2005－CSEC－29（4）
2005／5／19

サービスを秘匿するTCPコネクション確立方式の有効性検証

梅澤　健太郎[†]　　高　橋　俊　成[†]

現在, サーバ計算機へのアプリケーション脆弱性を狙った攻撃が問題となっている. この問題に対して, 我々は TAP (TCP layer Application Protector) を提案している. この技術は, SYN パケットに格納した認証情報をもとに通信相手を認証し, 認証が成功した場合にのみ SYN/ACK を返信することで TCP コネクションの確立制御を行う方式である. 本方式により, 不当なコネクション要求を排除することができ, また, サーバアプリケーションの存在を秘匿することも可能である. 本論文では実装した TAP モジュールを利用した機能検証により TAP の接続性やサービス秘匿効果を確認するとともに, その有効性を通常の TCP 通信との性能比較や, DoS 攻撃を想定した実験での計測値により確認したので報告する.

Evaluation of a TCP Connection Scheme for cloaking services

UMESAWA Kentaro[†] and TAKAHASHI Toshinari[†]

Fixing software vulnerabilities which are exploitable via a network is one of the urgency for system administrators. But it is often hard to fix software vulnerabilities timely, because there are a lot of administrative problems in systems operation area and some of vulnerabilities don't have programs to fix it at that time. To solve this problem, we have developed TAP (Tcp layer Application Protector) which prevents attackers from establishing TCP connections with an authentication mechanism at TCP layer. In this paper, we prove effectiveness of the method by measuring a network and anti-DoS performance.

1. はじめに

現在, サーバ計算機において公開されているサービスに対して無差別かつ広範囲に行われる攻撃が問題となっている. この攻撃（以降, **単純攻撃**と呼ぶ）は, サービスの有無を確かめるためのポートスキャンと, ソフトウェア脆弱性（以降, 単に**脆弱性**と呼ぶ）に対する攻撃のいずれかもしくは両方によるものであり, ウイルス／ワームなどにより自動化され容易に実行が可能である. なお, ソフトウェア脆弱性に対する攻撃とは, ソフトウェアのバグへの攻撃であり, 攻撃対象の計算機の特権を得たりすることを可能とする. この攻撃は, 多くのサーバ／クライアントシステムが安全性の拠り所としているセキュリティアプリケーション (SSL (Secure Sockets Layer)[5], SSH (Secure SHell)[6]) に対しても有効である[7),8)] ことから, その影響は深刻である.

このような単純攻撃が有効であるのは, サービスの動作を確認する作業がある種のポートスキャンによって行われると, それを防ぐのが非常に難しいことや, 脆弱性への対策も様々な理由により困難であることが原因である. 後者の脆弱性への対策の基本はパッチの適用という単純な作業であるが, 計算機の再起動をともなったり, 既存アプリケーションの動作に影響を与えたりする場合があるため, 高いサービス継続性を必要とするサーバ計算機の管理業務においては, 迅速なパッチの適用が難しい状況もある. また, 最近のネットワーク機器ではネットワークの高速化に対応するため ASIC(Application Specific Integrated Circuit) などを利用してハードウェア実装されたものもあるが, この場合パッチの適用自体が非常に難しい. さらに, パッチの存在しない未知の脆弱性を利用した攻撃も起こりえる. 以上のような状況を鑑みると, 現状の対策ではカバーしきれていない部分があり, 脆弱性への攻撃の潜在的な危険性は極めて大きい.

この問題に対して我々は, TCP コネクションの確立を要求する SYN (SYNchronize) パケットに認証情報を格納し, それを利用して TCP コネクション確立時にアクセス制御を行う技術である TAP(TCP layer Application Protector) を提案している[1)~4)]. TAP は TCP コネクションそのものの確立を制御できることから, サービスの確認作業を防ぎ, さらにサービスの脆弱性に対する攻撃を事前に防御することができる. その結果, リモートアクセスやリモートメンテナンス用のサーバアプリケーションなどの安全性が高められ, 副次的効果として, パッチの適用に猶予を持たせることができることから保守コストを下げることも可能となる.

