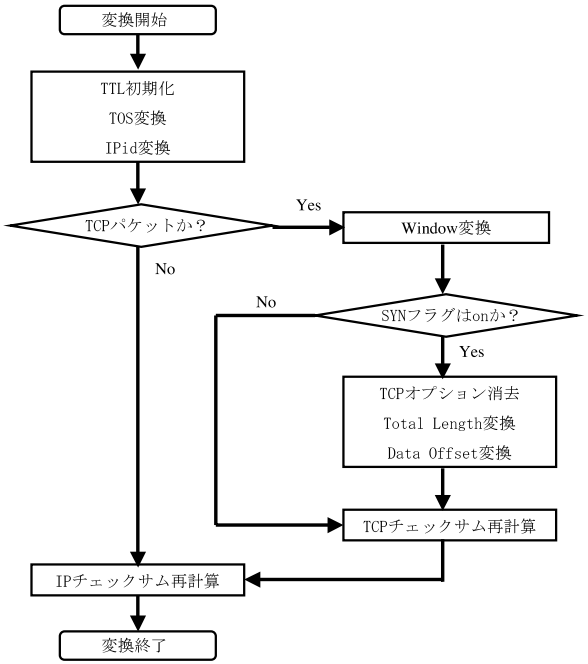


図 5　変換アルゴリズム
Fig. 5　The converting algorithm.

3.3　変換アルゴリズム

変換アルゴリズムを**図 5** に示す．AOFS の内側から外側へ中継されるすべてのパケットを対象にこの変換処理を実施する．これにより，AOFS の外側への OS Fingerprint の流出を防ぐ．IP ヘッダおよび TCP ヘッダの変換箇所は**図 6** および**図 7** のとおりである．

3.4　IPv6 への適用

IPv6 ネットワークでは IPv4 ヘッダ（図 6）の代わりに IPv6 ヘッダがパケットに付加される．ヘッダに含まれる情報のうち，Hop Limit は IPv4 ヘッダにおける Time To Live に相当し，ルータを経由するごとに 1 ずつ減算される．したがって，IPv4 ネットワークと同様に LAN 内部のトポロジを推定される可能性がある．また，初期値も OS ごとに決まっているため，OS 推定にも利用することができる．よって Hop Limit を同一の値に初期化する．同様に Traffic Class や Flow Label についても同一の値に変換する．IPv6 ヘッダの変換箇所は**図 8** のとおりである．このように，提案手法は IPv6 ネットワークにおいても適用することができる．しかし，IPv6 では拡張ヘッダにおいても OS ごとの固有値が特徴として現れる可能性があるため，変換箇所についてはさらに調査する必要がある．また，IPv6 ネットワークにおいて EUI64[27] 方式により IP アドレスを自動的に割り当てる場合には，送信元 IP アドレスはネットワークインタフェースの物理アドレスをもとにして生成される．よって，IP アドレスから物理アドレスに関する情報がセグメント外に流出する可能性もある．

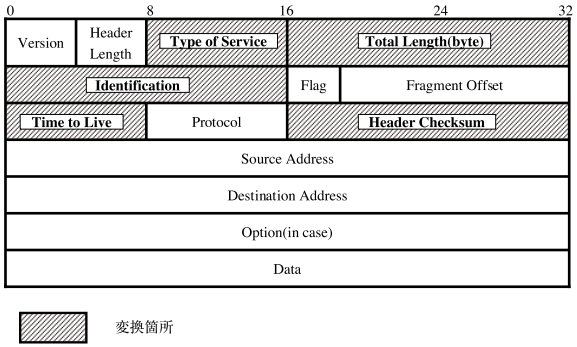


図 6　IPv4 ヘッダの変換箇所
Fig. 6　Converted fields in IPv4 header.

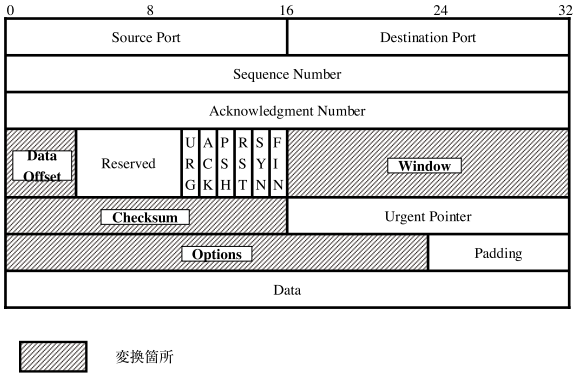


図 7　TCP ヘッダの変換箇所
Fig. 7　Converted fields in TCP header.

