

IP アドレスを用いたセッション管理による DTLS サーバへの DoS 攻撃の軽減

須賀 灯希¹ 落合 秀也¹ 江崎 浩¹ 岡田 和也²

概要: 昨今、リアルタイム性が高く処理負荷の小さいデータグラム通信が広く用いられている。一方で悪意のある攻撃者から通信を守ることが強く求められており、様々なセキュリティプロトコルが利用されている。データグラム通信におけるセキュリティプロトコルには Datagram Transport Layer Security (DTLS) プロトコルがある。しかしながら DTLS のハンドシェイクプロトコルには仕様を悪用したサーバへの DoS 攻撃が可能であるという問題がある。この問題に対処する研究は第三者による認証を利用した研究が主であり、サーバ側の処理を変更する方向性での研究は少ない。本研究では DTLS におけるサーバ側の処理を変更し、IP アドレスの表をサーバに持たせ、各クライアントとのハンドシェイクの状態の一部分を管理させることで、サーバへの DoS 攻撃の影響を軽減する手法を提案した。提案手法により新たな機器を用意せずに DoS 攻撃の被害を軽減可能になる。評価では、提案手法を実装し、DoS 攻撃の被害を軽減できていることを確認した。さらに提案手法を組み込んだ際にハンドシェイク通信に新たに生じる処理時間について、表のデータ構造やエントリ数による違いを、実験を通して比較した。

Mitigation of DoS Attack against DTLS Server by Session Control with IP Address

TOKI SUGA¹ HIDEYA OCHIAI¹ HIROSHI ESAKI¹ KAZUYA OKADA²

1. はじめに

昨今、映像や音声のストリーミング、IP 電話などのリアルタイム通信の需要が高まっている。インターネット通信を支えるプロトコルには様々なものがあり、トランスポート層で動作する UDP プロトコルは同階層の TCP プロトコルよりもリアルタイム性の高いコネクションレス型の通信を提供する。さらに電子機器技術、センサーネットワークの発達により計算処理能力の低いセンサー等の機器もインターネットに接続されるようになり、これらの機器との通信には TCP よりも処理負荷の小さい UDP が利用される。

一方で大切なデータを悪意のある攻撃者に奪われないために通信を安全に行うことは常に求められており、様々なセキュリティプロトコルが利用されている。現在最も広く使われているセキュリティプロトコルの一つに TCP の上で動作する Transport Layer Security (TLS) プロトコル

[1] がある。しかし TLS は UDP のようなリアルタイム性が求められるデータグラム通信で再送機能が必要ない場合や、各ノードの計算処理能力が低いセンサーネットワークの用途には向かない。このような要求を満たすために、UDP の上で動作する Datagram Transport Layer Security (DTLS) プロトコル [2] が 2006 年に策定された。

しかしながら、DTLS では悪意のある攻撃者がサーバに対し大量のセッション確立要求を送ることでサーバに大きな負荷を与える DoS 攻撃が問題となっている。この問題に対する研究は複数あるが、サーバ側の処理を変更する方向性での研究は少ない。そこで本研究ではサーバ側のプロトコル処理を変更し、IP アドレスの表をサーバに持たせて各クライアントとのハンドシェイクの状態の一部分を管理させることにより、サーバへの DoS 攻撃の影響を軽減する手法を提案する。提案手法では、クライアント/サーバ以外の新たな機器を用意せずに DoS 攻撃の被害を軽減できる。

¹ 東京大学大学院 情報理工学系研究科

² 東京大学 情報基盤センター 情報メディア教育研究部門

2. DTLS の概要と DoS 攻撃

本節では、本研究で研究対象とする DTLS について概説し、DTLS のハンドシェイクプロトコルを利用したサーバへの DoS 攻撃について述べる。

2.1 Datagram Transport Layer Security (DTLS)

本研究で対象とする DTLS について概説する。DTLS は TLS に基づく、認証と暗号化によりサーバとクライアント間のデータグラム通信の安全を担保するセキュリティプロトコルで、2006 年に Internet Engineering Task Force (IETF) によってバージョン 1.0 が RFC4347 として策定された。2017 年現在では 2012 年に策定されたバージョン 1.2 (RFC6347) が最新である。