本論文では, 実装した TAP モジュールを利用した機能検証および性能評価の各種実験を行い, TAP の有効性を検証した. 機能検証では, インターネットのような実際の環境における接続性と, ポートスキャナによる探索行為に対するサービスの秘匿性を確認した. そして性能評価では, 通常の TCP 通信と比較したオーバーヘッドが問題とならないこと, さらに SYN パケットの大量送信攻撃に対して高い

† 株式会社東芝 研究開発センター コンピュータネットワークラボラトリ
Computer & Network Systems Laboratory, Corporate Research & Development Center, TOSHIBA Corporation

耐性をもち, サーバアプリケーションへの大量接続攻撃に対してはサーバアプリケーションの負荷を軽減する特徴を有していることを確認できた.

以降, まず第 2 章で TAP プロトコルの概要について, 次に第 3 章でその実装について, それぞれ説明する. 第 4 章では機能検証の実験結果について, 第 5 章では性能評価の結果について述べる. そして, 第 6 章で本論文をまとめる.

2. TAP の概要

2.1 背景原理

現在, 多く利用される TCP(Transmission Control Protocol) においてはクライアントがサーバに送信する SYN(SYNchronize) パケット, サーバがそれに応答して, クライアントへ送信する SYN/ACK(ACKnowledgement) パケット, それに答えてクライアントがサーバへ送信する ACK パケット, からなる 3 way handshake によってコネクションが確立され, その確立したコネクションを利用してクライアントとサーバとでデータを送受信する.

ここで注目すべき点は, サーバはクライアントからの SYN パケットを無条件に受け入れ, それに対して SYN/ACK パケットで応答している点である. 脆弱性を攻撃するためには, SYN/ACK パケットの応答をもとにサービスの存在を確認し, その後に TCP コネクションを確立することが必要であるため, SYN パケットの受信段階でクライアントを選別することで単純攻撃が防御できる. そこで TAP は, クライアントからサーバへの SYN パケットに格納した認証情報によりクライアントを認証し, 不正なクライアントへ SYN/ACK パケットを送信しない. これにより TCP コネクションの確立を制御し, サービスを秘匿・保護する. TAP を導入することにより, 攻撃者は脆弱性を攻撃することはおろか, SYN/ACK パケットを受信できないことから, そこにサービスが存在していることすら分からない (図 1). そのため, 万一 WWW サーバや SSL などの基幹アプリケーションに脆弱性があっても, 攻撃を受けるリスクを軽減できる.

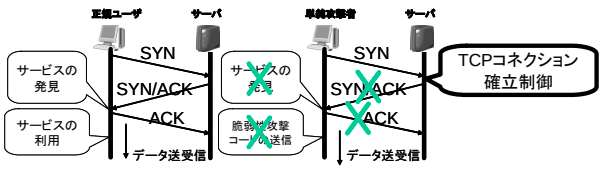

図 1 TAP 導入時の 3 way handshake

2.2 処理概要

SYN パケットによるクライアント認証は, 電子署名やメッセージ認証子などの既存の暗号技術を利用して, 認証情報を SYN パケット内の未使用フィールドに埋め込んだり, オプションとして付加したりすることで実現できる. しかしこういった変形パケットは通信経路上にある高機能な (インテリジェント) ルータ／ファイアウォールを通過できない危険がある. そこで TAP では, 確実にルータを通過するように構成した認証用 SYN パケットを, 本来の SYN パケットとは別に送信する (図 2).

