ＴＣＰセッション管理によるＤｏＳ耐性の考察

伊藤大輔[†]　泉裕[††]　齋藤彰一[‡]　上原哲太郎[‡]　國枝義敏[‡]
[†]和歌山大学大学院 システム工学研究科　[††]和歌山大学システム情報学センター
[‡]和歌山大学システム工学部

近年のネットワーク管理において，DoS（Denial of Service）攻撃対策は重要な課題の一つである．DoS 攻撃は，IP アドレスを偽造した TCP パケットを被攻撃者に送信し，過剰処理を行わせることでサービスを妨害する．DoS 攻撃が成功する根本的な原因は，被攻撃者が IP アドレスを元に攻撃者を判断するためである．したがって，IP アドレスによる従来のフィルタリングでは，DoS 攻撃を効果的に抑止できない．

本研究の目的は，偽造パケットによる DoS 攻撃から被攻撃者を守ることである．我々は，TCP 3WayHandShake において，IP アドレス等を要素として生成した特殊なシーケンス番号による送信者の特定と，セッション確立の際にフィルタリングを行う手法を考案した．本手法では，特定された送信者からのパケットのみ受け取ることで，カーネルへの過剰処理を抑止し，効果的な DoS 攻撃対策を実現できる．さらに本論文では，本手法がルータやスイッチに適用し拡張性を持つことについて考察する．

The Research of The TCP Session Control against DoS Attack.

Daisuke Ito[†]　Yutaka Izumi[††]
Shoichi Saito[‡]　Tetsutaro Uehara[‡]　Yoshitoshi Kunieda[‡]
[†]Graduate School of Systems Engineering, Wakayama University
[††] Center for Information Science, Wakayama University
[‡]Faculty of Systems Engineering, Wakayama University

It becomes rapidly important to protect against DoS Attack in the network management. DoS Attack transmits the TCP packet forged its IP address to the target server(s), then drop its performance because of high overload. It is difficult to avoid DoS attack, because the recent filtering technology almost depend on IP address, and we cannot specify the attacker by IP address forged. In this paper, we suggest the effective method which purpose is protection against DoS attack using IP address forged. This method works as specifying the source by the sequence number generated by above IP address and so on, then filtering by its sequence number. This paper shows one of the method of effective TCP session control against DoS attack for deterrence of high overload in server(s).

Futhermore, we show adapting above method to the network manufacture (e,g Router, L3 Switch) and its extensibility.

１．はじめに

近年のネットワーク基盤拡充に伴い，インターネットにおける不正アクセスへのセキュリティ対策が急務となっている．様々な組織でセキュリティレベルの向上が急速に図られる一方，不正アクセスの傾向も「いかにホストに侵入するか」から「いかにホスト（ネットワーク）にダメージを与えるか」に移行してきている．後者にはウィルス（ワーム）と並び DoS（Denial of Services）攻撃があるが，現在は特に DoS 攻撃への対策が希薄である．カリフォルニア大学の試算では，１週間に世界で発生する DoS 攻撃の被害が 4000 件にも及んでおり[1]，DoS 攻撃を防ぐ有効な手段が確立していないと推測できる．DoS 攻撃には様々な手法がある．例えば，Nimda，CodeRed，Klez に代表される自己増殖ワームは，ショートパケットを大量発生させて間接的に DoS 攻撃を引き起こす．一方，意図的かつ直接的な DoS 攻撃には，特定ポートへの SYN Flood や，主としてポートスキャンで使用される XMAS/NULL 等のスキャンを適用した攻撃，ホスト内 IDS（侵入検知システム）等フィルタ機能の負荷を圧迫させる攻撃などがある．したがって我々は DoS 攻撃対策を総合的な視点で早急に検討する必要がある．本論文では TCP セッションを従来よりも厳密に管理することで，直接的な DoS 攻撃への耐性を高める一手法を検討し，システムの設計構築および評価について述べる．さらに総合的な DoS 攻撃対策について考察する．

２．DoS 攻撃

2.1.DoS 攻撃の概念

DoS 攻撃とは，サーバで提供しているサービスを妨害し，サービスの利用不能状態あるいはサービス機能の著しい低下を引き起こす攻撃方法である．サーバへの直接的な DoS 攻撃としては，サーバマシン内の OS やサーバプログラムをハングアップさせる攻撃と，ソケット資源の浪費やサーバ内負荷の意図的な増大によって実質的に利用不能にする攻撃がある．前者の例に

