Vol.2014-CSEC-65 No.6
Vol.2014-IOT-25 No.6
2014/5/22

利用者認証ネットワークにおけるNATの検出および通信の遮断

末永 光弘[1]　田中 久治[1]　大谷 誠[2]　堀 良彰[3]　岡崎 泰久[1]　渡辺 健次[4]

概要：利用者認証を行うネットワークにおいて，認証時にブロードバンドルータのようなNAT機能をもつ機器を経由すると，認証者と実際の利用者が違う場合がある．そこで，本研究では，Javaアプレットを用いてユーザ端末の実際のIPアドレスを取得することでNATを検出する手法を考案し，Webブラウザを用いて認証を行うシステムに実装した．また，Javaアプレットが動作しないユーザ端末に対しては，パケットのTTL値を検証することでNATを検出する方法を実装した．これにより，認証時にNATの有無を検出し，NATを経由する通信を遮断することが可能である．本稿では，このNAT検出手法の詳細と実装したネットワーク利用者認証システムにおける動作検証について述べる．

キーワード：ネットワーク利用者認証，NAT，NAT検出

Detection and Blocking of a host behind NAT in the User Authentication Network

Mitsuhiro Suenaga[1]　Hisaharu Tanaka[1]　Makoto Otani[2]　Yoshiaki Hori[3]　Yasuhisa Okazaki[1]
Kenzi Watanabe[4]

Abstract: In the network which requires user authentication, when hosts exist behind NAT, there is a problem that the authenticated user may differ from an actual user. In this research, we devise a method for detecting NAT by acquiring the actual IP address of the user terminal using a Java applet, and have implemented this method to the system which requires authentication through web browser. Moreover, we have implemented the method that to detect NAT by verifying the TTL value of packets for terminals without Java Applet. As a result, we are able to detect the existence of NAT during authentication, and block the communication through the NAT. In this report, we describe the details of this NAT detection method and the operation verification in the authentication system which have been implemented this method.

Keywords: Network User Authentication, NAT, NAT detection

1. はじめに

近年，部外者によるネットワークの利用やユーザによる不正な利用を防止するため，企業や大学などでネットワーク利用を開始する際に，ユーザ認証を求められることが一般的になっている．

しかし，ユーザ認証の後に，IPアドレスやMACアドレスなどのユーザ端末の識別情報に対して通信許可を与えるネットワーク利用者認証システムにおいては，NAT機能を持つ機器が存在すると，NAT機能を持つ機器に対して通信許可を与えてしまう．そのため，同一のNATを経由する通信を，すべて同一のユーザ端末による通信とみなし，最初の一台以外は認証を経ず，複数の端末やユーザによる

[1] 佐賀大学大学院工学系研究科
Graduate School of Science and Engineering, Saga University
[2] 佐賀大学総合情報基盤センター
Computer and Network Center, Saga University
[3] 佐賀大学全学教育機構
Organization for General Education, Saga University
[4] 広島大学大学院教育学研究科
Graduate School of Education, Hiroshima University

ネットワークの利用を許すことになる．これにより，認証システムはユーザごとの個別の認証や利用記録の取得が行えないという問題が発生する．

特に大学のような場所においては，ネットワークの知識が少ない利用者が無線 LAN ルータのような NAT 機能をもつ機器を設置して問題となるケースが発生している．利用者にとっては，無線 LAN ルータなどを設置することに関して，インターネットを自由な場所で利用したいといった理由が大半であり，特に悪意があるケースは少ないと考えられる．しかし，ユーザ認証により利用許可を与えるネットワークに対してこのような行為を行うことは，不正利用の調査やコンピュータウィルスの感染源の特定が困難になるなど，ネットワーク管理上好ましくない．

NAT の検出手法としては，Bellovin らの IPid を用いた通信ホスト台数の検知技術 [1] を応用した，NAT 検知手法 [2] や，検出システム側から送出したトレースパケットに対する応答パケットの TTL 値を観測することで，能動的に NAT を検出する手法 [3] などがすでに提案されている．しかし，これらの手法では，ネットワーク利用者認証システムに組み込む場合，ネットワーク利用認証時に検出することが難しい．