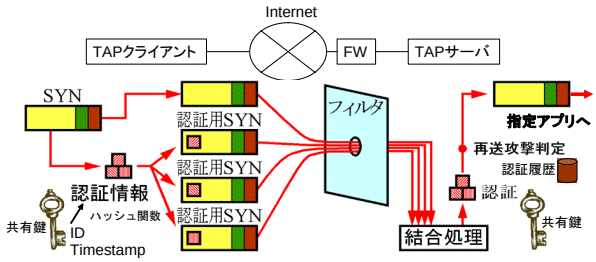

図 2 TAP の処理概要

この認証用 SYN パケットには, クライアントとサーバとの間で予め共有した共有鍵によって生成された認証情報を埋め込むが, ルータ等ではじかれる危険性を無くすために通常の TCP のそれと全く同じ形式とする必要がある. この制約下では, 一つの SYN パケットに埋め込むことのできる情報量が限られる. そこで, 認証情報を分割して複数の認証用 SYN パケットに埋め込み, サーバ側で認証情報を再構築した後に検証処理を行う. 具体的な手順は以下のようになる：

Step.1 SYN パケット送信：TAP クライアント (TAP のクライアント機能が実装されたプログラム) は TAP サーバ (TAP のサーバ機能が実装されたプログラム) に本来の SYN パケットとともに複数の認証用 SYN パケットに格納した認証情報を送信する. 認証情報は, TAP サーバと TAP クライアントとの間で共有した共有鍵によって生成されたメッセージ認証子や, リプレイ攻撃防止のための時間情報 (図 2 中の Timestamp) や複数の SYN パケットから認証情報を再構成するための情報からなる. 生成された認証情報は, 通信経路上にあるルータやネットワーク制御技術との整合性を考慮し, TCP ヘッダのシーケンス番号に埋め込んでいる.

Step.2 SYN パケット検証：TAP サーバは TAP クライアントから複数の SYN パケットを受信し, そこに格納された認証情報を TAP クライアントと予め共有した共有鍵を利用して検証処理を行う. ここで複数の SYN パケットを送信してきたクライアントの同一性は, 送信元 IP アドレスや他フィールドの値などを利用して判断する.

Step.3 SYN パケット通過：TAP サーバは TAP クライアントから受信した複数の SYN パケットの認証情報が正しい場合には, 受信した本来の SYN パケットを通過させる. その後, その SYN パケットに対する SYN/ACK パケットの返信処理が行われる. この際には, クライアントの送信した SYN パケットに対する応答処理として SYN/ACK パケットを返答する. そのため, 返信される SYN/ACK パケットは通信経路上にあるルータを問題なく通過できる上, クライアントも通常の SYN/ACK パケットとして受信でき, TCP コネクションを確立できる.

2.3 効果と安全性

攻撃者は TAP で保護されたサービスの存在に気がつかないため, そのサービスが攻撃にさらされる機会が減少する. また単純攻撃がなされた場合にも, コネクションの確立そのものが防止されているため, サービスへの攻撃が成功することはない. また, 仮に TAP の存在を知った攻撃者がサーバを手動で攻撃した場合でも, TAP プロトコルはリプレイ攻撃や認証情報の偽造に対して安全に構成されている.

リプレイ攻撃は, TAP クライアントが送信時点の時刻情報を認証情報に含めて送信し, TAP サーバが検証時に現在時刻を確認して SYN パケットの時刻情報とのずれが許容時間内にあることを確認して防御する. 仮に, 攻撃者が通信路の盗聴などにより過去の SYN パケットを取得した上で, その時刻情報を現在時刻に改竄して送信してきた場合でも, 認証情報を共有鍵で検証することにより検知できるため問題とならない. 認証情報の偽造は, その生成時に共有鍵を利用したハッシュ関数を適用することで防いでいる.

なお, TAP は TCP コネクション確立の安全性を向上させるものであり, 相互認証や通信路の機密性・完全性を提供するプロトコルではない. そのためセッションハイジャックや中間者攻撃といった攻撃は, TAP の上位層において, SSL や SSH などを併用して防止する.

2.4 関 連 技 術

SYN パケットに対する応答制御は, パケットフィルタリング技術として, ファイアウォール等でも用いられている. しかし, 提供するサービスによってはクライアント端末の IP アドレスが動的に変化するなど, パケットフィルタリングには限界がある.