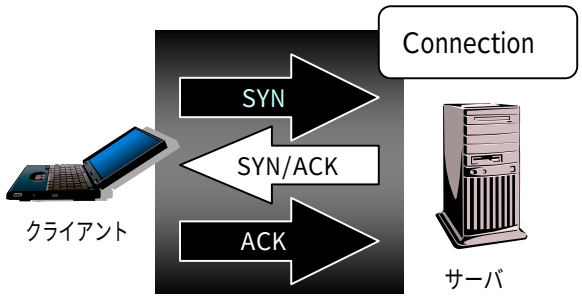

図 1 3WayHandShake

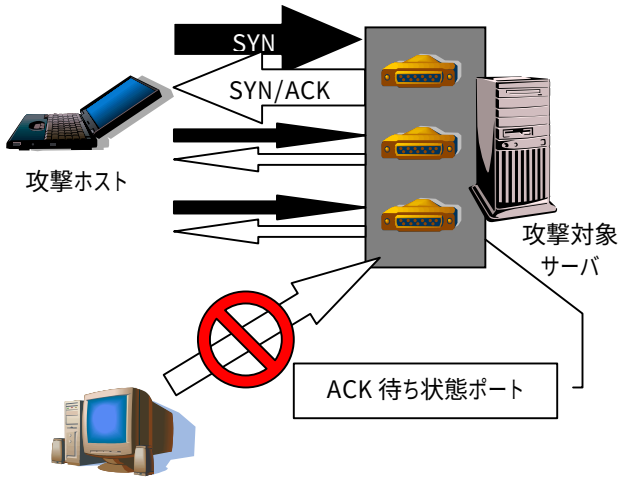

図 2　SYN Flood

は Out Of Bounds パケットを送信することで特定 OS の機能を停止させる OOB 攻撃がある．一般には NUKE や規格外の ICMP による Ping of Death がある．しかし，近年多様化し問題となっているのは後者による DoS 攻撃である．本来後者の DoS 攻撃にはブロードキャストを悪用した Smarf やネットワーク帯域等の大量消費を狙う各種 Bomb 攻撃が含まれる．中でも，特に TCP セッションを悪用した攻撃は効率が良いと考えられている．これは Bomb 攻撃等が，攻撃側のデータ送信能力や帯域等，攻撃能力についてスケーラビリティを確保する必要があるため，DDoS（Distributed DoS）を含めて実行に困難が伴うためである．

TCP によるコネクションを確立するには 3WayHandShake（図１）が必須である．しかし 3WayHandShake を不完全な状態で放置する SYN Flood は代表的な DoS 攻撃の一つである．SYN Flood 攻撃は SYN フラグを立てた大

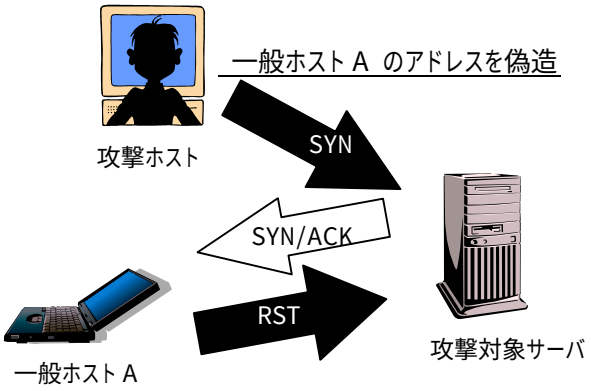

図3 送信元を偽った SYN Flood

量のパケット（以下 SYN パケット）を攻撃対象サーバに送信して待機状態を意図的に作りだす．結果，サーバのリソース（ソケット等）を不正に消費させて他ホストとの通信を困難にすることが可能である（図２）．さらに SYN Flood 攻撃では TCP セッションの確立が完了しないので，各種 OS やサーバプログラムのログ機能では SYN Flood を正確に検出することは不可能である．

IDS（侵入検知システム）を導入しているサーバやネットワークでは，一般的に SYN Flood を検出し，攻撃元を特定することも原理的に可能である．しかし実質的な防御との連携が困難であり，攻撃元の IP アドレスを詐称された場合には（図３），攻撃の事象だけが不正アクセスの統計に反映されるのみである．さらにポートスキャンで用いられる XMAS スキャンや NULL スキャン等，様々なフラグパターンについても同様である．一方，IDS が不正アクセスと判断する各条件に合わせた様々なアクセスパターンを送信することで，IDS フィルタ機能を負荷で圧迫する対 IDS 攻撃も存在する．対 IDS 攻撃は Stick[2]のように各種 IDS の持つ条件に依存するため，汎用的では無いと考えられるが，攪乱によるセキュリティサービスの実質的な利用不能を引き起こすため，憂慮すべき攻撃手法である．

上記のように，現有システムの設定等で得られる副次効果を期待して，DoS 攻撃に対する根本的な耐性強化を図ることは困難である．

2.2.DoS 攻撃への対策

DoS 攻撃への具体的な対策を考慮したシステムも存在する．ルータを含む Firewall 機器と連携する NetRanger[3]のような特定の IDS は，不正アクセスと認められるアクセスパターンが検出された場合，送信元からのデータ通信を拒否するよう Firewall のフィルタ条件を動的に変更する．しかし送信元 IP アドレスを詐称された場合には効果が極端に薄くなる．さらに大量の DoS パケットが Firewall のフィルタ機能自体を負荷によって圧迫させると，組織内ネットワーク全体に影響する．

送信元 IP アドレスを詐称した DoS 攻撃への対策として，ルータ等ネットワーク上の中継機器間で実際の経路をさかのぼり，送信元を特定する iTrace（ICMP Traceback[4]）が検討されている．しかし経路上すべての中継機器にプロトコル実装させる必要があり，導入および連携を含めて現実的な解決は困難である．

