XFW：アドレス偽造に対応したオープンスペース用ネットワークアクセスサービスの実装と導入

福田浩章[†,††] 山本喜一[†]

現在コンピュータ（端末）の小型化や低価格化にともない，端末を持ち歩くユーザが急速に増加している．また，ホテルや空港，大学のキャンパスといった公共の場所では，その端末に対してネットワーク接続サービスを提供するために，無線 LAN や情報コンセントが急速に設置され始めている．このような環境で無差別にネットワーク接続を許可した場合，不正利用の温床になることは容易に想像できる．したがって，あらかじめ登録された正規ユーザが利用する正規端末だけ接続を許可し，かつ IP アドレスや MAC アドレスの偽造による不正端末のネットワーク接続を防止する仕組みが必要である．一方，正規ユーザはどのような端末を使用しても容易にネットワークに接続できることが望ましい．そこで本論文では，一般的な端末のほとんどにあらかじめインストールされているウェブブラウザだけを使って，正規ユーザの識別および，IP アドレスや MAC アドレスを偽造した不正端末を使ったネットワークの利用を防止するシステム，XFW（Extensive Firewall）を提案し，その実装と結果を報告する．

XFW: A System Implementation against Address Spoofing for Open Network Access Service

Hiroaki Fukuda[†,††] and Yoshikazu Yamamoto[†]

Personal computers are getting much smaller and easier to carry about in these days. LAN sockets and Access Points of wireless network are being installed in public areas like hotels, airports and universities to provide network access methods for these computers. Howver it is easy to imagine that if everyone can access network without any authentications, there will be a lot of illegal accesses from these places. Therefore, we need a system that provides network connections for only computers that are used by authorized users and prevents malicious computers that spoof IP and/or MAC addresses. On the other hand, it is desirable that authorized users can access network with any computers they like.

In this paper, we propose a system called XFW (Extensive Firewall) that can distinguish authorized users and prevent illegal accesses from malicious computers by using only simple web browsers that are preinstalled in about all computers today. In addition, we report the implementation and result of XFW.

1. はじめに

現在ノート型コンピュータや PDA といった，携帯型端末（以下，端末と呼ぶ）の低価格化によって個人の端末を持ち歩いて利用するユーザが増えている．これにともなって，大学や企業に限らず，駅や空港といった不特定多数が出入りする場所でも，個人の端末をネットワークに接続するために無線 LAN や情報コンセントが急速に設置され始めている．このように多数のユーザが対象であり，かつユーザが個人の端末を使用してネットワークに接続する場合，利用する権利を持った正規ユーザが使用する端末（以下，正規端末と呼ぶ）のネットワーク接続だけを許可し，それ以外の端末（以下，不正端末と呼ぶ）のネットワーク接続を防ぐ必要がある．また，IP アドレスや MAC アドレスといった端末の識別子を偽造することによって，不正にネットワークを使用する行為も防がなければならない．しかし使用する端末がユーザ個人の所有物である場合，ユーザは自由に端末の設定が可能であるため，管理者がそれらの不正行為を防ぐことは容易ではない．不正端末のネットワーク接続を防ぐ従来の手法のほとんどは，端末の識別に IP アドレスまたは MAC アドレスを使用しているため，それらを偽造されると防ぐ

† 慶應義塾大学大学院理工学研究科開放環境科学専攻
Graduate School of Science for Open and Environmental Systems, Keio University
†† 慶應義塾インフォメーションテクノロジーセンター
Keio University Information Technology Center

手段がない．

一方，ユーザも初心者や熟練者までさまざまであり，専用ソフトウェアを使用した不正アクセス対策の場合，インストールや設定など混乱の原因となる場合が多い．したがって，ユーザは煩わしい作業をせず，どんな端末でも簡単に接続できることが望ましい．

そこで本論文では，一般的な端末のほとんどにあらかじめインストールされているウェブブラウザ（以下，ブラウザと呼ぶ）だけを使って，正規ユーザの識別および，IP アドレスや MAC アドレスを偽造した不正端末のネットワーク接続を防止するシステム，XFW の提案と実装を行った．

以下，2 章では，ユーザ認証によるネットワーク接続，または切断する方法と IP アドレスや MAC アドレス偽造に対する不正利用防止法について述べる．次に 3 章では XFW の具体的な設計と実装について述べ，4 章で XFW の運用と評価について述べる．そして 5 章で関連研究について述べ，最後に 6 章ではまとめと今後の課題について述べる．

2. 不正利用防止法

多数のユーザにネットワーク接続を提供する場合，**図 1** に示すようなプライベートネットワークを利用して接続させることが多い．この環境では，情報コンセントに接続するか，無線アクセスポイントの電波を受信した端末は動的な IP アドレスの割当てを受け，NAT BOX を経由してグローバルネットワークに接続できる．NAT BOX では一般的に DHCP や DNS，NAT，Firewall といった機能を提供している．XFW はこの NAT BOX 上で動作してユーザ認証を行い，Firewall の機能を利用して正規端末の接続を許可し，それ以外の不正端末を使用したアクセスをすべて切断する．正規ユーザの認証情報は Radius [9] や LDAP [10] といった外部認証サーバを利用する．