また, SYN パケットを利用した認証処理としては, port knocking[9] などと呼ばれる運用技術が提案されている. このような技術は, 認証情報を格納した複数の SYN パケットを受信した際に, TCP コネクションを確立するための SYN パケットを受け入れるようフィルタリングルールを変更するものである. それに対し TAP は, 認証情報を格納した SYN パケットに対する SYN/ACK パケットの返答処理を制御するものであり, TCP コネクションの確立処理と認証処理を一体化させることで, 攻撃者が攻撃を仕掛ける余地を減らし安全性をさらに向上させている.

この他にも, サービスの脆弱性対策としては, 侵入検知システムにより攻撃用のコードが送信された段階で防御する方法もある. しかし, この方法では未知の脆弱性を完全には防御できない. この他に, サーバ計算機に施す対策として, プログラム実行時に異常を検知し, プログラム実行を停止する方法も存在する. しかしながら, この場合はサービス自体の動作も停止させてしまい, 正規のユーザにも影響が生じるなどサービス継続性に影響を与えることがある.

3. 実　　装

現在までに, TAP サーバ, TAP クライアントともに試作モジュールが完成済みである. TAP サーバは, linux 版のみ, TAP クライアントは, BSD, linux, Windows® ☆ 版が存在している. linux 版 TAP システムは図 3 のように構成されており, サーバ, クライアントともに linux2.4.x,2.6.x に対応している.

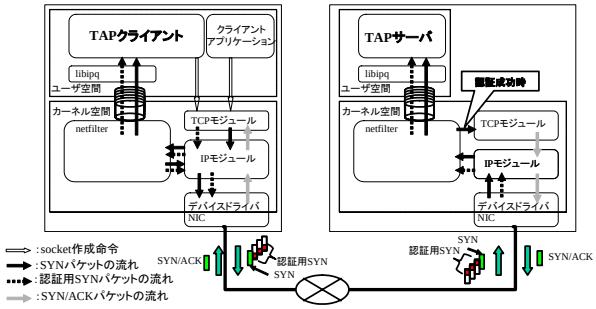

図 3　実装の構成図

netfilter[15]：linux のカーネルコードのネットワークスタック内部に作りこまれたフックの集合で, パケットがこれらのフックを行き来するたびに, 登録されたコールバック関数が呼び出されてこれを処理する. この際に, カーネル内部で保持しているフィルタリングルールを参照しており, このテーブルの編集は, iptables[15] を利用する.

libipq：フィルタリングルールにマッチしたパケットをユーザ空間のプログラムから取得し, そのパケットの処理方法を決定する機能を提供するライブラリ. iptables とともに提供されている.

TAP サーバ, TAP クライアント：netfilter の機構を利用してユーザ空間のサーバプログラムとして実装. netfilter でフィルタした SYN パケットを libipq を利用してカーネル空間から取得し, 2.2 の処理を実行する.

4. 機 能 検 証

第 3 章で示した linux 版の TAP サーバおよびクライアントモジュールを利用して機能検証を行う.

4.1 接続性の検証

TAP の設計にあたっては途中経路のネットワーク機器の影響によりネットワーク上での接続性が低下してしまうことを防ぐことを目標の一つとした. その接続性を検証すべく, 各種プロバイダ上に配置した TAP サーバに向けて, 各地より TAP プロトコルによる接続を試みる接続実験を行った [2]. その結果, これまでのすべての実験環境において TAP 接続が成功しており, TAP 設計の接続性の高さが確認できたと考える.

4.2 サービス秘匿効果の検証

代表的なポートスキャンツールである nmap[16] を利用して TAP サーバへのポートスキャンを実施し, 通常は防ぐことが難しい TCP コネクションスキャン, TCP ハーフスキャンの 2 つのポートスキャンに対して, サービスの存在が秘匿されることを確認した. これらのスキャンは, 実際に

☆ Windows は, 米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です

オープンしているポートに対する通常の TCP コネクション確立作業を利用するので, 正当な接続要求との判別が難しい. まず, 上記２つのポートスキャンについて簡単に解説する:

(1)TCP コネクションスキャン：実際に調査対象のポートに対して接続し, TCP コネクションを確立することでサービスが提供されているか調査する方法.