したがって，現時点における現実的な解として我々は，サーバへの直接的な DoS 攻撃を対象とし，TCP セッション管理によって DoS 攻撃を効率よく排除するシステム構築が必要であると考える．TCP セッション管理では IP アドレスおよびサービスを提供するポートによるフィルタリングだけでなく，本来 TCP セッションを確立させるプロシージャに着目する．

TCP レベルでの DoS 対策を考慮したシステムとして，FreeBSD や Linux など UNIX 系 OS には SYNCookies や SYNCache が存在する．サーバクライアント間の TCP セッション管理という本研究の方針に対し，SYNCookies や SYNCache はサーバのカーネル空間における TCP 処理の一部を変更している．SYNCookies や SYNCache には SYN Flood への耐性は十分にあると考えられるが，TCP active/passive Open のように上位層プロトコルによって挙動が違うサービス（http,telnet または ssh,smtp 等）への対応や，ネットワーク中継機器への適用を含む拡張性に問題がある．本論文では主と

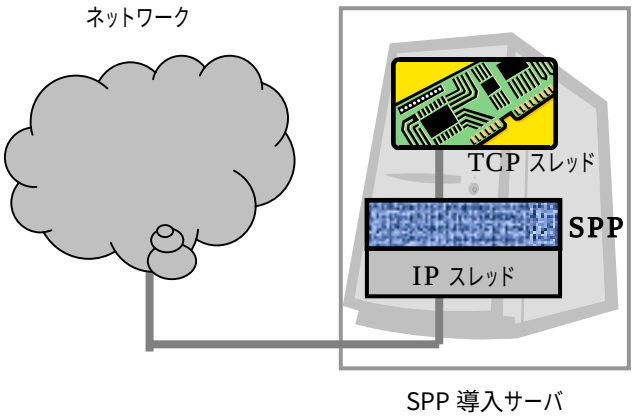

図4 SPP イメージ

して SYNCookies や SYNCache を比較の対象とし, 定量的な DoS 攻撃耐性および拡張性について評価する.

３．SPP の構築

3.1.目的

本研究ではホストへの直接的な DoS 攻撃に対する防御手法の提案と耐性評価を目的とする. 特に TCP レベルの DoS 攻撃を対象とし，主に SYN Flood，ステルス型ポートスキャンに含まれる攻撃, および対 IDS 攻撃からの防御をシステム機能要件とする. 自己増殖ワームによる間接的な DoS 攻撃の多くは，TCP コネクションの確立を前提としているため, TCP セッション管理による根本解決は困難である. 本論文では間接的な DoS 攻撃への対応について, 上位層との連携を考慮した統合システム構想の概要を考察の章で述べる.

3.2.SPP の概要

本研究で構築したホストベースの SPP（SYN Packet Pacifier）では，現在多くのフィルタシステムで用いられる概念を逆に適用する. すなわちフィルタに明瞭に該当するアクセスパターンを排除して他を通過させる[5]のではなく，IP アドレス詐称の疑いが皆無と思われるアクセスのみに TCP セッション確立後上位層とのアクセスを許可する. SPP のイメージを図４に示す.

SPP は IP 層で実装し機能する. 本来 TCP 層においてセッションは確立されるが, TCP 層をラッピングする IP 層において 3WayHandShake を確立させた後 TCP 層とのセッションを確立させる. 本手法によって, IP アドレスを詐称した各種 DoS 攻撃を防ぐことが可能である. さらに本来の TCP 層での挙動は保持されるため TCP active/passive Open における問題は発生しない. 加えて将来的にはネットワーク中継機器への実装が原理的に可能である. ポート番号で設定する各種上位プロトコルに対応したゲートウェイの挙動をシミュレートすれば，組織内ネットワーク対外接続部分への適用も可能である.

我々は IP 層における SPP をカーネルのソースコードにモジュールとして add-on できるよう実装している. したがって，カーネルの再構築が必要であるが SPP 導入は容易であると考える. 次節では構成と挙動について述べる.

3.3.SPP の構成と挙動

送信された TCP パケットに SYN フラグが立っていた場合, SYN Flood であるか正規の接続要求であるかは, TCP セッションの確立によって判明する. TCP セッションの「確立完了」は，通信を望む送信元が確実に存在し，なおかつ通信の意思があることを意味するため, SYN Flood である可能性はない. 一方で SYN Flood の場合にはセッションは確立しない.

セッションの確立が確定するのは，3WayHandShake において，ACK フラグの立ったパケット（以後，ACK パケット）が送信された場合である（図５）（以後，セッション確立の際の ACK パケットを“AU_ACK”とする）. すなわち，結果的に 3WayHandShake の監視を行うことで，SYN Flood の検出は可能である.