DTLS は、内部でハンドシェイクプロトコルとレコードプロトコルに分かれている。ハンドシェイクプロトコルによってサーバとクライアント間で図 1 に示す順序で規定されたメッセージを交換しセッションと呼ばれるコネクションを確立する。セッション確立までの間に暗号化アルゴリズムの決定・暗号鍵の交換を行う。暗号化されたメッセージはレコードと呼ばれる単位に分割されて通信相手に送られ、レコードのフォーマットを定めたのがレコードプロトコルである。またハンドシェイクプロトコルにおいては各メッセージ群はフライトと呼ばれる単位にまとめられる。

文献 [2] にて指摘されているように、データグラムを扱うセキュリティプロトコルは一部の Denial of Service (DoS) 攻撃を受けやすい。DTLS ではこうした攻撃への対策としてハンドシェイクプロトコルの始めにステートレスなクッキーの交換を行う。クライアントがフライト 1 をサーバに送るとサーバはクッキーを生成しそれを加えたフライト 2 をクライアントに返す。クライアントは受け取ったクッキーを加えてフライト 3 を送り、サーバは受け取ったクッキーを検証して自らが送ったものと一致したときのみクライアントにフライト 4 を送り、ハンドシェイクを次に進める。これによりサーバとのハンドシェイクを完遂しようとするクライアントはサーバからのクッキーを一度受け取らなければならない、送信元 IP アドレスを詐称した攻撃者のサーバとのハンドシェイクが困難になる。

2.2 DTLS における DoS 攻撃

2.1 節で述べたクッキーの交換は DoS 攻撃への抜本的な対策にはなっておらず、サーバに接続できる有効な IP アドレスを持つ攻撃者はサーバへの攻撃が可能である。具体的には、図 1 においてサーバからフライト 4 を受け取った後に続くフライト 5 を返さず、サーバにタイムアウトまでの間攻撃者との接続を保持させることで、一定時間リソースを消費させるといった攻撃が可能である。この攻撃によ

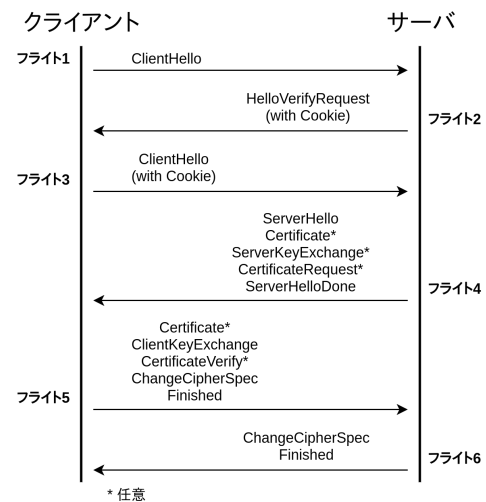


図 1 DTLS のハンドシェイクプロトコル

てサーバに生じる状況はハーフオープンセッションと呼ばれる。ハーフオープンセッションを短時間に大量に生成させるための攻撃は、単一の攻撃元から行われる場合とボットネットを利用した大量のボットなどの複数の攻撃元から行われる場合がある。

ハーフオープンセッションが短時間に大量に生成されるとサーバやネットワークに大きな負荷がかかり、正規のクライアントとの通信に大きな遅延が生じたり、最悪の場合正規の通信そのものができなくなる。本研究では実際に攻撃を受けたサーバに起こる問題として次の 2 種類を取り上げ、これらを解決・軽減するための手法を提案する。

問題 1 サーバ資源の浪費

サーバは各クライアントとの接続一つ一つに対し一定量のメモリを割り当てるため、ハーフオープンセッションの数が増加するとそれに比例してサーバの使用メモリ量は増加する。サーバの性能によっては、攻撃を受けるとメモリの大部分をハーフオープンセッションに費やしてしまい正規のクライアントとの通信など他の機能が損なわれる場合がある。具体的なメモリ使用量の評価は 5 節にて示す。