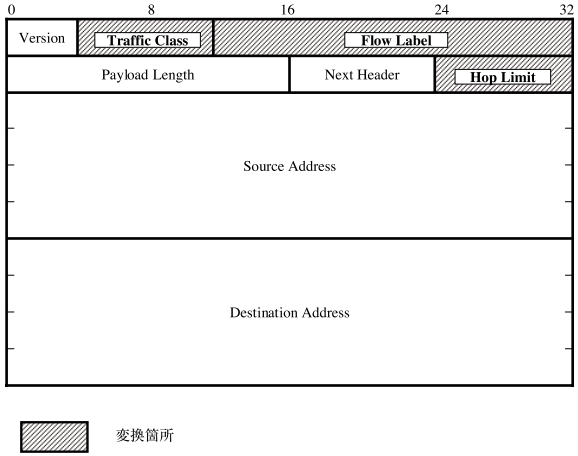


図 8　IPv6 ヘッダの変換箇所
Fig. 8　Converted fields in IPv6 header.

4. 検 証 実 験

この章では実装した AOFS について検証実験を行い，OS Fingerprinting 対策の効果を確認する．

4.1 実 験 環 境

検証実験における AOFS の配置と実験ネットワークの構成を**図 9** に示す．各ホストは 100BASE/T イーサネットで接続されており，図中に示す OS が動作する．各 OS の OS Fingerprint は，表 1 および表 2 に示した内容と一致する．図中のルータは 2 つのインタフェースを備えた PC であり，Linux（Fedora）が動作している．実験ではこの PC を通常のルータと呼び，通常のルータのカーネルを提案手法に従って変更したものを AOFS と呼ぶ．したがって，通常のルータと AOFS の差異はカーネルの変更箇所のみとなる．また，図中のルータより上方をルータの外側，下方を内側とする．ルータの外側のホスト A で，ルータの内側に設置された各ホストからのパケットを観測する．

4.2 アプリケーションの動作確認

AOFS の内側のホスト B，C，D，E の各アプリケーションを用い，AOFS の外側と通信確認を行った結果を**表 3** に示す．traceroute 以外のアプリケーションについては通信に

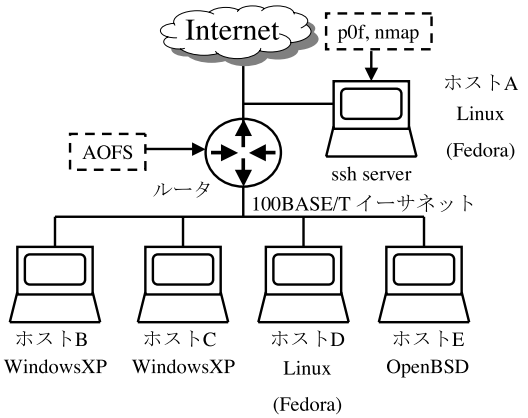


図 9　実験ネットワーク
Fig. 9　The network for the experiment.

表 3　通信確認結果
Table 3　Results of communication check.

Application	Protocol	Result
ping	ICMP	○
traceroute	ICMP/UDP	×
nslookup	UDP/TCP	○
ssh	TCP	○
telnet	TCP	○
smtp	TCP	○
http	TCP	○
pop3	TCP	○
skype	TCP (P2P)	○

支障はなかった．traceroute は指定した宛先までの経路上のルータを検出し，ネットワークの経路を調査するアプリケーションである．traceroute に支障が生じたのは，traceroute は TTL を利用してホップ数の計測を行うためである．AOFS によって TTL が初期化されるため，正しくホップ数を計測することができず，ルータの検出に失敗する．このように，AOFS は IP ヘッダや TCP ヘッダの情報を利用するアプリケーションには影響がある．このようなアプリケーションを利用する場合には，別の経路を利用したり，変換を行わない等の対策を個別に実施する必要がある．

```
# ./p0f
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth0', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.
192.168.5.2:1032 - UNKNOWN [5840:64:1:40:.:.:?:?]
 -> 192.168.4.1 :22 (link: unspecified)
192.168.5.3:1055 - UNKNOWN [5840:64:1:40:.:.:?:?]
 -> 192.168.4.1 :22 (link: unspecified)
192.168.5.4:32774 - UNKNOWN [5840:64:1:40:.:.:?:?]
 -> 192.168.4.1 :22 (link: unspecified)
192.168.5.5:21944 - UNKNOWN [5840:64:1:40:.:.:?:?]
 -> 192.168.4.1 :22 (link: unspecified)
```