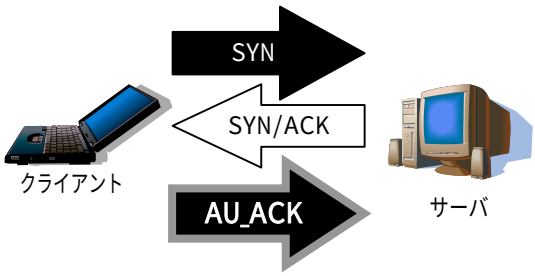

図 5　通常の 3-WayHandShake

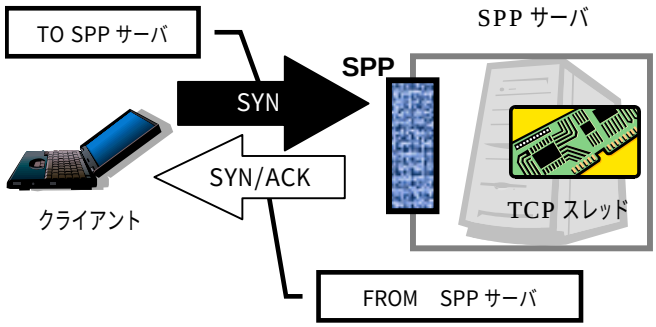

図 6　SYN パケット受信時

通常，サーバカーネル内の TCP スレッドが 3WayHandShake を監視するため，攻撃時にはカーネルの機構全体に直接負荷がかかり，DoS 攻撃が成立することになる．本案では，SPP が監視を行うことで，SPP 導入サーバ（以後 SPP サーバ）のカーネルへの負荷を軽減する．

3.3.1.通信までの挙動

SPP は，クライアントと TCP スレッドとの通信を 3 段階の手順を踏んで実現する．それぞれの段階を以下に記述する．

Level1.クライアントとのコネクション確立

クライアントから SPP サーバ宛に送信されたパケットが SYN パケットであった場合，SPP は SYN パケットを TCP スレッドへリレーしない．一方で，SPP サーバの IP アドレスを送信元アドレスとした，SYN/ACK フラグが立ったパケット（以後，SYN/ACK パケット）を生成し，クライアントへ送信する（図６）．パケットを受信したクライアントでは，SPP サーバが接続要求を受け付けたように見える．

仮に，送信されたパケットが SYN Flood による偽造パケットであった場合でも，攻撃対象である SPP サーバのカーネル（TCP スレッド）には，パケットが到達していないため，必要以上に負荷がかかることはない．また，送信する SYN/ACK パケットのシーケンス番号は

- クライアントのアドレス
- 双方のポート番号
- クライアントの ISN
  (Initial Sequence Number)

などのエレメントから生成した SecretFunction により生成し，クライアントに送信したパケットのリスト（通常はポート番号，シーケンス番号など TCB（TCP Control Block）への記録・接続用のメモリ割り当て）は生成しない．これは SYN Flood であった場合に多くのカーネルメモリ領域が枯渇することへの対策である．

Level2.カーネルとのコネクション確立

送信した SYN/ACK パケットに対する ACK パケットを受信した場合，通常ならば TCP スレッド（tcp_input()関数）で TCB の中から条件に合致するもの（IP アドレスやコネクションの状態が“SYN RECVD”であるかなど）を検索し処理する．しかし前述のように，SPP ではカーネルメモリ領域の枯渇を防ぐため TCB に該当するものは持たない．つまり，SPP は受信した ACK が果たして過去に送信した SYN/ACK に対するものであるかを過去の記録から判別することは不可能である．しかし，SPP は受信した ACK パケットのシーケンス番号の再計算を行うことで，ACK パケットの正当性（過去に SYN/ACK を送信した相手か）を確認することが可能である．正当性が認められなかった場合には，不正パケットと判断してパケットをドロップする．正当性が認められた，つまり AU_ACK であると判断された場合には，TCP スレッドに対し，送信元 IP アドレスをクラ