問題 2 正規クライアントのセッション確立の困難化

サーバの OS とその設定に起因し、サーバが同時に生成できるクライアントとのセッション数には上限が存在する場合がある。例えば Linux では 1 つのプロセスが作成できるファイルディスクリプタの数には上限が設けられており、サーバとクライアント間でファイルディスクリプタの一つであるソケットインターフェースを用いた通信を行う場合はサーバが同時に生成できるセッション数はファイルディスクリプタの上限数に制限される。上限に到達するとクライアントとの新しい接続は作成できなくなるため、この状況を意図的に生じさせて正規のクライアントがサーバに接続できないようにする DoS 攻撃が可能であるという問題があ

る。生成可能なセッションの上限数は変更可能であるが、上限数を増加することは攻撃を受けた際のサーバのメモリ使用量の最大値を増加させることにつながり、サーバの性能によって増加できる上限数に限度がある。その上限数を上回る数の接続要求を送りつける攻撃は常に想定されるため、生成セッション数の上限を変えることは根本的な問題の解決にはならない。

3. DoS 攻撃対策と関連研究

本節では、一般的な DoS 攻撃への対策を比較検討し、TLS・DTLS における既存研究をまとめる。

3.1 一般的な DoS 攻撃とその対策の比較検討

DoS 攻撃はインターネット上のサーバ、ルータなどの機器、もしくはネットワーク全体を標的にしてそのサービスや機能を妨害したり停止させる攻撃のことで、様々な攻撃手法がある [3]。よく知られた攻撃としては、TCP ハンドシェイクの仕様を突いた攻撃である TCP SYN-flood 攻撃 [4] 等がある。本研究で取り上げる DTLS サーバへの DoS 攻撃は TCP SYN-flood 攻撃に近い攻撃として分類される。

次に TCP SYN-flood 攻撃への既存の対策を整理・分類し、DTLS への適用について議論する。TCP SYN-flood 攻撃についてはこれまでに様々な対策が提案されており、本研究ではこれらをその方向性から次の 3 種類に分類する。

(1) アクセス制限

ルータやファイアウォールなどの外部機器によって攻撃と思われるパケットを遮断する方式である。適切な設定がなされれば攻撃を防ぐことができるが、外部機器を新たに設置するコストが発生する問題や、設定の不備により正規のクライアントからの通信も遮断されてしまう可能性を排除できない。また、時々刻々と変化する攻撃を防ぐにはアクセス制限対象となる IP アドレス群を常に更新し続ける必要がある。

(2) サーバの負荷軽減

外部機器やサーバの処理の工夫によって攻撃の影響を軽減する方式である。クライアントからの SYN パケットを受信した際にサーバが消費する記憶領域をより小さくすることで同時接続可能な TCP コネクションの数を増やす SYN cache [5]、サーバがパケットのシーケンス番号をクライアントから送られてきた SYN パケットから算出することで記憶領域を割り当てないようにする SYN cookie [6]、TCP 層ではなく IP 層でパケットのフィルタリングを行う SYN Packet Pacifier [7]、ルータがサーバの代わりに SYN-ACK パケットを送信し適切な ACK パケットを受け取った場合のみサーバに SYN パケットを送る方式 [8] 等がある。またハンドシェイクのタイムアウト時間を短くする設定の変更もこれに該当する。この方式では攻撃そのものを

防ぐことができないという問題がある。

(3) クライアントの負担増加

クライアントに新たな処理を付加させることにより攻撃のコストを増大させ、抑止力とする方式である。サーバが接続を要求してきたクライアントに暗号問題を解かせ正しく解けたクライアントからの接続要求のみを受け付けるようにする Client Puzzles [9] 等があるが、クライアントに新たな処理を付加させた結果正規のクライアントの通信に無視できない遅延を引き起こす可能性がある。またこの方向性での問題解決は DTLS が計算能力の低い機器でも扱えるように軽量性を指向したプロトコルであるという事実とそぐわない。