次に XFW を導入した場合のネットワーク接続方法と切断方法を示し，最後にアドレス偽造による不正利用を防ぐ方法について述べる．

2.1 接続方法

XFW では **図 2** の番号で示す手順でアクセス制限を行う．XFW を利用する場合，事前に Firewall 機能で DNS と HTTP 以外のパケットを破棄する設定にする必要がある．

(1) プライベートネットワークにつながった情報コンセントに接続するか，無線アクセスポイントから電波を受信し，端末が IP アドレスを取得するために DHCP リクエストを出す．

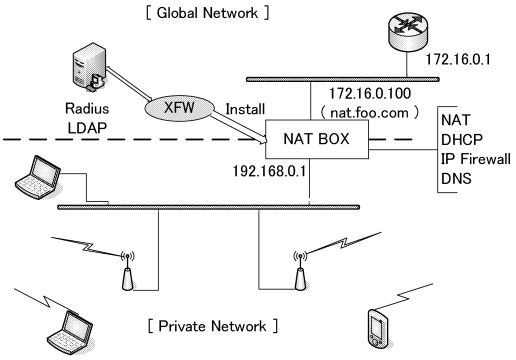

図 1 プライベートネットワークを使った接続法
Fig. 1 Network connection with private network.

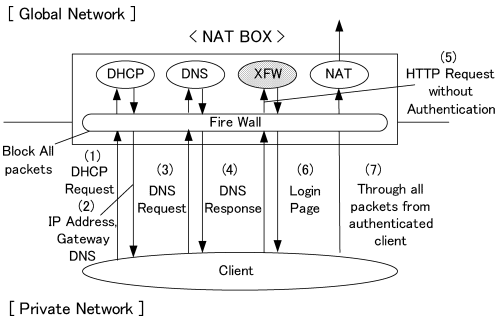

図 2 XFW を利用した接続方法
Fig. 2 How to connect with XFW.

(2) DHCP サーバが端末に IP アドレス，デフォルトゲートウェイ，DNS サーバ情報などを提供する．

(3) ユーザがブラウザを起動する．ブラウザは HTTP リクエストを出すサーバ名と IP アドレスの対応を解決するために DNS リクエストを出す．

(4) DNS サーバから IP アドレス情報を受け取る．

(5) ブラウザが，HTTP リクエストを出す．Firewall の機能によって，HTTP リクエストは破棄せず，強制的に XFW にフォワードする．

(6) XFW はユーザがリクエストしているコンテンツに関係なく ID，パスワードを要求する認証用コンテンツを返す．

(7) ユーザ認証に成功すると，XFW はその端末が正規端末であると認識して，使用している IP アドレスと MAC アドレス情報を取得し，その端末が出すパケットを通過させるために Firewall の設定を変更する．その結果，正規端末からのパケットだけが Firewall を通過する．すなわち，正規端末だけがネットワーク接続を許可さ

れる．

2.2 接続維持と切断の方法

2.1 節の方法で正規端末の接続を許可し，その後ユーザが利用をやめたあと再び Firewall の設定を元に戻す必要がある．この処理を行わないと，利用をやめた正規端末と偶然同じ IP アドレスを取得した端末が，認証を受けずにネットワークに接続できてしまう．このため，正規端末が接続しているかどうかをつねに確認（生存確認）し，利用の終了を検出する必要がある．

正規端末の生存確認をするために，XFW では正規端末から XFW に対して，生存を通知するためのセッション ID を一定間隔で送信させる．XFW ではこのセッション ID を利用し，次のようにして端末の切断を検出し，実行する．

(1) XFW は，正規端末の IP アドレスと MAC アドレス，セッション ID を関連付け，セッション ID を受け取るたびに最終受信時刻を更新する．
(2) XFW は一定間隔でセッション ID の最終受信時刻をチェックし，一定時間更新のない正規端末を切断したと判断する．そして切断対象の端末が使用する IP アドレスと MAC アドレス情報を抽出する．
(3) (2) で抽出した端末の IP アドレスと MAC アドレス情報をもとに，Firewall の設定を変更して対象となる端末を切断する．

2.3 アドレス偽造の防止法

XFW では，IP アドレスと MAC アドレス，セッション ID を使用して正規端末を識別しており，3 つの値が完全に一致しない場合は不正利用と見なして即座に端末を切断する．しかし，2.1 節で述べたように，不正端末であってもプライベートネットワークには接続できてしまうため，正規端末との通信を傍受され，セッション ID を盗まれると不正利用されてしまう．また，セッション ID のやりとりを HTTPS で行うことも考えられるが，事前の評価の結果せいぜい秒間 20 リクエスト程度しか処理できない．そこで XFW では，秘密鍵と MD5 ハッシュアルゴリズムを利用し，ワンタイムパスワードと同様な **図 3** で示す方法で正規端末と XFW でやりとりするセッション ID を毎回変更する．

(1) XFW は正規端末に対して秘密鍵を発行し，認証成功を示すコンテンツに埋め込んで返す．この間の通信には HTTPS を使用するため，安全に端末に秘密鍵を送ることができる．
(2) 正規端末では受け取った秘密鍵の MD5 ハッシュ値を計算し，その値を保存するとともに，HTTP を使って XFW に送信する．
(3) XFW でも正規端末と同様に，その正規端末に発行した秘密鍵の MD5 ハッシュ値を計算し，受け取った値と比較する．そして 2 つが同値の場合，その端末を正規端末であると認識し，受け取ったセッション ID と時刻を更新する．
(4) 正規端末は一定時間待った後，(2) で保存した値と秘密鍵を結合し，結合した値の MD5 ハッシュ値を計算する．そして (2) と同様にその値を保存し，XFW に送信する．
(5) XFW でも正規端末と同様に，前回受け取った値と秘密鍵を利用してハッシュ値を計算し，受け取った値と比較することによって正規端末を識別する．
(6) 以後，(4) と (5) を繰り返し，正規端末の接続を維持する．