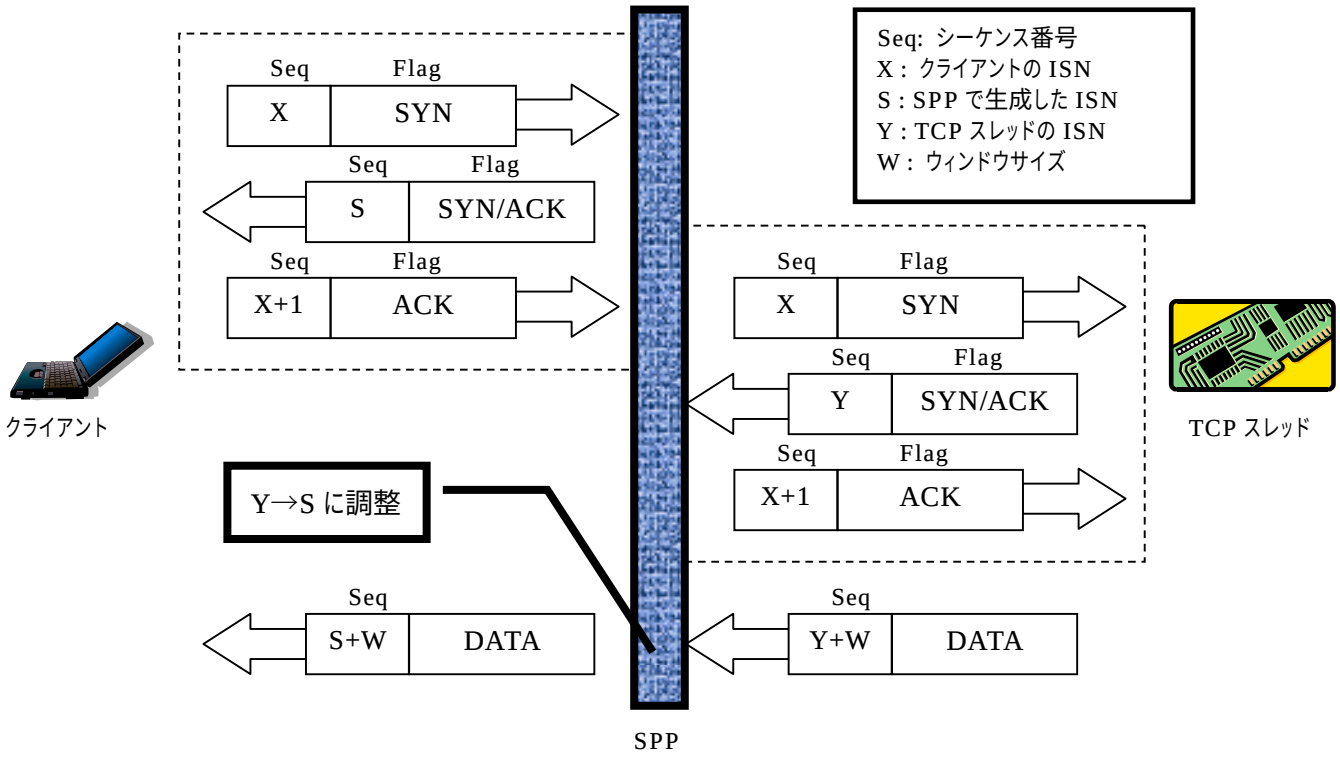

図8　シーケンス番号の関係と調整

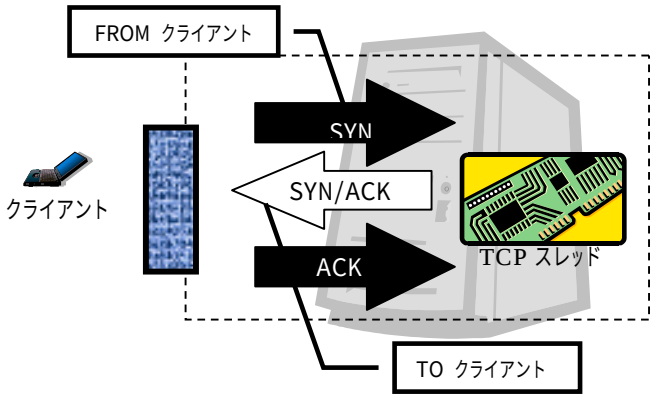

図 7 SPP サーバ内におけるｾｯｼｮﾝの確立

イアントのアドレスとしたパケットを生成し，今度は SPP 自身がクライアントであるかのように振る舞い, TCP スレッドとの間にコネクションを確立する（図７）.

Level3.通常通信

SPP と TCP スレッドとの間でセッションの確立が成功した場合, SPP では対象のクライアントを Access Control List（以下 ACL）に登録する. ACL は SPP 内に存在するリストで, TCP スレッドとのセッション構築を許可したクライアントのデータのみが格納されている.

なお ACL には

- 送信元 IP アドレスとポート番号
- 受信先 IP アドレスとポート番号
- シーケンス番号の差分

などのパラメータを登録する.

以後は SPP が ACL に該当するクライアントから送信されたパケットのみを TCP スレッドへとリレーさせるが，リレーの際にはシーケンス番号の差分が必要になる. SPP がクライアントへ送信する SYN/ACK パケットのシーケンス番号（ISN）は SecretFunction により SPP が独自に生成した番号であり，SPP が TCP スレッドとコネクションを開設する際には，別のシーケンス番号が使用されることになる（図８）. したがってリレーの際には，クライアント側と TCP スレッド側のシーケンス番号の調整を行うためのパラメータとして，シーケンス番号の差分が必要になる.

SPP を挟んでクライアントと TCP スレッドの間にコネクションが確立した後に受信したパケットについては, ACL の内容と一致するものであれば，内部へのリレーを許可する.

しかし, 前述の３段階を踏んでいない ACK パケ

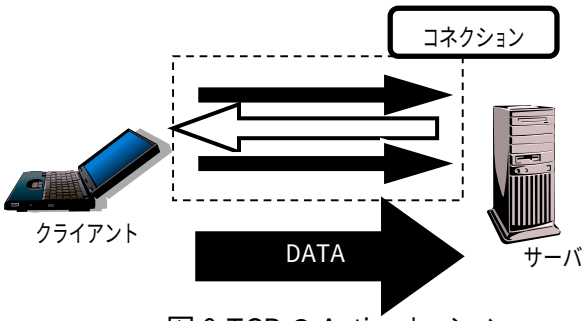

図 9 TCP の Active セッション

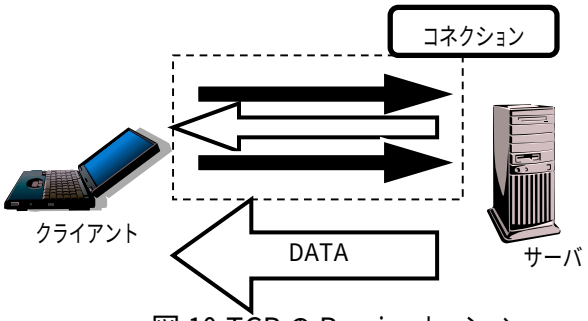

図 10 TCP の Passive セッション

ットや, ほかのフラグの立ったパケット, ACL に該当するリストが無い不審なパケットは, SPP によってすべてドロップされる.