上記の対策の内、サーバの負荷軽減を目的とした SYN cache はハッシュテーブルによって TCP コネクションを管理し、ハッシュ関数の引数として宛先と送信元それぞれの IP アドレスとポート番号を用いる。したがって SYN cache と同じ機構を本研究で扱う DTLS の問題への対策として適用した場合、単一の IP アドレスを持つ攻撃者が送信元ポート番号を変えて攻撃を行った際に各セッションに対するハッシュ値は異なるものになり、ハッシュテーブルのエントリ数を圧迫してしまう。SYN cookie は TCP パケット固有の特徴を利用した対策であり、本問題に適用できない。ハンドシェイクのタイムアウト時間を短くする設定の変更は適用できるが、正規の通信に遅延が生じた場合に通信が切断される可能性が高くなるという問題が生じる。本研究ではこれらの問題を回避しつつ DoS 攻撃に対して被害サーバの負荷を軽減可能な対策を提案する。

3.2 TLS・DTLS における既存研究

本研究で問題とする、ハーフオープンセッションを大量に生成させる DoS 攻撃は TLS においても同様のことが可能であり、問題となっている。対策としては、Client Puzzles を TLS に拡張したものや [10]、プロキシサーバを用いた方法 [11] などがある。

DTLS のハンドシェイクを利用した DoS 攻撃については、問題提起も含めてこれまで様々な研究がなされている。本研究ではハーフオープンセッションを大量に生成させる DoS 攻撃を行うためにはサーバに接続のできる有効な IP アドレスが必要であると述べたが、送信元 IP アドレスの詐称と通信の傍受のできる攻撃者を仮定すればサーバへの DoS 攻撃が可能であると述べる論文がある [12]。この論文では問題の解決策としてクライアントでもサーバでもない第三者 (Trust Anchor:TA) を用意する機構を提案している。この方式では TA からの認証を受けられないクライアントはサーバとハンドシェイクを開始することができないため、サーバへの DoS 攻撃を防ぐことができる。しかし、新たに TA を設置する必要があり運用上のデメリットが生

じる。さらには TA が DoS 攻撃の標的になりうるという問題や、クライアントと TA の通信の安全性に問題がある。また別の先行研究としてクライアントとサーバの間においてゲートウェイ (Trust Party) を用いて ClientHello メッセージの認証を行うというものがあるが [13]、こちらも第三者を用いた認証という点で上述の問題は回避できない。

4. 提案手法

本節では、ハーフオープンセッション数を制限しサーバへの DoS 攻撃の影響を軽減する手法を提案し、提案手法の DoS 攻撃への対処やデータ構造について説明する。

4.1 提案手法の概要

2.2 節で述べた DoS 攻撃は、単一もしくは複数の IP アドレスがサーバに対しハーフオープンセッションを大量に生成させることによって起こるものである。したがってサーバに接続を試みる各ノードについてハーフオープンセッションの状態にある IP アドレスを管理し、サーバハーフオープンセッション数を制限することで被害を軽減できる。本研究ではサーバ側のプログラムに新たに IP アドレス表を付加することでこれを実現する。

IP アドレス表の各エントリにはクライアントの IP アドレスと接続情報が格納され、1 つのエントリが 1 つの接続に対応する。図 2 に提案手法を用いたハンドシェイク通信の概要を示す。サーバの処理に変更が加わるのは、クライアントからフライト 3 を受け取りフライト 6 を送るまでの間だけである。主な変更点は次のようになる。

- フライト 3 を受信後、送信元 IP アドレスが IP アドレス表に存在するか探索
- 探索の結果同じ送信元 IP アドレスを持つエントリが表内に格納されている場合、対応する接続を切断し当該エントリを削除
- 現在のエントリ数を確認し上限値に達している場合は、最も古くに格納されたエントリに対応する接続を切断し当該エントリを削除
- フライト 3 からクライアントの IP アドレスと接続情報を取り出し、エントリを作成して IP アドレス表に格納
- フライト 5 を受信後 IP アドレス表を探索
- 探索の結果同じ IP アドレスを持つエントリがある場合、当該エントリを削除

処理の変更によってクライアントに送る各フライトの内容は変化しないため、クライアント側は一切処理を変更せずにサーバとのハンドシェイクを完遂できる。以上より IP アドレス表は次の要件を満たす構造となる。