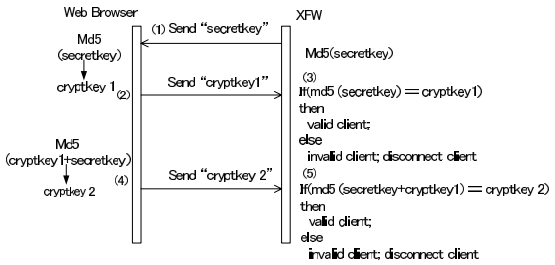

図 3 セッション ID の変更方法
Fig. 3 The method for changing sessionID.

この方式によって，仮に IP アドレスと MAC アドレスを偽造され，セッション ID を盗まれたとしても，盗んだセッション ID はすでに無効であるため正規端末と不正端末を識別することができる．

ワンタイムパスワードでは，事前に秘密鍵を交換する必要があるが，上述の方式では認証終了時に XFW から秘密鍵が交付されるので，事前の鍵交換が必要ない．また，後述するとおり，XFW ではこの方式を Javascript が動くブラウザだけで実現している．

2.1 節で述べたように，XFW を使用してネットワークに接続するためには，端末が DHCP で IP アドレスを取得したあと，少なくとも 1 回は HTTP リクエストを送信する必要がある．しかし現実には，ネットワークに接続するユーザの多くがウェブブラウジングを行うため，大部分のユーザを認証画面に誘導することができる．

3. XFW の設計と実装

本章では，はじめに XFW を構成する各モジュールの機能と，それぞれの関係について述べる．次に，

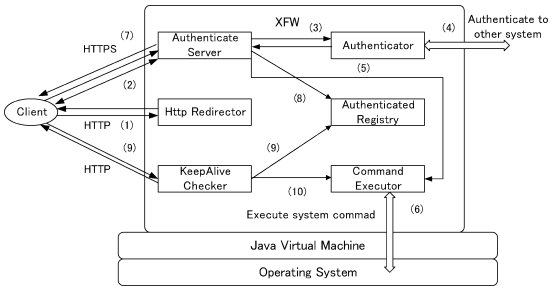

図 4 XFW のアーキテクチャ
Fig. 4 Architecture of XFW.

ユーザ認証時にブラウザと XFW，XFW と外部認証サーバでやりとりされるデータについて述べる．そして，セッション ID を変更する方法について述べ，最後に XFW を動作させるために管理者が設定するパラメータを示す．

3.1 XFW

XFW 本体は，それが動作する OS に依存せず動作を保障するために Java 言語を使用する．**図 4** に示すように，XFW は大きく分けて 6 つのモジュールで構成される．次にそれぞれのモジュールとその役割について述べる．

3.1.1 HttpRedirector

HttpRedirector は，受け取った HTTP リクエストに対してつねに Authenticate Server へリダイレクト応答を返す（図 4 の (1)）．Firewall のフォワード機能によってフォワードされた HTTP リクエストは，すべてこのモジュールが処理する．

3.1.2 Authenticate Server

HttpRedirector がリダイレクト応答を返した結果，ブラウザが Authenticate Server にリクエストを出す．Authenticate Server はリクエストを受け取ると，ユーザ認証用コンテンツを返す（図 4 の (2)）．また，ユーザが送る認証情報（ID とパスワード）を取得し，認証処理を Authenticator に依頼する（図 4 の (3)）．認証に成功した場合，Firewall の設定を変更するために Command Executor に処理を依頼する（図 4 の (5)）．そして，ユーザ認証と Firewall の設定変更に成功すると，秘密鍵を発行し，ユーザに認証成功を示すコンテンツを返す（図 4 の (7)）．それと同時に Authenticated Registry に端末の IP および MAC アドレス情報と，秘密鍵を登録する（図 4 の (8)）．一方，認証に失敗，または設定変更に失敗した場合は，認証失敗と再認証を促すコンテンツをユーザに返す．

3.1.3 Authenticator

Authenticator は認証情報を受け取り，さらに外部の認証システム（Radius，LDAP etc.）と連携してユーザ認証を行う（図 4 の (4)）．そして Authenticate Server に認証結果を返す．このモジュールと他のモジュール（Authenticate Server など）は，Java のインタフェースで結合しており，実装クラスを入れ替えることで，さまざまな外部認証システムとの連携が可能になる．

```
HTTP/1.1 307 Temporary Redirect
Connection: close
Location: https://nat.foo.com/login.html
```

図 5 リダイレクトレスポンス
Fig. 5 Redirect response.