3.3.2.プロトコル判別通信

上位プロトコルの多くは TCP を使用しているが，プロトコルによる特定の分類を行うことができる. コネクション確立後の動作が Active（図９，クライアント側から最初のデータパケットを送信）であるか，Passive（図１０，サーバ側から最初のデータパケットを送信）である．さらに一度の通信に使用するコネクション数，コネクションの開設時間の長短などが分類対象になる．例えば http は Active セッションであり，コネクションの数が多く，それぞれのコネクションの開設時間は短い. 逆に ssh は Passive セッションであり，コネクション数は１つ，開設時間は長い．以上のように，コネクションは上位プロトコルの仕様によって対応する必要がある.

http の場合コネクション数が多くなるため，当然サーバへの負荷が高くなる．これを解決するには, SecretFunction のシーケンス番号計算アルゴリズムを簡単にすることがシンプルで最も効果的である．一方，ssh や telnet プロトコルなど長時間に渡って接続するプロトコルの場合には，アルゴリズムのために SecretFunction が推測され，パケットの成りすましによる不正アクセスが行われる可能性がある．このトレードオフを解決するため, SPP ではプロトコル別に SecretFunction を管理者が選択・設定する方式を取っている.

４．他パッケージとの比較

SPP と同様の機能を提供するパッケージとして Linux 系に実装されている SYNCookies[6]と BSD 系に実装されている SYNCache[7]とがある. SPP と２つのパッケージの違いを表１に示す.

4.1.SYNCookies

SYNCookies の基本概念は SPP とほぼ同じであり，SPP とのその他の相違点としては

- Cookie を内部キャッシュに保存
- キャッシュが溢れた場合には古いものから削除
- キャッシュに無いものに関してはシーケンス番号を Cookie として判断
- SecretFunction は1つ
- シーケンス番号の範囲が狭く，推測されやすい
- Passive セッションの通信ができない

のような点が挙げられる．最も決定的な違いは，SYNCookies は 3.3.2 節で述べた Passive セッションの通信をができないという点である．これは SYNCookies が http プロトコルなどをターゲットにしているためであり，導入した場合のサービスが限定されることになる.

4.2.SYNCache

SYNCache は，未開設のコネクションに対して，従来よりも軽いデータ構造 (syn cache) に情報を保持しておき，大量の SYN パケットが届いて syn cache も溢れた際には，未開設のコネクションの古い SYN を捨てることによって DoS の状況を避けるというものである.

SYNCookie に比べ有効であるが, キャッシュが溢れるほどの DoS 攻撃下ではコネクションが確立できなくなる可能性を内包している.

パッケージ / 評価項目	SYNCookies	SYNCache	SPP
Passive セッションへの対応	×	○	○
DoS 攻撃下における コネクションの確立性	○	△	○
偽造 ACK パケットによる DoS 攻撃への耐性	○	△	△
接続要求 (SYN パケット受信, SYN/ACK パケットの返信) 時の対応	・Cookie 作成し シーケンス番号へ格納 ・記録はキャッシュに保存 ・キャッシュがあふれた場合は 古い SYN から削除	・独自の高速キャッシュに 保存 ・キャッシュがあふれた場合は 古い SYN から削除	・Cookie を作成し シーケンス番号へ ・キャッシュなし
Cookie 作成	・1 つの SecretFunction シーケンス番号の範囲が 狭く推測されやすい	× (未使用)	・サービスにあわせた SecretFunction を使用 ・設定選択可能
AU_ACK の判別	・キャッシュを検索 ・キャッシュに無いものは Cookie による認証	・キャッシュの記録のみ	・Cookie による認証のみ
処理層	TCP 層	TCP 層	IP 層