- 現在のエントリ数が常に確認可能
- ある IP アドレスが与えられた際にその IP アドレスを

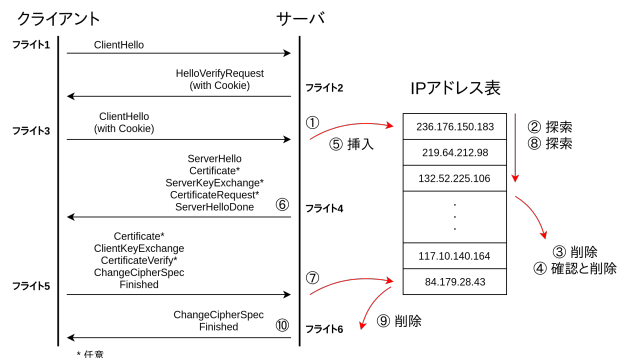


図 2 提案手法を用いたハンドシェイク通信

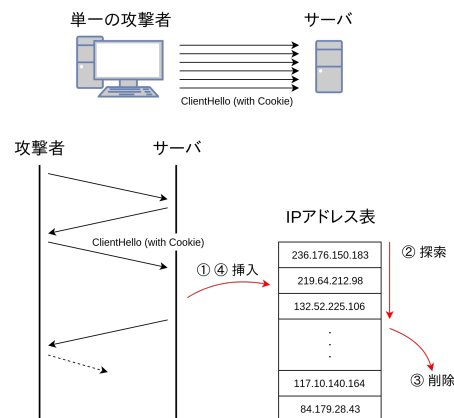


図 3 提案手法の単一の攻撃元からの攻撃における動作

持つエントリを有限時間で探索可能

- 最も古くに格納されたエントリを参照・削除可能

4.2 DoS 攻撃への対処

DoS 攻撃を受けた際の提案手法の動作について解説する。

単一の攻撃元からの攻撃における動作は次の手順のようになる。概要を図 3 に示す。手順の番号と図 3 の丸囲み番号が対応している。

- (1) 1 回目のフライト 3 については、フライト 3 を受け取った後 IP アドレス表を探索し、同じ IP アドレスを持つエントリは存在しないため攻撃者の IP アドレスを持つエントリが格納される。
- (2) 2 回目以降については、フライト 3 を受け取った後 IP アドレス表を探索し、同じ IP アドレスを持つエントリが存在することがわかる。
- (3) 当該エントリを削除し対応する接続を切断する。
- (4) フライト 3 から新たにエントリを作成し IP アドレス表に格納する。

これにより IP アドレス表には各クライアントのエントリは 1 つしか存在できず、接続も 1 つしか保持しないため、同一 IP アドレスからの攻撃を抑制できる。

複数の攻撃元からの攻撃に対しての動作は次の手順のようになる。概要を図 4 に示す。手順の番号と図 4 の丸囲み

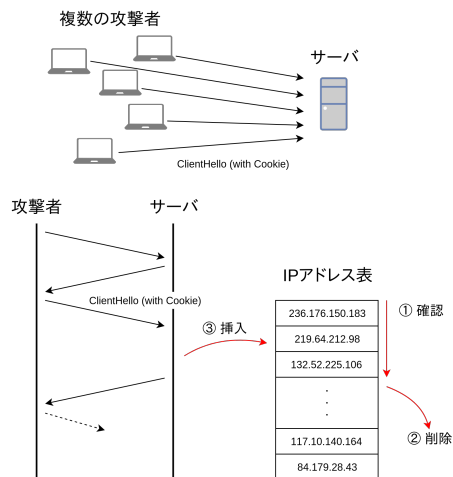


図 4 提案手法の複数の攻撃元からの攻撃における動作

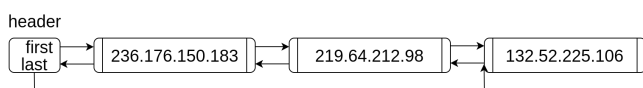


図 5 本研究で用いる線形リスト

番号が対応している。

- (1) フライト 3 を受信後 IP アドレス表のエントリ数を確認する。
- (2) エントリ数が上限値に達していた場合、IP アドレス表の中で最も古いエントリを削除しその接続を切断する。
- (3) フライト 3 から新たにエントリを作成し、格納する。