3.1.4 CommandExecutor

Java の実行環境から，OS のシステムコマンド，または外部プログラムを呼び出す（図 4 の (6)）．

3.1.5 KeepAlive Checker

KeepAlive Checker は，セッション ID を受け取り，そのセッション ID を送った端末が正規端末であることを確認した後，最終受信時刻の更新処理を Authenticated Registry に依頼する（図 4 の (9)）．一方，不正端末であると判断した場合，Command Executor に切断処理を依頼するとともに，Authenticated Registry のエントリを削除する（図 4 の (10)）．また，このモジュールは Authenticated Registry から一定間隔で登録されているすべてのセッション ID と最終受信時刻を取得する．そして，一定時間更新されていない端末は切断したと判断し，不正端末の場合と同様に該当する端末を切断する．

3.2 認 証 処 理

認証処理では，HttpRedirector とブラウザ，Authenticate Server とブラウザ，そして Authenticator と外部認証システム間で次に示すデータがやりとりされる．

3.2.1 HttpRedirector とブラウザ間

HttpRedirector は受け取ったリクエストに対して **図 5** で示す HTTP レスポンスを返す．ここでは説明のため，図 1 で示したように NAT BOX にはグローバル IP アドレス “172.16.0.100” が割り当てられ，DNS サーバによって “nat.foo.com” として名前解決ができるものとする．この結果，ブラウザは “Location” ヘッダで示した URL にリクエストを再送する．

3.2.2 Authenticate Server とブラウザ間

HttpRedirector のレスポンスによって，ブラウザは Authenticate Server に認証用コンテンツをリクエストする．認証用コンテンツは管理者が自由にカスタマイズ可能だが，HTTP の POST メソッドを利用

```
<form action="/auth.html" method="post">
<input type="text" name="loginId" value=""/>
<input type="password" name="passwd" value=""/>
<input type="submit" name="Send">
</form>
```

図 6 認証用コンテンツに必要な部分
Fig. 6 Necessary part of contents for authentication.

```
public interface Authenticator {
    public boolean authenticate(String loginId, String passwc);
}
```

図 7 Authenticator インタフェース
Fig. 7 Authenticator interface.

して “/auth.html” に ID（“loginId”）とパスワード（“passwd”）パラメータを送信できるものでなくてはならない．具体的には**図 6** で示す部分を含む必要がある．

ユーザがログイン ID とパスワードを入力し，送信ボタンを押すと，HTTP の POST メソッドでデータが送られ，Authenticate Server が認証情報を取得する．この間のデータは HTTPS で暗号化されるため，安全にやりとりできる．

3.2.3 Authenticator と外部認証システム

Authenticator は**図 7** で示すインタフェースで定義される．したがって，このインタフェースを実装する具象クラスを入れ替えることで，さまざまな認証方式が実現できる．現在の実装では，Radius プロトコルを使用して外部 Radius サーバと連携する RadiusAuthenticator を実装している．

3.3 セッション ID の変更方法

2.3 節で述べた方法を実現するためには，

- 生成した秘密鍵を安全に端末に送り，かつ端末から漏洩しない仕組み，
- 端末で MD5 ハッシュ値の計算と，値の保持，

という要件を満たす必要がある．HTTP だけを使った通信では，端末に送られた秘密鍵を保存しておくことが難しい．唯一クッキーを使うことで保存することはできるが，通信は暗号化されていないため，ネットワークを盗聴されると簡単に盗まれてしまう．その結果，セッション ID を毎回変更したとしても端末の偽造が可能になる．

そこで XFW では，HTML の FRAME と Javascript を利用して，HTTP だけを使って上で述べた要件を満たす．Javascript は通常のコンピュータに限らず，PDA などの携帯端末でも動作するが，ユーザによってはセキュリティなどの理由で無効にしている場合も考えられる．これに対応するために，XFW は端末ごとに Javascript が有効か無効かを判定する．次に Javascript が有効な場合にセッション ID を変更する方法について述べ，Javascript の動作判定と無効な場合の対応について述べる

```
<HTML>
<FRAMESET ROWS="0%,0%,*">
<FRAME SRC="" NAME="$SECRET">
<FRAME SRC="script.html" NAME="script">
<FRAME SRC="success.html" NAME="success">
</FRAMESET>
</HTML>
```

図 8 Top.html
Fig. 8 Top.html.

```
<HTML>
<HEAD><TITLE>Authenticate Success</TITLE></HEAD>
<BODY>
   認証成功.<BR/>この画面は閉じないでください.<BR/>
</BODY>
</HTML>
```

図 9 Success.html
Fig. 9 Success.html.

```
<HTML>
<HEAD><SCRIPT type="text/javascript">
var secret, sessionId;
var url = "http://nat.foo.com:8000/success.html?sessionId=";
function keepalive() {
   setTimeout("keepalive()",5000);
   secret = parent[0].name;   (1)
   var currentKey = secret + document.forms[0].sessionId.value;
   sessionId = MD5_hexhash(currentKey);   (2)
   document.forms[0].sessionId.value = sessionId;   (3)
    var sendUrl = url + sessionId;
    parent.success.location.href = sendUrl;   (4)
}
function MD5_hash(data) {
 // encrypt data
 ----------
 return crypted data;
}
</SCRIPT></HEAD>
<BODY onload="keepalive()">
<FORM NAME="form">
<INPUT TYPE="hidden" name="sessionId" value=""/>
</FORM>
</BODY>
</HTML>
```

図 10 Script.html
Fig. 10 Script.html.