そこで，本研究では，NAT を経由する通信が行われた場合に，ユーザのネットワーク利用認証時に検出することを目的とし，二つの手法の実装を行った．一つは Java アプレットを用いて認証システムの把握しているユーザ端末の IP アドレスと，実際のユーザ端末の IP アドレスの比較により NAT を検出する手法である．もう一つは TTL 値の観測により，NAT を検出する手法である．この二つの手法を，Web を用いて認証を行うネットワーク利用者認証システム (以下，認証システム) に対して実装し，検証を行った．

2. NAT の検出と通信の制御

ユーザ認証後に，IP アドレスや MAC アドレスなどのユーザ端末のネットワークデバイスの識別情報に対して利用許可を行うネットワークでは，ユーザ端末と認証システムの間に NAT を設置されると，正しくユーザの管理ができない．最初の一台の認証により，認証システムは NAT に対して通信許可を出すため，二台目以降のユーザ端末は認証を経ず，ネットワークを利用できる．そこで，不適切な利用の防止のために NAT を検出し，その通信を制御する仕組みが必要となる．

NAT の存在を検出するための手法はいくつか提案されているが，本研究では，Web をブラウザを通してユーザ認証を行い，ユーザ端末に対して利用許可を行う認証システムを想定し，NAT の特性を考慮した二種類の実装を行った．

2.1 Java アプレットによる NAT の検出

通常，ネットワークに接続されるユーザ端末は，DHCP サーバにより自動的に IP アドレスを取得する．ユーザ端末の IP アドレスや MAC アドレスを用いて認証許可を行うネットワークにおいては，管理者がユーザ端末が接続するネットワーク上に NAT となる機器を設置することはない．そのため，認証システム側において，ユーザ端末の IP アドレスを取得する場合，それはユーザ端末の実際の IP アドレスと同一となるはずである．

しかし，NAT を経由して接続した場合，アドレス変換が行われるため，認証システムが把握している IP アドレスと，実際にユーザ端末に割り振られている IP アドレスが一致しない．

この二つの IP アドレスを比較することで NAT を検出することが可能であるが，そのためには，ユーザ端末に実際に割り振られている IP アドレスを調査・取得する必要がある．PHP や Javascript を用いた手法では，ユーザ端末の NIC に実際に割り振られている IP アドレスの取得は困難である．そこで本研究では，署名済み Java アプレットを使用することでそれを実現した．通常の Java アプレットではループバックアドレスしか取得できないが，署名済み Java アプレットを用いると，ユーザ端末の Web ブラウザ上で，ユーザ端末に実際に割り振られている IP アドレスを取得することができる．

ただし，Java 1.7 update51 以降は，自己署名によるコードサイニングを行った Java アプレットでは Java のセキュリティ設定によってはブロックされ，動作しないため，正式なコードサイニング証明書を取得し署名する必要がある [4]．

IP アドレスは Apache などの Web サーバの環境変数から取得でき，パラメータとして Java アプレットに渡すことが可能である．Java アプレットが取得したユーザ端末の実際の IP アドレスと，認証システムに接続している IP アドレスを比較することで，NAT を通した通信であるかを判定する．

Web による認証時に二つの IP アドレスが一致し，NAT でない場合は通常通りに通信を許可し，二つの IP アドレスが一致しない場合は通信を許可せず，警告を行う Web ページへと誘導する．

2.2 TTL の監視による NAT の検出

現状では，Java 自体がインストールされておらず，アプレットを実行できない端末も多いと考えられる．そこで，Java アプレットが実行できない場合も NAT の検出が可能なように，TTL 値の監視による NAT 検出手法も実装した．この手法は通常は Java アプレットによる検出と併せて用いるが，Java アプレットが動作しない環境の端末である場合は独立して動作する．