これにより攻撃を受けた場合でも接続数がエントリ数の上限値を超えることはなく、接続の総数は上限で一定に保たれる。そのためサーバのメモリ使用量を抑えられる。またエントリ数の上限値がサーバの生成セッション数の上限値以下であれば、攻撃を受けている間であっても正規のクライアントからの通信による接続は作成され、フライト 3 を送ってエントリが格納されてからフライト 5 がサーバに届くまでにエントリが削除されない限りは、ハンドシェイクを完了しサーバとの接続を確立できる。

4.3 IP アドレス表のデータ構造

IP アドレス表の探索に要する時間はエントリ数に依存し、IP アドレス表のデータ構造の種類によっても変化する。本研究ではこの探索に要する時間を評価するため、IP アドレス表のデータ構造として次の 2 種類を利用する。

(1) 線形リスト

図 5 のような双方向リストを用いる。図 5 では 132.52.225.106、219.64.212.98、236.176.150.183 の順にデータが格納されており header において first から順にノードをたどることで全ノードを探索できる。また last からノードを参照することで最も古いノードを参照できる。

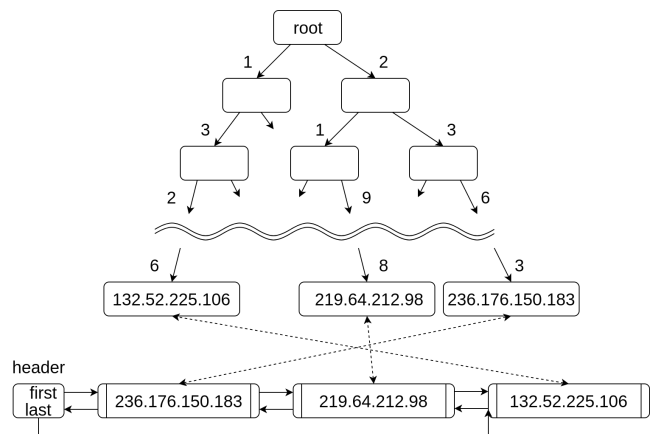


図 6 本研究で用いる基数木

(2) 基数木

基数木とは木構造の 1 種であり、あるノードが配下の子ノード全てに共通する接頭部を持つ構造であるトライ木において、子ノードを 1 つしか持たないノードをまとめたデータ構造である。基数木において最も古くに格納されたノードを参照するため、本研究では図 6 のような基数木と双方向リストを組み合わせた構造を用いる。最も古くに格納されたノードを参照する際には双方向リストにおいて last からノードを参照する。

5. 評価

本節では提案手法の評価について述べる。評価のために提案手法の実装を行った。実装は DTLS のバージョン 1.2 を実装した C 言語のライブラリである OpenSSL [14] を拡張する形を取った。拡張した OpenSSL のバージョンは 2017 年 1 月 10 日時点で最もサポート期間の長いバージョン 1.0.2j である。また IP アドレス表のデータ構造の実装については、線形リストには TAILQ [15]、基数木には libart [16] と TAILQ を用いた。

5.1 評価項目

本研究では 2.2 節で取り上げた 2 種類の問題に関連して次の 2 つの評価項目を定めた。

(1) サーバのメモリ使用量

DoS 攻撃を受けた際にサーバが消費するメモリ量である。計測は Linux の top コマンドでサーバのプロセスを指定し 1 秒ごとに RES のフィールドを参照した。

(2) ハンドシェイク時間

フライト 1 からフライト 6 までハンドシェイクが完全に行われ、その後 100byte のデータのやり取りの完了を到達と定義する。通信の開始から到達の完了までの時間を計測する。