3.3.1 Javascript が有効な場合

図 8 に FRAME のトップページ（top.html），**図 9** と**図 10** に FRAME 内部で使用する success.html と script.html のコンテンツを示す．図 8 に示すように，top.html では FRAMESET タグで 3 つの FRAME を定義しているが，最初の FRAME 指定は “SECRET” 部分に秘密鍵を埋め込むために使用し，2 番目の FRAME には MD5 ハッシュ値の計算をする Javascript と，値を保存するための form を持つ script.html を指定する．これらのフレームの幅はいずれも “0%” 指定であるため，ユーザが目にすることはない．

```
<HTML>
<HEAD>
<TITLE>Javascript Check</TITLE>
<meta http-equiv="Refresh"content="10 ;
URL=http://nat.foo.com:8000/nojsauth.html?sessionId=xxxx"/>
<SCRIPT type="text/javascript">
  function checkjs() {
      location.href="top.html?sessionId=xxxx";
  }
</SCRIPT>
</HEAD>
<BODY bgcolor="#FFFFCC" text="#000000" onload="checkjs()">
  ----------
</BODY>
</HTML>
```

図 11 Jscheck.html
Fig. 11 Jscheck.html.

3 番目の FRAME は “100%” 指定で，認証成功を示す success.html を指定する．3.2.2 項で述べたように，XFW ではユーザ認証時の通信に HTTPS を使用するため，安全に秘密鍵を端末のブラウザに送ることができる．script.html の Javascript では top.html に埋め込まれた秘密鍵（図 10 の (1)）を使って MD5 ハッシュを計算し（図 10 の (2)），form の hidden タグに保存する（図 10 の (3)）．そしてハッシュ値を success.html の GET パラメータ “sessionId” として付加し，3 番目の FRAME をリロードする（(図 10 の (4)）．この方式では，秘密鍵が埋め込まれる top.html と script.html は HTTPS で一度だけ送信されるため，安全な秘密鍵のやりとりができる．また，ユーザは認証後の画面を開き続けておくだけで接続を維持することができる（ブラウザは最小化してかまわない）．

3.3.2 Javascript の動作判定と無効な場合の対応

XFW では，**図 11** に示す jscheck.html を利用し，Javascript の動作判定を行う．

jscheck.html は，ユーザ認証成功直後に送信される．Javascrip が有効なブラウザは，onload イベントによって図 11 の “checkjs()” 関数を実行し，3.3.1 項で述べた方法でセッション ID を毎回変更しながら接続を維持する．一方 Javascript が無効なブラウザは，関数を実行せずに図 11 の ⟨meta⟩ タグで記述された URL にリクエストを出す．このときリクエストする nojsauth.html にも同様の ⟨meta⟩ タグ記述が存在するため，ブラウザは定期的に同じセッション ID を XFW に送信して接続を維持する．

図 10 と図 11 の例では，3.1.5 項で述べた KeepAlive Checker が nat.foo.com の 8000 番ポートで処理をするが，次に述べる管理者が設定するパラメータによって KeepAlive Checker の動作を変更可能である．

表 1 管理者が設定するパラメータ
Table 1 Configurable parameters for administrator.

設定するパラメータ名	パラメータの意味
HTTP_PORT	HttpRedirector が使用するポート
HTTPS_PORT	Authenticate Server が使用するポート
KEEPALIVE_PORT	KeepAlive Checker が使用するポート
KEEPALIVE_INTERVAL	クライアントがセッション ID を送信する間隔（秒）
CLOSE_INTERVAL	KeepAlive Checker が切断したと判断する間隔（秒）
CHECK_INTERVAL	KeepAlive Checker がポーリングチェックする間隔 (秒)
RADIUS_INFO	Radius サーバのアドレス，ポートと認証のためのキー

3.3.3 管理者が設定するパラメータ

表 1 に XFW を動作させるために管理者が設定するパラメータを示す．管理者は実行する環境に合わせてこれらのパラメータを設定する．特に XFW の動作を決定するのが KEEPALIVE_INTERVAL，CLOSE_INTERVAL であり，KEEPALIVE_INTERVAL の値が小さければ小さいほど，正規端末のブラウザは頻繁にセッション ID を送信するようになり，負荷が増大する．また，CLOSE_INTERVAL は必ず KEEPALIVE_INTERVAL よりも大きい値に設定する必要があり，さもなければ接続中の端末はすべて切断されてしまう．RADIUS_INFO に関しては，アドレス，キー，ポートをコロン（:）区切りにし，さらにその組合せをセミコロン（;）区切りにすることによって，複数の Radius サーバと連携することが可能である．その場合 XFW は，先頭に指定されているサーバから順に認証を始め，どこかのサーバで認証が成功した時点で認証成功と判断する．

4. 運用と評価

XFW の基本性能を示すために，XFW 単体での負荷テストを行った．また現在大学のキャンパスで XFW を導入し，無線と有線のネットワーク接続サービスを運用している．次にそれぞれについて述べる．

4.1 XFW 単体の負荷テスト

XFW 単体での性能を測定するため，我々は Web Application Stress Tool [14]（以下，Stress Tool と呼ぶ）を使って負荷テストを行った．XFW の方式では，接続するクライアントが増えた場合，最も性能に影響が出るのがセッション ID を処理する部分（以下，接続維持部分と呼ぶ）であるため，XFW への同時アクセスを 10〜100 まで 10 ずつ増やし，セッション ID を毎

表 2 テスト環境
Table 2 Test environment.