図 10　p0f のスキャン結果 2
Fig. 10　A result of p0f scanning 2.

```
# ./nmap -O 192.168.5.4
Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2007-03-09 08:37 JST
Interesting ports on 192.168.5.4:
(The 1666 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
22/tcp  open  ssh
111/tcp open  rpcbind
Device type: general purpose
Running: Linux 2.4.XI2.5.XI2.6.X
OS details: Linux 2.4.0 - 2.5.20, Linux 2.4.7 - 2.6.11

Nmap finished: 1 IP address (1 host up) scanned in 2.320 seconds
```

図 11　nmap のスキャン結果
Fig. 11　A result of nmap scanning.

4.3　p0f による OS 推定

ホスト B，C，D，E から AOFS を経由したホスト A への ssh 接続を，p0f でスキャンした結果を**図 10** に示す．図 1 と比較すると，AOFS によって OS Fingerprint は消去されており，p0f は OS 推定に失敗していることが確認できる．また，TTL も初期化されているため，ホップ数の算出にも失敗している．AOFS は有効に機能しており，Passive 方式の

```
# ./nmap -O 192.168.5.4
Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2007-03-09 08:34 JST
Insufficient responses for TCP sequencing (1), OS detection may be less accurate
Insufficient responses for TCP sequencing (1), OS detection may be less accurate
Insufficient responses for TCP sequencing (1), OS detection may be less accurate
Interesting ports on 192.168.5.4:
(The 1666 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
22/tcp  open  ssh
111/tcp open  rpcbind
No OS matches for host (If you know what OS is running on it, see
http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=3.93%P=i686-pc-linux-gnu%D=3/9%Tm=45F09D87%O=22%C=1)
T1(Resp=Y%DF=Y%W=05B4%ACK=S++%Flags=AS%Ops=)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=05B4%ACK=S++%Flags=AS%Ops=)
T4(Resp=Y%DF=Y%W=05B4%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=05B4%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=05B4%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=05B4%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E
%UCK=E%ULEN=134%DAT=E)

Nmap finished: 1 IP address (1 host up) scanned in 13.001 seconds
```

図 12　nmap のスキャン結果 2
Fig. 12　A result of nmap scanning 2.

OS Fingerprinting 対策としてある程度の効果があることが確認できた．

4.4　nmap による OS 推定

AOFS の外側のホスト A から内側のホスト D に対し，nmap の OS 推定オプションを有効にしてスキャンした結果を**図 12** に示す．AOFS が有効となっていない**図 11** と比較すると，AOFS によって OS Fingerprint は消去されており，nmap は OS 推定に失敗していることが確認できる．これは，外側からの検査パケットはそのまま AOFS を通過するが，内側からの SYN ＋ ACK パケット等の応答パケットが AOFS によって変換されたためであ

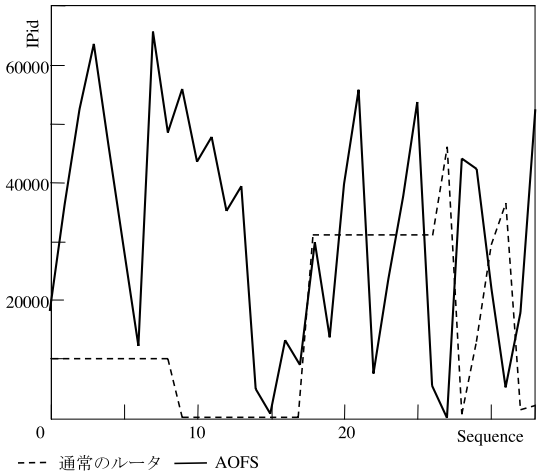


図 13　IPid 観測結果
Fig. 13　Results of the IPid observation.