パケットの TTL のデフォルト値は OS ごとに決まっている．表 1 に各 OS ごとの TTL のデフォルト値 [5] を示す．

表 1 各 OS のデフォルト TTL 値
Table 1 Default TTL Value of Each OS

OS	デフォルト TTL 値
UNIX,Linux,MacOS, Android,iOS	64
Windows OS	128
Solaris	255
その他 OS	30,32,60,200

一般的に，認証システムとユーザ端末の間には NAT は存在しないため，パケットの TTL 値は各 OS のデフォルト値のままである．しかし，認証システムとユーザ端末の間に NAT が存在する場合，TTL 値は通過した NAT やルータの数だけ減少するため，認証システムを通過する際には OS のデフォルト値とは異なる値を持っていることになる．そこで，各 OS のデフォルト値以外の TTL 値を持つパケットが認証システムあるいはルータを通過した際に，ファイアウォールなどの機能を用い，ログを残す設定にしておく．これにより，ユーザ端末が NAT を通して通信する場合，ログに記録される．本研究の動作検証においては，認証システムのファイアウォールを使用し，ログの記録等を行った．

Java アプレットが動作しない環境である場合は TTL 値による NAT 検出機構が動作する．パケットが OS のデフォルト値ではない TTL 値を持つときに残されるログに，ユーザ端末からのパケットが記録されていないかどうかを検査し，認証時の当該端末からの通信とみられるログがある場合にはネットワーク利用の許可を行わず，署名済み Java アプレットによる検出時と同様に警告ページを表示する．そうでない場合は通常通りにネットワーク利用の許可が行われる．

3. 動作検証

本手法を検証するため，Web を用いた認証を行うネットワーク利用者認証システム Opengate[6] に対して実装し，検証を行った．また，実装においては，すでに稼働中の実環境への機能追加を容易にするため，Opengate 本体へのコードの変更などは行わず，一部設定の変更と Java アプレットなどの独立したプログラムの追加のみで実現した．

3.1 Opengate

Opengate は許可されたユーザのみにネットワークの利用を許可し，その利用記録を取得することを目的とし佐賀大学で開発されているネットワーク利用者認証システムである．ユーザは特別なソフトウェアやデバイスを必要とせず，ネットワーク利用開始時に Web ブラウザを通して認証を行うことで，個人の端末をインターネットに接続することができる．

Opengate は Web ブラウザを通して簡単な認証画面で認証を行うため，既存の LDAP や RADIUS，POP3 などを認証に使用することができ，Shibboleth[7] によるシングルサインオンにも対応している [8]．また，Opengate はユーザによるネットワークの利用を，Ajax や JavaScript などを用い，Web ブラウザを通して監視しており，ユーザが認証に利用した Web ブラウザを終了すると即座にファイアウォール (IPFW) を閉鎖し，ユーザ端末のネットワークへの接続を閉じる．また，何らかの理由でネットワークへの接続が遮断された場合にもファイアウォールを閉鎖する．

Opengate は，ユーザ ID，利用端末の IP アドレス，MAC アドレス，Web ブラウザのユーザエージェント，利用開始日時，利用終了日時を SYSLOG を通して記録する．

Opengate の動作環境を表 2 に示す．

表 2 Opengate 動作環境
Table 2 System requirement of Opengate

OS	FreeBSD4.0 以降
必須ソフトウェア	Apache, IPFW, SQLite
推奨ソフトウェア	natd, DHCP, perl, BIND

図 1 に Opengate ネットワークの構築例を示す．

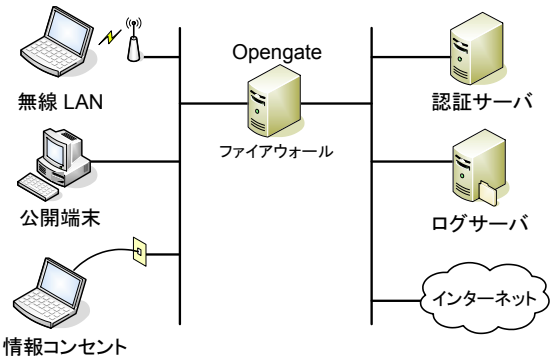

図 1 Opengate 構築例
Fig. 1 Opengate System

Opengate は C 言語で実装されたソフトウェアで，FreeBSD 上で動作する．Web サーバとして Apache，ファイアウォールとして IPFW を使用し，ユーザ端末を接続するネットワークの出口にゲートウェイとして設置される．