(2)TCP ハーフスキャン：3 way handshake においてサーバが送信した SYN/ACK に対して RST を送信し, コネクションの確立は行わないことでログに残りにくくしたスキャン方法.

さて, TAP が SYN パケットを利用した認証により上記のスキャンからもサービスの有無を秘匿できることを以下に示す. まず, TAP サーバを起動させない状況で TCP ハーフスキャン (nmap -sS), TCP コネクションスキャン (nmap -sT) を実行すると, ポート 22 番で SSH のサービスが起動していることが判る.

```
# nmap -sS 192.168.0.6
中略
Port       State       Service
22/tcp     open        ssh
中略
# nmap -sT 192.168.0.6
中略
Port       State       Service
22/tcp     open        ssh
```

次に TAP サーバを 22 番を保護するように起動させた上でスキャンを試みると, ポート 22 番がフィルタリングされている (filtered) と出力される.

```
# nmap -sS 192.168.0.6
中略
Port       State       Service
22/tcp     filtered    ssh
中略
# nmap -sT 192.168.0.6
中略
Port       State       Service
22/tcp     filtered    ssh
```

フィルタリングルールの設定は許可パケットのみを記述するホワイトリスト形式が一般的であるので, 攻撃者には当該ポートのサービスの有無についての情報を得ることはできない. さらに, TAP クライアントを起動させてポートスキャンを行った結果を示す.

```
# /usr/sbin/TAPclientd
# nmap -sS 192.168.0.6
中略
Port       State       Service
22/tcp     open        ssh
中略
# nmap -sT 192.168.0.6
中略
Port       State       Service
22/tcp     open        ssh
```

TAP は SYN パケット認証後の TCP フローには関知しないので, 通常時と同じ nmap 結果が得られる. 以上の結果から, TAP クライアント以外からのコネクション確立要求を制御し, サービスの有無を秘匿する様子を示した.

4.2.1 その他のポートスキャンについて

4.2 で秘匿効果を確認した TCP コネクションスキャン, TCP ハーフスキャンに加えて, X-Mas スキャン, Null スキャン, FIN スキャン, ACK スキャン, などのスキャン方法が存在する. これらのスキャン方法は特殊に構成された構造をもつパケットを利用してスキャンを行う. これらのスキャンは, パケットフィルタリングルールを適切に設定することで防止可能である. 実装においては, TAP サーバ起動時にこれらのスキャンを防止するためのフィルタリングルールを netfilter に追加するので問題とはならない.

5. 性 能 評 価

第 3 章で示した linux 版の TAP サーバおよびクライアントモジュールを利用した性能評価の結果を提示する.

5.1 基本性能の測定

TAP の基本的な通信性能を測定し, 通常の TCP と比較する. そのためにネットワークベンチマークツールである netperf[10] を用いて，サーバ計算機 S とクライアント計算機 C 間の TAP 通信のトランザクション数およびスループットを計測する.

実験環境：S と C を，スイッチングハブ $Hub1$ を介して接続した (図 4). S のスペックは，OS:linux2.4.18, CPU:Pentium4 2.4GHz，メモリ:512MB であり, C のスペックは， OS:linux2.4.18，CPU:Celeron 1.7GHz，メモリ:256MB である. ハブと計算機は，100Base-TX で全二重通信が可能である.

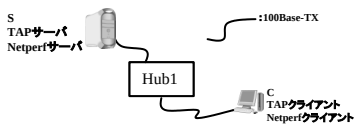

図 4 実験環境

測定結果：同一条件での通常の TCP での結果と比較しても, TAP 通信によるオーバヘッドは極めて軽微であると分かる：

	TAP 有	TAP 無
トランザクション数 (回/秒)	1,618	1,627
スループット (kbit/秒)	94,128	94,129

なお, ここでトランザクション数は, コネクションを毎回確立し, $C \rightarrow S$:32byte, $S \rightarrow C$:1024byte のデータ転送後にコネクションを切断する処理を 1 回として測定している.