ると考えられる．ホスト B，C および E に対しても nmap によるスキャンを実施し，同様に OS 推定に失敗することが確認できた．AOFS は Active 方式の OS Fingerprinting 対策としてもある程度の効果があることが確認できた．

4.5　IPid の観測

通常のルータと AOFS を経由した通信の IPid の観測結果を**図 13** に示す．図中の縦軸は傍受したパケットの IPid の値を示し，横軸はパケットの到着順序を示す．通常のルータを経由した通信では IPid が 1 ずつ増加する直線が 3 本観測されており，OS Fingerprint と合わせて分析することにより，ルータの内側で 4 台のホストが稼動していることが検出できる．AOFS を経由した場合には IPid はランダムとなり，AOFS 内側で稼動するホスト数を検出することはできない．

4.6　通 信 遅 延

パケットジェネレータである hping[28] を使用し，通常のルータと AOFS を経由した通信の RTT（Round Trip Time）を計測する．ルータの内側のホストからホスト A に対し，TCP SYN パケットを 10 パケット送信し，各応答を得るまでの所要時間を 3 回計測した．RTT の計測結果を**表 4** に示す．通常のルータを経由した通信と AOFS を経由した通信の RTT にはほとんど差はなく，通信遅延はほとんど発生していないことが確認できた．これ

表 4　RTT 計測結果
Table 4　Results of the RTT measurement.

	通常のルータ			AOFS		
Count	1	2	3	1	2	3
1	0.4	0.5	0.5	0.5	0.5	0.7
2	0.4	0.4	0.5	0.6	0.4	0.4
3	0.5	0.4	0.5	0.4	0.4	0.4
4	0.4	0.6	0.4	0.4	0.5	0.5
5	0.5	0.5	0.5	0.5	0.7	0.5
6	0.4	0.7	0.5	0.4	0.5	0.7
7	0.4	0.4	0.4	0.4	0.5	0.5
8	0.4	0.4	0.5	0.5	0.6	0.5
9	0.5	0.5	0.4	0.4	0.6	0.4
10	0.4	0.5	0.6	0.4	0.5	0.5
Average	0.43	0.49	0.48	0.45	0.52	0.51

ms

はプロセッサによる TCP/IP スタックの処理速度と，イーサネットカードによるパケットの処理速度に大きな差があるためであると考えられる．

4.7　処 理 能 力

通常のルータと AOFS の処理能力を比較するため，scp（Secure CoPy）コマンドで 1 GB のファイルを転送し，転送量 100 MB ごとのスループットを計測する．

4.7.1　内側から外側への処理能力

まず，ルータの内側から外側へのファイル転送時のスループットを計測する．ルータの内側のホスト D および E から外側のホスト A に対し，scp コマンドで 1 GB のファイルを転送した．スループットの計測結果を**表 5** に示す．通常のルータと AOFS のスループットにはほとんど差はなく，処理能力はほとんど低下していないことを観測した．このことから，AOFS におけるパケットの変換処理によるオーバヘッドはほとんど発生していないことが確認できる．ホスト D から A およびホスト E から A のスループットに差が出た原因は，OS がサポートする TCP オプションや初期ウィンドウサイズの違いであると考えられる．初期ウィンドウサイズの大きさは，パケットを受信するためのバッファの大きさを意味する．OS によってパケットを送受信するためのバッファの大きさが異なるため，スループットに差が生じたものと考えられる．

4.7.2　外側から内側への処理能力

次に，ルータの外側から内側へのファイル転送時のスループットを計測する．ルータの

表 5　スループット計測結果 1
Table 5　Results of the throughput measurement 1.

	通常のルータ		AOFS	
	D から A	E から A	D から A	E から A
0–100 MB	16.7	10.0	16.7	11.1
100–200 MB	14.3	11.1	16.7	11.1
200–300 MB	14.3	12.5	16.7	12.5
300–400 MB	16.7	11.1	14.3	11.1
400–500 MB	14.3	11.1	16.7	10.0
500–600 MB	14.3	11.1	16.7	11.1
600–700 MB	14.3	10.0	14.3	11.1
700–800 MB	16.7	10.0	14.3	11.1
800–900 MB	16.7	10.0	14.3	10.0
900 MB–1 GB	14.3	11.1	16.7	12.5
Average	15.26	10.8	15.74	11.16

MB/s

表 6　スループット計測結果 2
Table 6　Results of the throughput measurement 2.