3.2 動作検証環境

本研究における Opengate および NAT 検知システムの動作検証に用いた環境を表 3 に示す．

3.3 NAT 検出システムフロー

図 2 に NAT の検出とネットワーク通信の遮断の流れを示す．

表 3　動作検証環境
Table 3　Verification environment of operation

OS	FreeBSD 8.1-RELEASE
Web サーバ	Apache 2.2.25
ファイアウォール	IPFW
データベース	SQLite 3.7.9
DHCP サーバ	isc-dhcp-server 4.1
NAT 検出プログラム	PHP 5.2.16, OpenJDK 1.7.0, Oracle Java 1.7.0
その他	OpenSSL 1.0, Perl 5.10.1, Shibboleth 2.3.1, natd

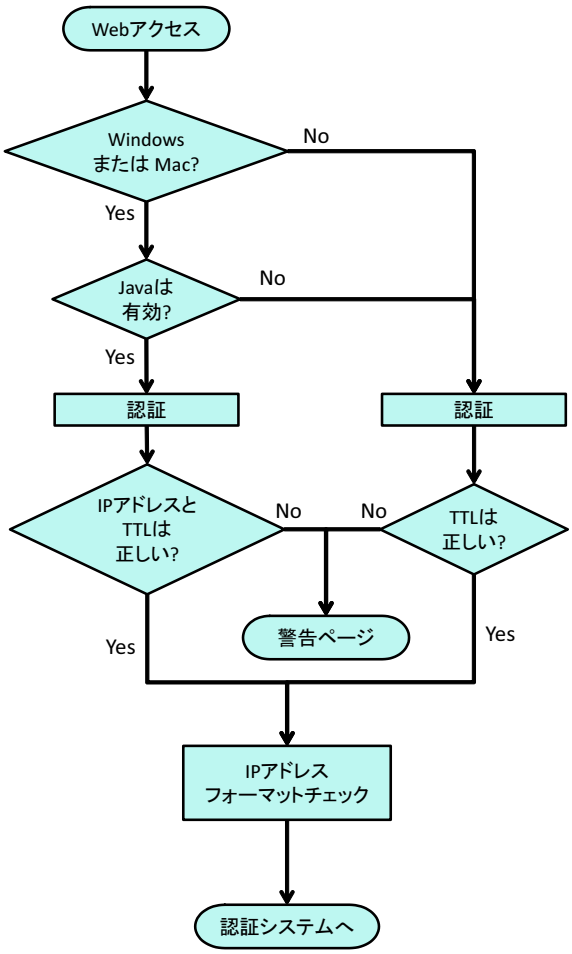

図 2　NAT 検出のフローチャート
Fig. 2　Flow Chart of NAT Detection

(1) ユーザ端末が Web アクセスを行うと，OS 判別ページへリダイレクトする
(2) OS が Windows あるいは MacOS であるかを判定する．Yes なら (3) へ，No ならば (5) へ
(3) Java が有効なブラウザかどうか判定する．Yes ならば (4) へ，No ならば (5) へ
(4) Java が有効な場合，認証後に IP アドレスの比較と TTL 値の観測を行う
　(a) IP アドレスの比較結果が一致し，パケットの TTL 値が OS のデフォルト値ならば (6) へ
　(b) IP アドレスが一致しないか，TTL 値が減少していれば警告ページへ (終了)
(5) Java が無効な場合，認証後に TTL 値の観測による NAT 検出判定を行う．TTL 値が妥当ならば (6) へ，デフォルト値でなければ警告ページへ (終了)
(6) IP アドレスのフォーマットチェックを行い，正しければ 認証システムへ，正しくなければ警告ページへ (終了)

iOS や Android においては Java アプレットが動作しない．そのため，(2) において分岐させ，TTL 値の観測による NAT 検出のみを行う．また，Linux においては，セキュリティ設定上，自己署名の Java アプレットが動作しなかったため，現状では TTL 値の観測による NAT 検出手法のみを適用している．