5.2 単一の攻撃元からの攻撃によるメモリ量の変化

本評価ではハーフオープンセッション生成攻撃時に提

表 1 実験に使用した PC

PC1	CPU	Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz × 4
	Memory	16GB DDR3 1600MHz
	OS	Ubuntu 16.04 LTS 64bit with Linux kernel 4.4.0-59-generic
PC2	CPU	Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz × 8
	Memory	16GB DDR3 1600MHz
	OS	Ubuntu 14.04 LTS 64bit with Linux kernel 4.4.0-59-generic

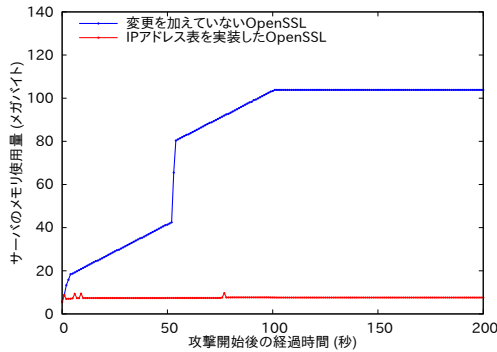


図 7 単一の攻撃元からの攻撃を受けたときのサーバのメモリ使用量の比較

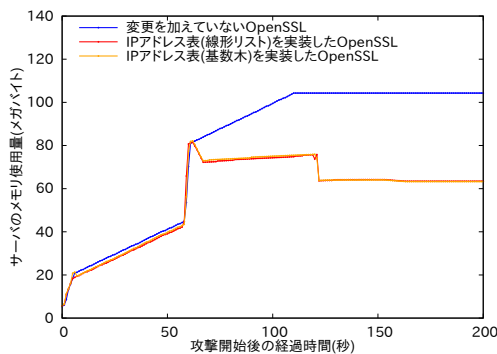


図 8 複数の攻撃元からの攻撃を受けたときのサーバのメモリ使用量の比較

案手法の有無によりサーバのメモリ使用量の違いをみる。サーバとクライアントには、表 1 に示す PC1, PC2 を用いた。攻撃は、0.1 秒ごとにスレッドを生成し、各スレッドからハーフオープンセッション生成攻撃をサーバに対して行った。OpenSSL に変更を加えていない場合と提案手法（線形リスト）でそれぞれメモリ使用量を計測した。図 7 は計測の結果である。横軸は攻撃用プログラムを作動させてからの経過時間、縦軸はサーバのメモリ使用量である。結果より、変更を加えていない OpenSSL に単一の攻撃元から攻撃を行った場合サーバのメモリ使用量が増加する。一方提案手法を実装した場合攻撃を受けたときのサーバのメモリ使用量の増加を軽減できている。

5.3 複数の攻撃元からの攻撃に対するメモリ量の変化

本評価では複数の攻撃元からの攻撃を再現するため、IPv6 Address Generator [17] から取得したランダムな IP アドレスをそれぞれのスレッドに持たせて送信元 IP アド

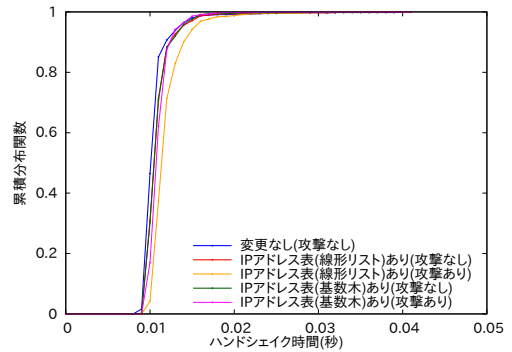


図 9 提案手法の有無によるハンドシェイク時間の比較

レスとする。これにより複数の攻撃元からの攻撃を擬似的に再現する。また IP アドレスを詐称したパケットを受け取れるように、全体の処理時間が変化しない範囲でサーバ側のプログラムを一部変更した。評価では、OpenSSL に変更を加えていないもの、提案手法ありの OpenSSL（線形リスト・基数木）を利用した。IP アドレス表の上限値は 500 エントリとした。結果を図 8 に示す。結果より、提案手法を実装した場合 IP アドレス表のデータ構造が線形リストと基数木のどちらの場合でも攻撃を受けたときのサーバのメモリ使用量の増加を軽減できている。また線形リストと基数木でメモリ使用量の推移に大きな違いが見られないことから、サーバのメモリ使用量においては、提案手法はデータ構造の違いによらず動作することがわかる。