	通常のルータ		AOFS	
	A から D	A から E	A から D	A から E
0–100 MB	14.3	11.1	5.9	5.6
100–200 MB	14.3	11.1	5.6	5.9
200–300 MB	16.7	12.5	5.9	5.9
300–400 MB	14.3	12.5	5.9	5.9
400–500 MB	16.7	12.5	5.6	5.9
500–600 MB	16.7	12.5	5.9	5.9
600–700 MB	16.7	12.5	5.6	5.9
700–800 MB	16.7	12.5	6.3	5.9
800–900 MB	16.7	12.5	5.6	5.9
900 MB–1 GB	16.7	11.1	5.6	5.9
Average	15.98	12.08	5.79	5.87

MB/s

外側のホスト A から内側のホスト D およびホスト E に対し，scp コマンドで 1 GB のファイルを転送した．スループットの計測結果を**表 6** に示す．表 5 と表 6 の AOFS のスループットに着目すると，全般的に低下していることが観測できる．AOFS では外側から内側へのパケットは変換しないため，この原因はパケットの変換処理によるオーバヘッドではない．TCP オプションを消去したことと，初期ウィンドウサイズを変換したことが原因として考えられる．ホスト D の OS は Linux（Fedora）であり，表 2 によるとウィンドウスケールオプションに 2 が設定されている．よって，表 1 から実質的な初期ウィンドウサイズは 23,360 となる．ホスト E の OS は OpenBSD であり，初期ウィンドウサイズは 16,384 である．しかし，AOFS の外側から内側へのファイル転送時には，ウィンドウスケールオプションの消去と初期ウィンドウサイズの変換により，初期ウィンドウサイズは 5,840 に減少される．したがって，表 5 の D から A のスループットに対し，表 6 の A から D のスループットが低下した原因は，ウィンドウスケールオプションが消去されたためであると考えられる．また，表 5 の E から A のスループットに対し，表 6 の A から E のスループットが低下した原因は，初期ウィンドウサイズを減少させたことであると考えられる．TCP オプションや初期ウィンドウサイズの変換は，AOFS の外側から内側へのファイル転送時の処理能力に大きく影響することが確認できた．

実験結果から，AOFS を設置することでセキュリティの向上は期待できるが，外側から内側へのスループットが低下することが分かった．したがって，AOFS を設置する場合には要求されるスループットを考慮し，外側から内側へのスループットの低下を許容できるかどうかを検討する必要がある．

5. 考　　察

この章では検証実験の結果をまとめ，AOFS と従来の OS Fingerprinting 対策手法を比較し，AOFS の有効範囲について考察する．

本論文では主に Passive 方式の OS Fingerprinting 手法に着目し，内側から外側へのパケットを変換することにより OS Fingerprinting 対策を試みた．検証実験から得られた AOFS の有効性を，Protocol Scrubber [15),16)] や Network Scrubber [20)] と比較した結果を**表 7** にまとめる．表中の○は対応する OS Fingerprinting 手法に対して対策を実施していることを示す．△は対策が不十分であり，×は対策を実施していないことを示している．AOFS では簡易なパケットの変換のみ実施し，主として Active 方式と Passive 方式に共通で利用される OS Fingerprint を消去することに努めた．Network Scrubber では IPid の変換を実施しているが，フラグメントパケットの再構築は考慮されていない．AOFS では変換にハッシュ関数を利用することにより，フラグメントパケットの再構築も考慮した．Protocol Scrubber ではパフォーマンスへの影響を考慮し，TCP オプションへの対策は順番を変えるのみとなっている．Network Scrubber では MSS オプションと Timestamp オプションの値を変える処置がとられている．しかしながら表 2 の例では，TCP オプションの順番を

表 7　AOFS の有効性
Table 7　Effectiveness of AOFS.

OS Fingerprinting 手法	AOFS	Protocol Scrubber	Network Scrubber
TOS フラグの違い	○	○	×
IPid の変化の規則性	○	×	△
TCP オプションの順序と違い	○	△	△
TTL の違い	○	×	○
初期ウィンドウサイズの違い	○	×	×
初期シーケンス番号と変化の規則性	×	○	×
様々な TCP パケットに対する応答の有無	×	○	○
ICMP エラーメッセージの内容とタイミング	×	○	○