Java アプレットによる判定および TTL 値による判定の双方で表示される警告ページを図 3 に示す．

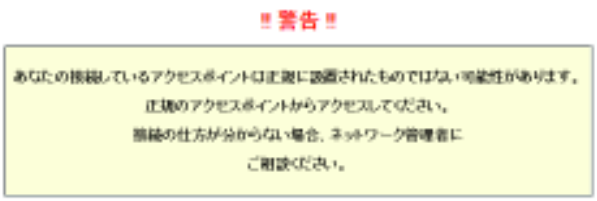

図 3　警告ページ
Fig. 3　Warning Page

3.4　ログの記録

NAT を使用していると考えられるユーザを検知した場合，ネットワーク利用の遮断とともに，そのユーザについてのログを SYSLOG 経由で nat_warning.log に，アクセス日時，ユーザ名，ユーザ端末の実際の IP アドレス，認証システムから観測した IP アドレス，ユーザ端末が通信に使用している NIC の MAC アドレスを記録する．MAC アドレスは NAT を経由してアクセスした端末の特定のために記録する．

4.　検証結果

4.1　通常環境 (非仮想化環境) における検証

本手法を導入した認証システムを用いて，NAT を設置しない場合，設置した場合，Java アプレットを有効にした場合，無効にした場合の動作検証を行った．それぞれの場合の NAT 検出の結果を表 4 に示す．

ユーザ端末と認証システムの間に NAT を設置した場合，Windows および MacOSX では，Java アプレットが有効な場合，無効な場合の両方で認証後に警告ページが表示され，通信が遮断された．また，ユーザ端末と認証システムの間に NAT が存在しない場合は，Java アプレットが有効な場合，無効な場合ともに警告ページは表示されず，認証シス

表 4　検証結果
Table 4　Verification Result

NAT	OS	検出結果	
		アプレット有効	アプレット無効
なし	Windows7	非検出	非検出
	Mac OSX	非検出	非検出
	Ubuntu 13.10	-	非検出
	iOS7.1	-	非検出
	Android4.2.2	-	非検出
あり	Windows7	検出	検出
	Mac OSX	検出	検出
	Ubuntu 13.10	-	検出
	iOS7.1	-	検出
	Android4.2.2	-	検出

テムによりネットワーク利用が許可されたため，想定通りの動作をしたと考える．

Linux では自己署名を行った Java アプレットの実行時に警告が表示され，ブロックされたため，TTL 値の観測による検出のみを行っている．この場合も NAT がある場合には検出し，NAT がない場合には検出されなかった．また，iOS と Android ではアプレットが動作しないため，TTL 値の観測による手法のみで検出を行い，正常に検出できた．

また，NAT を通して接続したとき，nat_warning.log には次に示すログが記録されていることが確認できた．

nat_warning.log

```
Apr 9 11:14:17 vmgate NAT_WARNING[29013]:
2014/04/09 15:24:01 Username: suenaga RemoteIP:
192.168.200.71 ClientLocalIP: 192.168.1.100 Client-
MacAddr: 64:80:99:38:5e:84

Apr 9 11:24:44 vmgate NAT_WARNING[29245]:
2014/04/09 16:29:28 Username: suenaga RemoteIP:
192.168.200.71 ClientLocalIP: NA NatMacAddr:
00:22:cf:34:28:a9
```

このうち，上段のログは Java アプレットが動作する状態で利用者認証を行うネットワークにアクセスし，認証システムが把握しているユーザ端末の IP アドレスと，実際にユーザ端末に割り振られている IP アドレスが一致しなかったために記録されたものである．そのため，認証システムが把握している IP アドレスおよび実際のユーザ端末の IP アドレスの比較が行われ，NAT が検出された結果としてアクセス日時，ユーザ名，二つの IP アドレス，およびユーザ端末の NIC の MAC アドレスが記録されている．