表1　SPP と他パッケージの比較

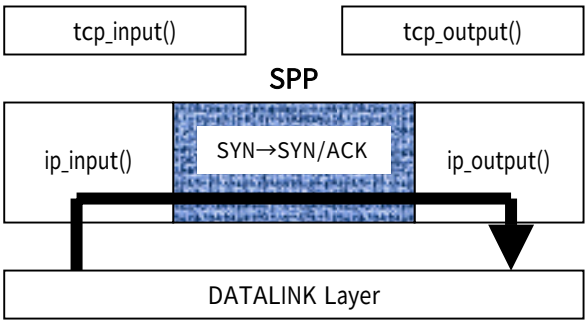

図11　SPP の配置と SYN 到達時の挙動

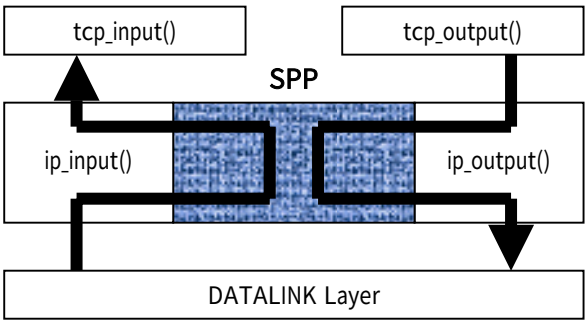

図12　コネクション構築後

5．実装

我々は SPP を FreeBSD カーネルによって開発しており，今回は FreeBSD4.4-Release で実装した．SPP は関数として IP スレッドに存在し，IP スレッドの入力関数である ip_input()と出力関数である ip_output()から呼び出される．SYN 到達時には図１１のように TCP の SYN/ACK パケットを TCP スレッドの処理を行うことなく出力し，コネクション確立後は図１２のように対象の送受信パケットを SPP 経由させ, ACL によって正当性を判定し，送受信を行っている．

なお，SPP では SYN/ACK パケットを返信する際には，受信した SYN パケットに対し

- アドレスとポート番号の入れ替え
- チェックサムとシーケンス番号の計算
- フラグを SYN/ACK へ変更

を施しただけのもの（mbuf）を送信し，CPU とメモリ資源を節約している．

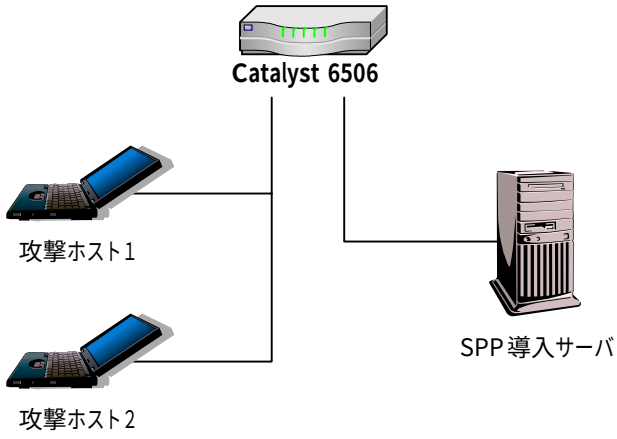

図13　実験環境

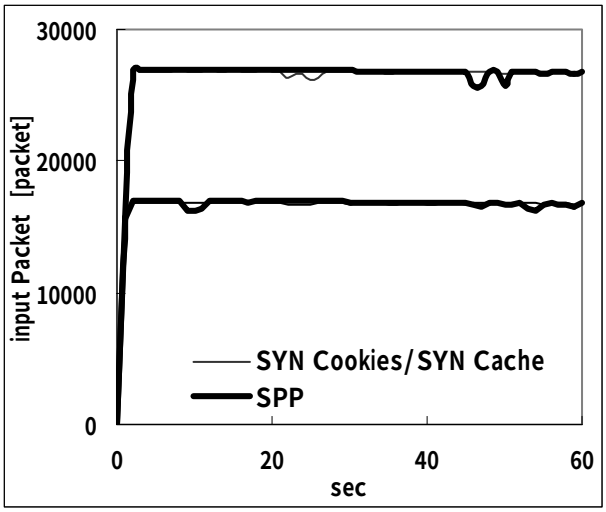

表2 SPP の SYN/ACK 返答回数

6．評価

今回は図１３の環境で実験用ネットワークを構築した．なお，SPP 導入サーバの構成を以下に示す

CPU ： Intel Celeron 1GHz

Memory ： 256M

NIC ： Intel EtherExpress Pro (10/100Base-T)

6.1.SYN/ACK パケットの返答状況

まず，受信した SYN/ACK パケットを遅延無く返答するかについて実験を行った．同じスペックのサーバに SPP を導入したサーバと SYNCookies/SYNCache を導入したサーバ（FreeBSD4.6-Release）を用意し，それぞれに秒間 27000 前後, 16000 前後の SYN パケットを送信し，返答してきた SYN/ACK パケット数を測定した（表２）．

結果として，双方は拮抗しており送信した数とほぼ同数の SYN/ACK パケットの応答を確認した．以上より，SPP 稼動による特別な負荷というものは確認できなかった．