変えても OS ごとの違いをなくすことはできない．したがって，TCP オプションへの対策は不十分であると考えられる．また，Protocol Scrubber や Network Scrubber では，初期ウィンドウサイズについても対策を実施していない．初期ウィンドウサイズと TCP オプションが OS を推定するための最も有力な情報となること[20),22)] を考慮すると，従来の OS Fingerprinting 対策手法では OS が推定される可能性が高いといえる．AOFS ではすべての TCP オプションを消去し，初期ウィンドウサイズも一定値に変換しているため，これらの情報から OS を推定することはできない．したがって，AOFS では OS が推定される可能性を下げることができたものと考えられる．また，検証実験により TCP オプションや初期ウィンドウサイズの変換がパフォーマンスに与える影響を明らかにした．

AOFS では TCP シーケンス番号への対策を実施していないため，その点では従来手法には劣る．シーケンス番号はランダムに変化するが，OS ごとに使用している擬似乱数生成アルゴリズムが異なるため，初期値や変化の規則性を分析することにより OS を推定される可能性がある．シーケンス番号を変換するためには，シーケンス番号と応答番号の対応関係を AOFS 内に保持する必要がある．また，AOFS は nmap による OS 推定を妨げることはできたものの，様々なパケットを送信した場合の応答の有無により OS を区別する手法や，応答のパケットの内容やタイミングにより OS を区別する手法については対策を実施していない．これは，既存のルータへのハードウェア実装も視野に入れ，簡易なパケット変換のみで対策手法を検討したためである．これらの OS 推定手法に対しては，ファイアウォールで適切にフィルタリングを行うことで対処することができる．

AOFS や Protocol Scrubber 等のネットワークの経路上でパケットを変換する手法は，個々のホストごとの OS 等を考慮する必要がなく，大規模なネットワークでの利用に適する．しかしながら，IPsec の認証ヘッダ[26)] 等の改ざん防止処置がとられている通信は，経路の途中でパケットを変換することができない．これらの通信についても別の経路を利用したり，変換を行わなかったりする等の対策を個別に実施する必要がある．

6. おわりに

本研究ではネットワーク外部に流出した OS Fingerprint を収集し，分析することにより，内部ホストの OS 推定，ホスト台数検出やネットワークトポロジの推定が可能であることに着目した．そして，経路上のルータで OS Fingerprint を消去し，TTL を初期化することによってこれを妨げる手法を提案した．さらに，AOFS を Linux カーネルに実装し，AOFS の通信への影響，効果，通信遅延および処理能力を検証実験により確認した．本研究で提案した手法を用いれば，AOFS の内側のネットワーク情報の流出を防ぎ，攻撃にかかるコストを増大させることができる．AOFS を NAT と合わせて利用すれば，NAT ルータを単一のホストのように見せることも可能である．これにより，内部ホストが攻撃を回避できる可能性を高め，ネットワークのセキュリティを向上させることができる．本研究では，代表的な OS Fingerprinting アプリケーションである nmap と p0f を用いて対策効果を確認した．したがって，高度な知識を持たない悪意ある第三者に対しては効果が期待できる．しかし，nmap や p0f を用いた検証には限界がある．nmap や p0f の検出を無力化することにより，完全に OS Fingerprinting を無力化できたわけではない．また，Application Banner を用いる OS 推定手法についても対策を講じる必要がある．この手法は，TELNET，FTP，HTTP のリクエストや DNS クエリ等のデータ領域から特定の文字列を検出し，OS を推定する手法である．多くのアプリケーションや OS が，初期の設定では Application Banner を出力してしまう場合が多い．Application Banner を用いる手法に対しては，AOFS と同じように経路上で Application Banner を消去する手法が対策として考えられる．この手法を実現するためには，ネットワーク層で中継を行うルータではなく，アプリケーション層で中継を行う proxy 等に対策手法を実装することが望ましい．

今後の課題としては，TCP シーケンス番号やそのほかの OS Fingerprinting の対策が考えられる．TCP シーケンス番号は Active 方式だけでなく Passive 方式でも利用できるため，受動的にネットワーク内部のホストの OS を推定される可能性がある．IPv6 の拡張ヘッダについても同様に，ネットワーク内部の情報を流出させる可能性がある．これらの対策は，パケットを変換する手法で実現することが可能である．しかし，様々なパケットを送信した場合の応答の有無により OS を区別する手法や，応答のパケットの内容やタイミングに