5.2 DoS 攻撃への耐性

TAP は TCP 層での接続制御方式であるため, その存在がサーバのボトルネックとなる可能性がある. そこで, TAP サーバに大量アクセス型の DoS 攻撃をしかけてその耐攻撃性能を確認する. さらに, サーバアプリケーションへ大量アクセス攻撃が仕掛けられている環境において, TAP サーバがサーバアプリケーションの負荷を軽減できることを確認する.

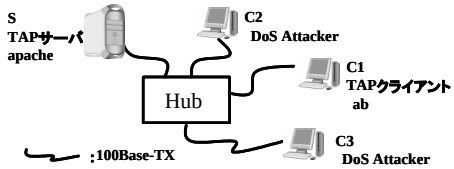

図 5 実験環境

5.2.1 実験環境

サーバ計算機 S とクライアント計算機 $C1$, $C2$, $C3$ を, スイッチングハブを介して接続した (図 5). S のスペックは, OS:linux2.4.18, CPU:Pentium4 2.4GHz, メモリ:512MB であり, $C1$, $C2$, $C3$ のスペックは, OS:linux2.4.18, CPU:Celeron 1.7GHz, メモリ:256MB である. ハブと計算機は, 100Base-TX で全二重通信が可能である. そして, S には TAP サーバ及び Web サーバソフト apache[17] を, $C1$ には TAP クライアント及び Web サーバベンチマーク用ソフト ab (apache bench)[17] をそれぞれ導入する. $C2$, $C3$ には悪意をもって大量接続を行う負荷発生用の計算機として利用する.

5.2.2 耐 DoS 基本性能の測定

TAP で保護されたサービスは基本的に攻撃者に存在を知られることはない. しかしここでは敢えて, TAP によって保護されたサーバに対する大量アクセス型の DoS 攻撃を想定した実験をおこなった. この実験では, TAP によって保護された Web サーバの存在を知り得た攻撃者が, その可用性を損なうために大量アクセス型の DoS 攻撃を行った状況を仮定した. この状況をシミュレートするために, ソース IP アドレスを偽装した SYN パケットを大量送信する SYN Flood 攻撃により TAP サーバを導入したサーバ計算機に負荷を与えた. そして, その状況で TAP を利用した Web 閲覧の成功確率を測定した. ここで, SYN Flood 攻撃で負荷を与えるのは, TAP は TCP の SYN パケットのみを処理対象とする機構であるため, 送信コストの小さい SYN パケットの大量送信が最も強力な攻撃方法であることを考慮した.

実験概要: 5.2.1 の環境において, $C2$, $C3$ には SYN Flood 攻撃ツール (synk4 を改良して流量調節機能を持たせたもの) を導入する. そして, $C2$, $C3$ が S に向けて SYN Flood 攻撃を実施している中で, $C1$ は S に TAP を利用した Web 閲覧 (ab を利用) を並列に 100 回試みて, その成功確率を計測した結果を図 6 に太線で示す. ここで, 横軸は毎秒あたりに $C2$ と $C3$ が送信する SYN パケットの合計数 (pps) を, 縦軸は TAP を利用した Web 閲覧の成功確率 (%) を表す. なお, 実験では実ユーザの感覚に近づけるために, SYN パケット送信のタイムアウトをデフォルトの 180 秒から 45 秒に変更している☆.

実験結果: 46,000pps 程度の SYN Flood 攻撃下でも TAP を利用した Web 閲覧がほぼ 100%成功する. この値は, TAP 未動作時の S において 100pps 程度の SYN Flood 攻撃で接続不能状態に陥る (コネクションキューの制限値による) ことと比較しても極めて安定した動作ができることを示している. 46,000pps 以降は成功確率が徐々に低下し, 80,000pps の SYN Flood 攻撃では接続不能となる. また, S の TAP サーバを停止し, SYNCookie☆☆を有効にした場合の結果を図 6 に細線で示している. TAP サーバでは netfilter, libipq を利用することによるオーバヘッドがあることを考慮すると, 実装によっては TAP サーバの接続成功確率を向上させることは十分可能であり, SYN Flood 攻撃への有効な対策技術である SYN Cookie と同等レベルの高い耐 DoS 攻撃性能をもつことが確認できた.