CPU	Pentium4 2.8 (GHz)
メモリ	1 (Gbyte)
ネットワーク帯域	1,000 (Mbps)
Java VM	J2SDK-1.4.2_04

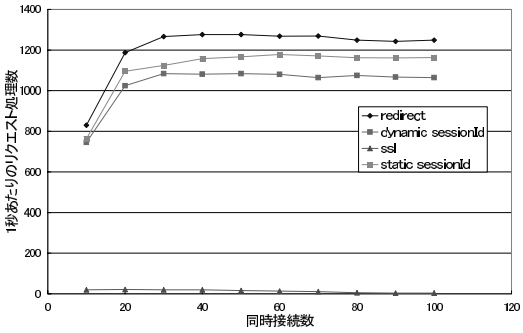

図 12 1 秒あたりのリクエスト処理数
Fig. 12 The number of processed requests for 1 second.

回変える場合と固定の場合で，1 秒間あたりに XFW が処理するリクエスト数を測定する．また，比較対象として，最も処理が軽い HttpRedirector がリダイレクトする部分（以下，リダイレクト部分と呼ぶ）と，最も処理が重い SSL を使った認証部分でも同様に 1 秒間あたりのリクエスト処理数を測定する．使用するコンテンツの大きさは接続維持部分で 298 (Byte)，認証部分で 2.42 (KByte) である．**表 2** にテスト環境を示し，**図 12** に測定結果を示す．

図 12 を見ると，同時接続数が 30〜100 までの間はリダイレクト部分も接続維持部分もそれほど変化が見られない．つまり，テスト環境において XFW が 1 秒間に処理できるリクエスト数はリダイレクト部分で約 1,260，接続維持部分では Javascript 対応のためセッション ID を固定した場合約 1,160，毎回変更した場合約 1,070 である．セッション ID を毎回変更する場合，固定した場合と比較すると約 10%，リダイレクト部分と比較すると約 15%性能が低下する．しかし仮に 10 秒間隔でセッション ID をやりとりする場合，理論上約 10,000 ユーザが接続可能であり，実用上問題ない．一方，認証部分では同時接続数が 40 までは秒間約 20 リクエストを処理できているが，50 以降は接続エラーが頻発し，最終的に同時接続数 100 の場合は秒間 4 リクエストしか処理できなかった．接続維持部分と認証部分のコンテンツサイズを考慮しても，この違いは SSL の処理コストであると考えられる．ただし，認証処理は接続維持と異なり接続時に 1 度だけの処理のため，ユーザがランダムに利用を開始する現実の環境においては実用上の問題は起こらない．仮に問題が起きた場合には複数の XFW で負荷分散すればよい．

表 3 運用サーバの仕様
Table 3 Specification of the working server.

CPU	Pentium4 2.8 (GHz)
メモリ	1 (Gbyte)
ネットワーク帯域（global）	1,000 (Mbps)
ネットワーク帯域（private）	1,000 (Mbps)

表 4 XFW で使用するソフトウェア
Table 4 Softwares for XFW.

実行に必要な設定項目	使用したソフトウェア
Operating System	Linux（Fedra core3）
NAT	iptables
Firewall	iptables
DHCP	ISC-DHCP
Authenticate Server	OpenRadius
Java VM	J2SDK-1.4.2_08

表 5 大学での運用実績
Table 5 Working result in University.

平均延べユーザ数（人/日）	415
最大同時接続ユーザ（人）	71

4.2 大学での運用

大学では **表 3** で示す仕様のサーバに，**表 4** で示す OS やソフトウェアを使って XFW を導入し運用を行っており，Windows，MacOS，Unix 互換 OS のクライアントで動作することを確認している．また，運用実績として 1 日あたりのユーザと，最大同時接続ユーザ数を 1 カ月間測定し，平均延べユーザ数と最大同時接続ユーザ数を**表 5** に示す．

XFW はシステムのログとして，端末の接続と切断時にログイン ID，使用する IP アドレス，MAC アドレス，時刻を記録するため，iptables のログとあわせて不正利用の発見やトラブルに対応している．

4.3 評　　価

ここでは，セキュリティ，性能，ユーザの利用方法，導入コストの点から XFW を評価する．そして XFW の問題について述べる．

4.3.1 セキュリティ

XFW は IP アドレスと MAC アドレスに加え，毎回異なるセッション ID を使うことで正規端末を識別する．したがって，IP または MAC アドレスを偽造された場合でも，不正端末のネットワーク接続を防ぐことができる．しかし，不正端末であっても IP アドレスを取得することはできるため，プライベートネットワーク内部での不正アクセスは防ぐことができない．これを防ぐためには無線 LAN アクセスポイント，あ

るいはネットワークスイッチレベルでのアクセス制限が必要でり，そのようなアクセス制限に XFW は適さない．また，3.3.2 項で述べたように，Javascript が無効な場合，IP アドレスと MAC アドレスを偽造され，セッション ID を盗まれると不正に接続できてしまう．したがって，Javascript が無効である場合の接続を許可するかどうかは，利用する環境に応じて管理者が決定できる仕組みにする必要がある．