下段のログは Java が無効な状態であるため，Java アプレットが動作せず，TTL 値の監視による NAT の検出が行われ，ログを残したものである．TTL 値の監視による手法ではユーザ端末の実際の IP アドレス，および MAC アドレスの取得ができないため，認証システムが把握しているユーザ端末の IP アドレスと，検出された NAT の MAC アドレスが記録されている．

これらの動作結果，およびログの記録状況は本手法の実装において想定した通りであり，正常に動作している．

4.2　仮想化ソフトウェア使用環境における検証

仮想化環境では，ホスト OS とゲスト OS 間の接続が NAT となることがある．そこでユーザが使用端末に仮想環境などを導入している場合を想定し，検証を行った．検証には，Windows 7 SP1 をホスト OS とし，仮想化ソフトウェアとして VMWare Player を用い，ゲスト OS に Fedora19 および Ubuntu13.10 を使用した．検証結果を表 5 に示す．この検証においては，ユーザ端末と Opengate サーバの間には NAT を設置していない．

表 5　仮想環境での検証結果
Table 5　Verification Result on Virtual Environment

仮想ネットワークモード	OS	検出結果	
		アプレット有効	アプレット無効
NAT	Windows7	非検出	非検出
	Ubuntu13.10	-	検出
	Fedora19	-	検出
ブリッジ	Windows7	非検出	非検出
	Ubuntu13.10	-	非検出
	Fedora19	-	非検出

ホスト OS とゲスト OS 間の接続を NAT モードとし，ゲスト OS 側でネットワーク利用認証を行った場合，実際にはユーザ端末-認証システム間に NAT を設置していないにもかかわらず，NAT が検出された．仮想化ソフトウェアを NAT モードで使用すると，TTL 値の減少や IP アドレスの変換が行われるため，NAT として検出される結果となる．これについては想定していた結果であるが，ユーザ端末自体は NAT を経由して接続していないため，ユーザからは誤検出と認識される可能性が高い．ホスト OS 側であらかじめネットワーク利用認証を行った場合は NAT は検出されず，ゲスト OS 側でもネットワークを利用可能である．そのためユーザに対しては，ホスト OS での利用認証を推奨する必要がある．

ホスト OS とゲスト OS 間の接続をブリッジモードで行った場合は，ゲスト OS 側の仮想 NIC にもホスト OS と同じネットワークアドレスを持つ IP アドレスが割り当てられるため，どのパターンにおいてもネットワーク利用認証が可能である．この場合はネットワークを利用する OS ごとに利用認証を行う必要があるが，すべてのケースにおいて，本研究の想定通りの動作をしている．

仮想ネットワークのモードによって，ゲスト OS とホスト OS のうち適切な OS で認証を行うようにユーザへ周知することで，ネットワークを適切に利用できる．

5. 考察

本研究では，Web ブラウザを用いてネットワークの利用許可を行う認証システムにおける NAT の検出手法として，署名済み Java アプレットを用いた手法，および TTL 値の観測に基づいた手法を用いている．

署名済み Java アプレットを用いる場合，クライアントとなるユーザ端末のネットワークインタフェース情報を直接取得することができる．これにより，認証システムがユーザ端末のものであると認識している IP アドレスと実際のユーザ端末の IP アドレスを比較できるため，NAT の検出が可能であると考えられる．しかし，この手法では Java アプレットの動作が前提となるため，Java がインストールされていない端末での動作が期待できない．また，Java 7 update51 以降は，正式なコードサイニング証明書を取得し，アプレットに署名する必要があるが，アプレットへの署名は一つで済むため，SSL サーバ証明書のようにサーバごとに取得する必要はない．

TTL 値の観測に基づいた手法では Java アプレットの動作に頼らずに検出することが可能であるが，仮想環境などがユーザ端末内部で動作している場合に，それらを通した通信では TTL 値が減少した状態でパケットが送信される可能性がある．これについては，ホスト OS とゲスト OS のうち適切な OS でネットワーク利用認証を行うことでネットワークを利用できる．