☆ /proc/sys/net/ipv4/tcp_synretries を変更

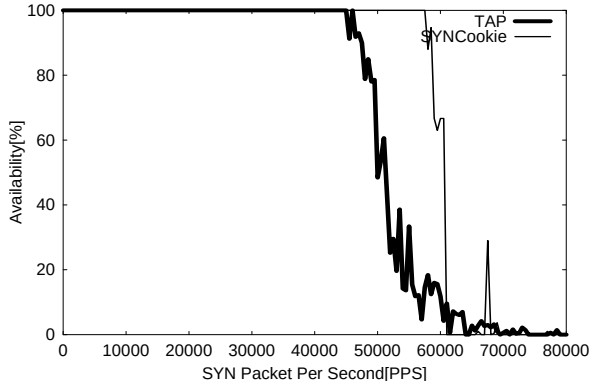

図 6 接続成功確率の推移

5.2.3 HTTP サーバへの適用時の効果

TAP は TCP 層での認証処理のためアプリケーション層は TAP を通過した接続要求のみを処理すればよい. そのため, アプリケーションに対して大量の接続要求による攻撃が想定されるシステムであっても, その負荷を軽減させることが可能である. この効果を確かめるために, TAP によって保護された Web サーバに対して, 大量の HTTP 接続要求を行った状況を想定した実験を行う.

実験概要: 5.2.1 の環境において, $C1$, $C2$ にも ab を導入する. そして, $C2$, $C3$ が S に向けて大量の HTTP 接続を行っている中で, TAP を利用する場合と利用しない場合のそれぞれで $C1$ は S に Web 閲覧 (ab を利用) を並列に 100 回試みて, その単位時間当たりのリクエスト処理数 (Requests per Second) を計測した (図 7). ここで, 横軸は $C2$ と $C3$ の合計の同時 HTTP 接続要求数を, 縦軸は $C1$ の Web 閲覧結果から算出された単位時間当たりのリクエスト処理数 (rps) を表す.

実験結果: TAP サーバが有効な状況では同時 HTTP 接続要求数の変化によらず, そのリクエスト処理数は 650 程度で安定している (図 7 中の太線). それに対して, TAP を無効化した場合には同時 HTTP 接続要求数が増加するに従ってリクエスト処理数は急激に減少していく (図 7 中の細線). この結果から, アプリケーションへの大量接続攻撃に際して, TAP サーバが上位層のアプリケーションの負荷を軽減する効果はかなり大きいことが確認できた. この効

☆☆ linux のカーネルに実装された SYN Flood 攻撃の対策技術. サーバが TCP コネクション用リソースを確保するタイミングを ACK 受信時とすることで、SYN/ACK 待ちによるリソース枯渇を防ぐ

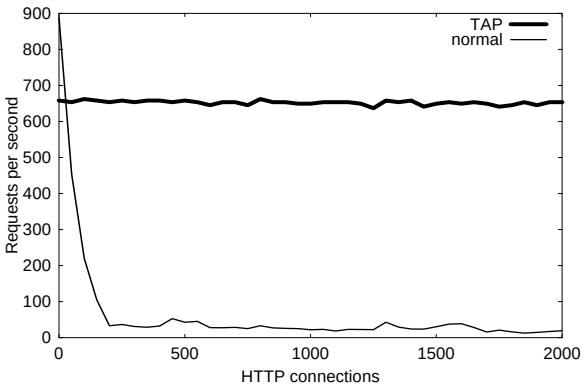

図 7 大量の HTTP 接続要求状態でのリクエスト処理数の推移

果は, 上位層のアプリケーションの処理が重くなるほどに顕著になると推定でき, SSL や SSH などのセキュリティアプリケーションの負荷を軽減する効果が期待できる.