より OS を区別する手法については別の対策を検討する必要がある．

参 考 文 献

1) 情報処理推進機構，JPCERT コーディネーションセンター：ソフトウェア等の脆弱性関連情報に関する届出状況，2007 年第 4 四半期 (2008). http://www.jpcert.or.jp/press/
2) Dostoevsky, F.: Remote OS Detection via TCP/IP Fingerprinting (2nd Generation). http://insecure.org/nmap/osdetect/
3) 情報処理推進機構セキュリティセンター：TCP/IP に係る既知の脆弱性に関する調査報告書改訂第 3 版 (2008). http://www.ipa.go.jp/security/
4) nmap. http://insecure.org/nmap/
5) p0f. http://lcamtuf.coredump.cx/
6) Metasploit Framework. http://www.metasploit.com/projects/Framework/
7) Arkin, O.: ICMP Usage in Scanning – The Complete Know How Version 3.0 (2001). http://www.sys-security.com/
8) OpenBSD. http://www.openbsd.org/
9) Bellovin, M.S.: A Technique for Counting NATted Hosts, *Proc. Internet Measurement Workshop 2002*, pp.267–272 (2002).
10) 高橋輝壮，甲斐俊文，篠原克幸：IPid を用いた NAT 検出手法の考察，情報処理学会研究報告，2006-CSEC-32, Vol.2006, No.26, pp.97–102 (2006).
11) Koblas, D. and Koblas, R.M.: SOCKS, *Proc.USENIX UNIX Security Symposium III*, pp.77–83 (1992).
12) Srisuresh, P. and Egevang, B.K.: Traditional IP Network Address Translator, RFC3022 (2001).
13) 宮本大輔，大江将史，木村泰司，門林雄基：OS Fingerprint 対策手法の実装と評価，第 4 回インターネットテクノロジーワークショップ（WIT2001）論文集，pp.17–25 (2001).
14) IP Personality. http://ippersonality.sourceforge.net/
15) Smart, M., Malan, G.R. and Jahanian, F.: Defeating TCP/IP Stack Fingerprinting, *Proc. 9th Conference on USENIX Security Symposium*, pp.229–240 (2000).
16) Watson, D., Smart, M., Malan, G.R. and Jahanian, F.: Protocol Scrubbing: Network Security Through Transparent Flow Modification, *IEEE/ACM Trans. Networking*, Vol.12, No.2, pp.261–273 (2004).
17) Handley, M., Paxson, V. and Kreibich, C.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics, *Proc. USENIX Security Symposium 2001* (2001).
18) Taleck, G.: Ambiguity Resolution via Passive OS Fingerprinting, *Proc. 6th International Symposium on Recent Advances in Intrusion Detection* (2003).
19) Beverly, R.: A Robust Classifier for Passive TCP/IP Fingerprinting, *Proc. 5th Passive and Active Measurement Workshop*, pp.158–167 (2004).
20) Greenwald, L.G. and Thomas, T.J.: Understanding and Preventing Network Device Fingerprinting, *Bell Labs. Technical Journal*, Vol.12, Issue 3, pp.149–166 (2007).
21) Hartmeier, D.: PF: The OpenBSD Packet Filter. http://www.openbsd.org/faq/pf/
22) Greenwald, L.G. and Thomas, T.J.: Toward Undetected Operating System Fingerprinting, *Proc. 1st Conference on 1st USENIX Workshop on Offensive Technologies* (2007).
23) The Linux Kernel Archives. http://www.kernel.org/
24) Fedora. http://fedoraproject.org/
25) Vine Linux. http://www.vinelinux.org/
26) Kent, S.: IP Authentication Header, RFC4302 (2005).
27) IEEE: Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority (1997). http://standards.ieee.org/
28) hping. http://www.hping.org/

(平成 19 年 12 月 10 日受付)
(平成 20 年 4 月 8 日採録)

三村　守（正会員）

1978 年生．2001 年防衛大学校情報工学科卒業．同年海上自衛隊入隊．2008 年防衛大学校理工学研究科情報数理専攻修了．同年海上自衛隊保全監査隊勤務（現職）．ネットワークセキュリティに興味を持つ．

中村　康弘（正会員）

1959 年生．1987 年防衛大学校理工学研究科オペレーションズリサーチ専攻修了．同年同大学校電気工学科助手兼共同利用電子計算機室勤務．1994 年同大学校情報工学科助教授．2005 年同大学校学術情報センター情報通信技術研究室勤務（現職）．2007 年同大学校情報工学科准教授（現職）．コンピュータネットワークセキュリティ，電子透かし，パターン認識に関する研究に従事．工学博士．電子情報通信学会，画像電子学会各会員．