4.3.2 性　　能

接続中の端末が出すパケットは，セッション ID の送受信を除き XFW を経由せず NAT BOX あるいはルータでルーティングされる．したがって 4.1 節で述べたように，接続する端末が増えた場合 XFW にかかる負荷は接続維持部分の処理だけであり，図 12 の結果から実用上問題ない．そして表 5 で示したように，大学で運用を行っているが，特に問題は生じていない．仮に負荷が高い場合には表 1 で示した KEEPALIVE_INTERVAL の値を増やすことで負荷を減らすことができる．しかしその場合，端末が使用をやめてから最長で設定した秒数だけそこで使用していた IP アドレスからのパケットは通過してしまう．そのため，その間に偶然同じ IP アドレスを取得した他の端末は，認証しなくてもネットワークを利用できてしまうことになる．したがって，負荷が高くなり KEEPALIVE_INTERVAL の値を増やす必要がある場合は，複数のネットワークとサーバを使用して分散処理をするほうが安全である．一方，正規端末が接続を維持するための負荷は，管理者が設定した間隔ごとに Javascript で MD5 ハッシュ関数を 1 回実行するだけであり，その負荷は無視できるほど小さい．

4.3.3 ユーザの利用方法

XFW は端末の HTTP リクエスト送信を契機にユーザ認証を行い，正規端末を特定したネットワーク接続を可能にする．この処理に必要なものはブラウザだけであり，最近の端末には標準でインストールされている．そして認証処理はウェブインタフェースを使用し，HTTPS で通信するため，安全で分かりやすく，初心者でも十分に利用することができる．

4.3.4 導入コスト

XFW 自体は Java で実装されたソフトウェアであるため，Java の実行環境があれば OS に依存しない動作が保障される．したがって 2.1 節で示したように，動作させる OS で利用できる Firewall ソフトウェアに HTTP リクエストを XFW にフォワードできる機能があり，XFW から Firewall のルールを変更できさえすれば導入可能である．実際に FreeBSD 上で ipfw を利用して動作を確認している．したがって，新規導入はもちろん，既存の NAT BOX に組み込むことも十分可能である．なお，本論文では XFW を NAT BOX に組み込んでいるが，NAT 変換は必須ではないのでルータに組み込むことも可能である．

4.3.5 XFW の問題点

XFW は 2.3 節で述べた方法で毎回異なるセッション ID を用いてアドレス偽造を検知し，不正利用を防ぐことができる．しかし，悪意のあるユーザが IP アドレスと MAC アドレスを偽装し，不正なセッション ID を送ることで接続中の正規端末を切断できてしまう．これに対しては正規端末が送るセッション ID が XFW が指定する間隔で送られることを利用し，不正な時刻に送られるセッション ID を無効とすることである程度防ぐことができるが，XFW の仕組みでは本質的に正規端末からのセッション ID と悪意のあるセッション ID を正確に区別することはできない．

5. 関 連 研 究

ネットワーク接続を認証つきで提供するという研究開発は過去にも事例があり，いずれも目的に対して十分な効果をあげている[1)~5)]．しかしこれらのシステムでは，ユーザ認証に専用クライアントなど，一般的な端末には標準でインストールされていないソフトウェアを使うため，あらかじめ別の環境でインストールしておかなければ利用できない．初心者がその作業をするのは困難かつ負担である．Beck のシステム[11)] では認証に telnet を使用しているが，認証情報が平文でやりとりされるため，セキュリティ上好ましくない．Opengate[4)] では，XFW と同様にブラウザだけを利用した認証方法を提供しており，認証時にブラウザにダウンロードされる Java アプレットが，サーバとの TCP コネクションを確立し，サーバがそれを監視することによって端末の切断を検知する．したがって，XFW と同様 IP や MAC アドレスを偽造したとしても不正利用することはできない．しかし，この仕組みには Java が動作するブラウザを使用する必要があり，Java が実装されていない PDA などの携帯端末では利用することができない．XFW では Javascript を使用しているが，PDA などの携帯端末にあらかじめインストールされているブラウザでも動作を確認しているため適用範囲が広い．また，Opengate では Java アプレットが実行できないクライアントに対し，認証後にユーザが明示的に利用時間を指定して接続を維持する仕組みが用意されている．しかし，毎回時間を指定することはユーザにとって少なからず手間がかかり，

設定する時間によってはユーザが利用をやめたあと偶然同じ IP アドレスを取得した端末が認証を受けずに長い時間接続できてしまう．XFW では，Javascript が無効であっても XFW 自身がそれを判断し，単純な HTTP だけで接続を維持するため，ユーザは接続維持や切断に関して特に意識する必要はない．そしていずれの場合にも，正規のユーザが利用をやめたあと，同じ IP アドレスを取得した別のユーザが利用できる時間は，最大でも管理者が設定した時間までである．

近年では，空港やホテルでも Universal Subscriber Gateway [12] や POPCHAT [13]，Apresia [6] といった製品を使用して，ネットワーク接続が展開されている．これらのシステムでは，XFW と同様に一般的なブラウザだけで認証を行うことができるが，正規端末を MAC アドレスを利用して識別しているため，MAC アドレスを偽造することによって不正利用が可能になる．また，既存のインフラがある場合には，これらの導入コストも無視できない．XFW では，すべてソフトウェアで構築できるため，既存のインフラと連携する場合でもシームレスに導入が可能である．LANA [2,3] では，802.1Q [7] に対応したスイッチを使用し，VLAN タグを使用して IP アドレス，MAC アドレスの偽造を防止している．LANA の場合，スイッチにカスケード接続されたハブを使って接続する正規端末を識別するために，必ず LANA クライアントが必要であり，無線 LAN アクセスポイントを接続することを考えると，すべての端末に LANA クライアントが必要になる．また，IEEE802.1X [8] に対応したスイッチ，または無線アクセスポイントを導入することも考えられる．しかし，実際に導入するためには，すべてのスイッチやアクセスポイントがこの規格に対応する必要があり，これを利用するソフトウェアも必要となる．XFW ではどのようなスイッチを使ったネットワークであっても関係なく，正規端末を識別するためには Javascript が有効であるブラウザだけあればよい．