5.2.4 SYN Flood 攻撃防御技術との比較

5.2.2 では, SYN Flood 攻撃に対する耐性を測定した. その中で SYN Cookie を比較対照として提示した. その他の SYN Flood 対策技術としては, SPP(SYN Packet Pacifier)[11] なども存在し, これらは SYN Flood 攻撃への高い耐性をもつ. しかしながら, 実際に TCP コネクションを確立する 5.2.3 のようなアプリケーションへの負荷攻撃が行われた場合にはそれを防ぐことはできない. TAP を利用すればそれらの攻撃を同時に防止することもできる.

6. ま と め

本論文では, TCP 層でアクセス制御を行う技術である TAP について解説し, 機能検証と性能評価を通じて有効性検証を行った. 機能検証の結果, TAP がインターネットのような実環境でも利用可能であり, そのサービス秘匿機能が有効に動作することが確認できた. そして, 性能評価の結果, 通常の TCP と比較したオーバヘッドは無視できるほど小さく, SYN Flood 攻撃に対しては高い耐性を有することが確認できた. そして, TCP 層においてアプリケーションへの同時アクセス数を正規ユーザ数程度に見積もることができることから, 大量のサーバアプリケーションへの DoS 攻撃下においてもその負荷を軽減できることが示された. 以上の結果, 本方式の有効性を示すことができた.

今後, セキュアシステム構築における一つのコンポーネントとしての本方式の有用性を確認したい. また, TAP サーバの機能をパケットフォワード部に入れることでルータ機器として動作させた場合の性能評価も併せて行ってゆく. さらに, アプリケーションプロキシなどが存在する環境では, TCP コネクションが分断されるため本方式は適用できない. このような環境での脆弱性対策についても別途検討していきたい.

参 考 文 献

1) 梅澤健太郎, 高橋俊成, 鬼頭利之, “サービスを秘匿する TCP コネクション確立方法の提案”, 暗号と情報セキュリティシンポジウム 2004, Jan 2004.
2) 高橋俊成, 梅澤健太郎,“TCP レイヤでの攻撃防御実験 1 - 概要と接続実験 ”, 電子情報通信学会全国大会 2005, Mar 2005.
3) 梅澤健太郎, 高橋俊成,“TCP レイヤでの攻撃防御実験 2 - 耐攻撃性の評価 ”, 電子情報通信学会全国大会 2005, Mar 2005.
4) “インターネット上のサーバを攻撃から守る通信方式”, 東芝レビュー.60, 3, 2005, p.32.
5) A.O.Freier,P.Karlton,P.Kocher,“The SSL Protocol Version 3.0 ”, Internet Draft, Transport Layer Security Working Group, Nov 1996.
6) T.Ylonen,“The SSH (Secure Shell) Remote Login Protocol ”, Internet Draft, Network Working Group, Nov 1995.
7) CERT Advisory CA-2003-24 Buffer Management Vulnerability in OpenSSH, http://www.cert.org/advisories/CA-2003-24.html, Sep 2003.
8) CERT Advisory CA-2003-26 Multiple Vulnerabilities in SSL/TLS Implementations , http://www.cert.org/advisories/CA-2003-26.html , Oct 2003.
9) M.Krzywinski, “Port Knocking - Network Authentication Across Closed Ports”,SysAdmin Magazine.10,6,2003, p.12-17.
10) “The Public Netperf Homepage”, http://www.netperf.org/
11) 伊藤大輔, 泉裕, 齋藤彰一, 上原哲太郎, 國枝義敏, “TCP セッション管理による DoS 耐性の考察”, インターネットコンファレンス 2002, Nov 2002.
12) H.Krawczyk, M.Bellare, R.Canetti, “HMAC: Keyed-Hashing for Message Authentication”, RFC2104, Feb 1997.
13) W. Richard Stevens, “詳解 TCP/IP Vol1. プロトコル”, ピアソンエデュケーション, 2000.
14) Gary R.Wright and W. Richard Stevens, “詳解 TCP/IP Vol2. 実装”, ピアソンエデュケーション, 2002.
15) “netfilter/iptables project homepage”, http://www.netfilter.org/
16) “Nmap -Free Security Scanner For Network Exploration & Security Audits”, http://www.insecure.org/nmap/
17) “Welcome! -The Apache HTTP Server Project”, http://httpd.apache.org/