6.2.ポートスキャン

次に一般的なポートスキャナーである nmap[8]を使用して, ポートスキャンへの挙動を調査した．今回使用したポートスキャンのモードは以下の通りある．

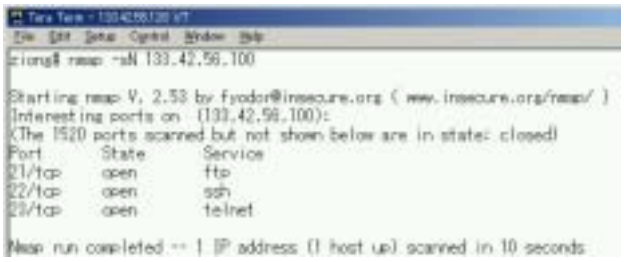

図14 SYN Cookies/SYN Cache へのポートスキャン結果

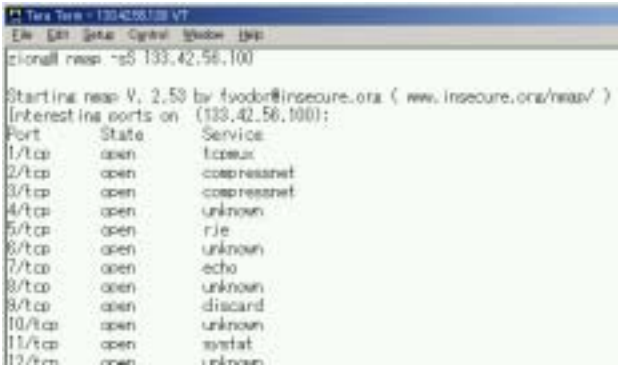

図15 SPP への SYN スキャン結果

図16 SPP への NULL スキャン結果

・Connect Scan

コネクションの通常接続によるポートスキャン．SYSLOG に痕跡が残る．

・SYN Scan

SYN の応答である SYN/ACK に対して ACK パケットではなく RST パケットを送信するポートスキャン方法．SYSLOG に痕跡が現れないステルス型ポートスキャン．

・NULL スキャン

フラグの立っていない不当パケットを送信し，RST パケットが返信されるかどうかでポート開閉を判断するステルス型ポートスキャン．

・FIN スキャン

NULL スキャン同様，不当な FIN パケットを送信し，RST パケットが返信されるかどうかでポート開閉を判断するステルス型ポートスキャン．

最初に，SYNCookies/SYNCache のサーバに各種ポートスキャンを行ったところ，どのモードでも図１４のように開閉ポート特定が可能といったセキュリティ上好ましくない結果になった．

同様に，SPP に対して各種ポートスキャンを行った．まず，Connect スキャンと SYN スキャンについては図１５のようにすべてのポートが開いているという結果が得られた．実際には開いてはいないのだが，SYN パケットには必ず SYN/ACK パケットを返信するという SPP の仕様のために，このような結果になった．しかし，現実問題として，すべてのポートが開いているという状況は存在しないため，十分な警告と不信感を不正アクセスユーザに与えることが可能であると考える．

次に NULL スキャンと FIN スキャンについては図１６のように，すべてのポートが閉じられているというまったく逆の結果が得られた．これも SYN パケットと正しいシーケンス番号を持った ACK パケット以外のパケットはすべてドロップするという SPP の仕様上の結果である．

SPP のポートスキャン実験の結果を総合すると，本当に開いているポートと特定は不可能であり，セキュリティ上好ましい結果と言える．

ポートの開閉を確認するだけでなく，SPP では OS Finger Print でサーバの OS を特定できない．これはベンダや OS バージョンが特定するとセキュリティホールが判明し対応が困難なネットワーク中継機器にとって有効である．

７．考察

SPP 最大の特徴は SYN Cookies と SYN Cache が TCP 層で処理を行うのに対し，IP 層で処理を行うという点である．IP 層で処理をするため，将来的にはネットワークベースに対応させネットワーク機器への導入といった拡張も望める．

さらに，ポート番号で設定する各種上位プロトコルに対応したゲートウェイの挙動をシミュレートすれば，組織内ネットワーク対外接続部分への適用も可能である．

SPP は仕様上，stick などの IDS を圧迫する攻撃を防ぐことができるが（６章では割愛したが，実際に stick でも防御実験を行っている），snort[9]などデータリンク層の機構である bpf（BerkeleyPacketFillter）を使用した IDS においては，その効果も薄くなる．我々の開発した Dionaea[10]も bpf ベースであるため，この点では弱点となるが，今後 IP ベースへと移行させ，SPP とあわせてひとつのパッケージとする予定である．このように，DoS，SYNFlood，ポートスキャン対策には SPP を，Connect スキャンや Connect 関数によるソケット消費型 DoS 攻撃に関しては Dionaea を用いることで総合的な対 DoS 機構とすることが可能であると考える．

８．最後に

本論文では SPP の DoS（SYNFlood, portscan, stick）への耐性を証明でき，拡張性についても述べることができた．次は攻撃ホストを増やし DDoS への耐性の検証を行う予定であるが，構造が TCP スレッドや SYNCookies/SYNCache よりもシンプルであるため DoS 攻撃へのスケーラビリティはあると考えられる．さらに，今後は Dionaea との融合について着手する予定である．

参考文献

[1] http://www.caida.org
/outreach/papers/backscatter/c

[2] http://bvlive01.iss.net/issEn/delivery/
xforce/alertdetail.jsp?id=advise74

[3] http://www.cisco.com/japanese/warp/
public/3/jp/product/doc/catalog/
soft/psecure/index.1.html

[4] http://www.ietf.org
/html.charters/itrace-charter.html

[5] Philip Miller , Mastering TCP/IP ,
Ohmsha(1998)

[6] http://cr.yp.to/syncookies.html

[7] http://people.freebsd.org/
~jlemon/papers/syncache.pdf

[8]www.insecure.org/nmap/

[9] http://www.snort.org/

[10] 不正アクセス検知機能を持つ
ホスト型防御システム“ Dionaea ”の構築：
伊藤大輔：和歌山大学学位論文 2001