6. ま と め

本論文では，多数のユーザを対象にネットワーク接続を提供する場合，利用するユーザを認証してアクセス制限を行うシステムである XFW の提案と実装を行い，負荷テストや実際に導入して評価することで有効性を示した．XFW では利用するユーザの利便性を重視し，一般的なブラウザだけでユーザ認証，および IP アドレスや MAC アドレス偽造による不正利用防止などの処理をすべて行うことができる．また，認証画面や認証後の画面をカスタマイズすることでユーザに必要な情報をアナウンスでき，ユーザサポートのコストも軽減できると考えられる．ただし，XFW では認証後に端末がやりとりするデータに関して暗号化などは行わないため，情報漏えいを防ぐためには SSL/TSL などの暗号化技術を別途導入する必要がある．また，オープンスペースでの無線 LAN 接続サービスには偽基地局による盗聴の問題もあり，これを防ぐには基地局と端末間で相互認証するなどの対策が必要である．

今後の課題として，XFW ではプライベートネットワーク内でのウィルス感染や，ウィルスを出すクライアントを強制的に切断することができない．これらに関しては，VLAN などを導入し，1 台のサーバで複数のプライベートネットワークを使用することで，ウィルスが広範囲に蔓延することをある程度防ぐことができる．また，Firewall のログを解析し，不正アクセスをしているクライアントを強制切断する仕組みが必要だと考えている．

参 考 文 献

1) 後藤英昭, 安西従道, 二階堂秀夫, 千田栄幸, 満保雅浩, 静谷啓樹：ユーザ認証機構を有する安全な無線 LAN・情報コンセント統合システムの構築, 平成 13 年度情報処理教育研究集会講演論文集, pp.382–385 (2001).
2) 石橋勇人, 山井成良, 安倍広多, 大西克実, 松浦敏雄：IP アドレス/MAC アドレス偽造に対応した情報コンセント不正アクセス防止方式, 情報処理学会論文誌, Vol.40, No.12, pp.4353–4361 (1999).
3) 石橋勇人, 山井成良, 安倍広多, 阪本 晃, 松浦敏雄：利用者ごとのアクセス制御を実現する情報コンセント不正利用防止方式, 情報処理学会論文誌, Vol.42, No.1, pp.79–88 (2001).
4) 只木進一, 江藤博文, 渡辺健次, 渡辺義明：利用者移動端末に対応した大規模ネットワークの Opengate による構築と運用, 情報処理学会論文誌, Vol.46, No.4, pp.922–929 (2005).
5) 広島大学情報メディアセンター：Portguard. http://www.portguard.org/
6) Apresia. http://www.apresia.jp/solution/secu.html
7) IEEE: 802.1Q-1998 IEEE Standards for Local area Metropolitan Area Networks: Virtual Bridge Local Area Networks: Virtual Bridge Local Area Networks, IEEE (1998).
8) IEEE: IEEE Draft P802.1X/D11: Standard for Port based Network Access Control, LAN MAN Standards Committee of the IEEE Computer Society (Mar. 2001).
9) Rigney, C.: RADIUS Accounting, RFC 2139 (1997).

10) Yeong, Y., Howes, T. and Kille, S.: Lightweight directory access protocol, RFC 1777 (Mar. 1995).
11) Beck, R.: Dealing with Public Ethernet Jacks-Switches, Gateways, and Authentication, *Proc. 13th System Administration Conference* (*LISA'99*), pp.149–154 (1999).
12) Universal Subscriber Gateway. http://www.nomadix.com/products/usg.asp
13) POPCHAT. http://www.popchat.jp/
14) Web Application Stress Tool. http://www.microsoft.com/technet/itsolutions/intranet/download/webstress.asp

(平成 17 年 11 月 25 日受付)
(平成 18 年 6 月 1 日採録)

福田 浩章（正会員）

1998 年慶應義塾大学理工学部計測工学科卒業，計算機科学専攻修士．現在，慶應義塾大学大学院開放科学専攻後期博士課程在籍および，慶應義塾インフォメーションテクノロジーセンター助手．主としてユビキタス環境における分散アプリケーションおよび，プラットフォームの研究に従事．日本ソフトウェア科学会会員．

山本 喜一（正会員）

1969 年慶應義塾大学工学部管理工学科卒業，管理工学専攻修士，工学博士（1987 年慶應義塾大学）．現在，慶應義塾大学理工学部情報工学科助教授．ソフトウェア科学，特にシステムの動的適合，ソフトウェア工学，ヒューマンインタフェース等の研究に従事．ACM，IEEE，情報システム学会，日本ソフトウェア科学会，電子情報通信学会，日本シミュレーション学会各会員．情報システム学会理事，編集委員長．