OS のデフォルト TTL 値は簡単に変更が可能であり，意図的に TTL 値を変更することで NAT が存在すると誤認させることも，NAT は存在しないと誤認させることも可能である．また，Java アプレットを無効化することも簡単であり，TTL 値の偽装と併せると本 NAT 検出機能をすり抜けることが可能である．また，Windows ネットワーク共有のような，ユーザ端末自体が NAT として動作するソフトウェアの使用を想定していないため，そういったソフトウェアで NAT 化したユーザ端末を通した利用は検出できない．ファイアウォールで各 OS のデフォルト TTL 値ではないパケットを遮断する方法もあるが，その場合，仮想ネットワークを NAT モードで使用している仮想化ソフトウェアのゲスト OS からの通信が不可能となるため，TTL 値の観測による即時遮断を行っていない．

NAT の検出においては，冒頭で紹介した IPid を用いた NAT 検知手法や，トレースパケットの送出による能動的な検出手法がある．しかし，IPid を観測する手法においては，パケット中のヘッダを解析して IPid 列をプロットし，NAT であると判定するためには，多くのパケットの観測が必要となる．そのため，ユーザ端末からアクセスが行われた場合に即座に NAT であるかを検出することは難しい．また，OpenBSD 系の OS は IPid が完全にランダムで設定され，Linux ではセッションが確立するまでは IPid がランダムである．そのため，それらの OS を持つ端末が接続されるネットワークにおいては，IPid の観測による NAT 検出手法では，NAT が存在すると誤検知する可能性が高い．本研究による手法は即時性が高く，認証時に NAT を検出することが可能である．また，IPid の観測を行う手法を用いる場合は認証システムに大幅な変更が必要であるが，本手法であれば実環境に対しても導入が容易であり，より適していると考える．

検出システムからトレースパケットを送出し，それに対する応答と通過するパケットの TTL 値を比較する方式は，認証システムへの組み込み，および認証時点での NAT の検出を実現するためには大きな手間が必要となると予想される．そのため，本手法と比べた場合，本手法の方がに実装と導入はより容易であると考える．

6. おわりに

本研究では，署名済み Java アプレットを用いて，認証システムが把握しているユーザ端末の IP アドレスと，実際のユーザ端末の NIC の IP アドレスを比較することで，利用者認証を行うネットワーク上に設置された NAT を検出し，それらを経由した通信を遮断する手法を導入した．また，Java アプレットが動作しない環境での代替手段として，ユーザ端末の通信の TTL 値の減少を監視することで NAT を検出する手法を導入した．検証実験においては 4 章で述べたとおり，問題なく動作している．今後はより大きな規模のネットワークに導入して検証を行う必要がある．

参考文献

[1] Steven M. Bellovin. “A Technique for Counting NATed Hosts.”, Proc. IMW’02, Nov. 2002
[2] 高橋輝壮, 甲斐俊文, 篠原克幸. IPid を用いた NAT 検知手法の考察, 情報処理学会研究報告. CSEC, [コンピュータセキュリティ], 2006(26), pp97-102(2006.3)
[3] 三村守, 中村康弘:TTL を用いた能動的 NAT 検出手法の実装と評価, 情報処理学会論文誌, Vol.48, No.10, pp.3375-3385(2007.10)
[4] Java 7 リリースの変更, 入手先 ⟨http://www.java.com/ja/download/faq/release_changes.xml⟩
[5] Initial TTL Values 入手先 ⟨http://noahdavids.org/self_published/TTL_values.html⟩
[6] Opengate - A Network User Authentication System for Public and Mobile Terminals, 入手先 ⟨http://www.cc.saga-u.ac.jp/opengate/⟩
[7] Shibboleth, 入手先 ⟨http://shibboleth.internet2.edu/⟩
[8] 大谷誠, 江藤博文, 渡辺健次, 只木進一, 渡辺義明:シングルサインオンに対応したネットワーク利用者認証システムの開発, 情報処理学会論文誌, Vol.51, No.3, pp.1031-1039(2010.3)