5.4 提案手法の有無による通信時間の変化

本評価では提案手法・攻撃の有無によるハンドシェイク時間の違いをみる。評価には OpenSSL に変更を加えていないもの（攻撃なし）、提案手法ありの OpenSSL（攻撃あり・なし、線形リスト・基数木）を利用した。IP アドレス表の上限値は 500 エントリとした。ここで攻撃なしの状態を攻撃者からの通信は一切なく IP アドレス表のエントリ数が 0 である状態とする。また攻撃ありの状態を複数の攻撃者から攻撃を受けていて IP アドレス表のエントリ数が上限値に達している状態とする。各状態においてハンドシェイク通信を 1,000 回行いそれぞれ通信の到達に要する時間を計測した。結果を図 9 に示す。横軸はそれぞれのハンドシェイクに要した時間、縦軸はある時間以内に通信が到達したハンドシェイクの累積分布関数である。結果より提案手法を実装するとハンドシェイクの通信時間は増加す

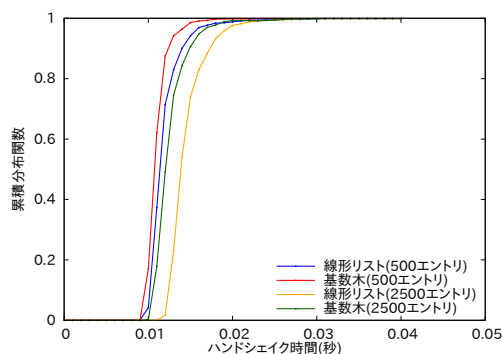


図 10 IP アドレス表のエントリ数によるハンドシェイク時間の比較

る。また表のデータ構造の違いについては、攻撃なしでは線形リストと基数木で通信時間に大きな差は見られないが、攻撃ありでは基数木の方が通信時間の増加が少なくなる。これはエントリ数の増加に伴う探索時間の増分が線形リストよりも基数木の方が小さいという事実と合致する。

5.5 表のエントリ数による通信時間の変化

本評価ではエントリ数による表の探索時間の違いをみる。評価には提案手法ありの OpenSSL（攻撃あり、線形リスト・基数木、表の上限値が 500・2,500）を利用した。複数の攻撃者からの攻撃により表のエントリが上限に達している状態でハンドシェイク通信を 1,000 回行い、それぞれ通信の到達に要する時間を計測した。計測したハンドシェイク時間の累積分布関数 (CDF) を図 10 に示す。結果より同じエントリ数では線形リストと基数木では基数木の方が IP アドレス表の探索時間が短くなることがわかる。またデータ構造が基数木である場合においてエントリ数が 500 から 2,500 に増加すると通信時間も増加しているが、これは基数木の探索時間はエントリ数によらず IP アドレスの文字列長でのみ決まるという事実と反する。

6. おわりに

本研究では、短時間にハーフオープンセッションを大量に生成させて DTLS のハンドシェイク通信を行うサーバに被害を与える DoS 攻撃の問題を提起し、その軽減手法としてサーバに IP アドレス表を持たせてハンドシェイクの状態の一部を管理させる手法を提案した。また実際に DoS 攻撃をサーバに対し行う実験を通して提案手法が確かに DoS 攻撃の被害を軽減できていることを確認した。さらに、提案手法によりハンドシェイク通信に新たに生じる処理時間について、表のデータ構造やエントリ数による違いを比較した。比較の結果エントリ数が少ないほど表の探索時間が短くなり、表のデータ構造として同エントリ数では線形リストよりも基数木の方が探索時間が短いことを確認した。

本研究の実験は、提案手法の妥当性を検証するために DTLS のハンドシェイク通信の部分だけを取り出し、外部

要因を排除するために 2 台のコンピュータ間、または 1 台のコンピュータ内のみで行った。DTLS を利用したアプリケーションに本研究の手法を組み込む際には、アプリケーション層など他の層を考慮する必要がある、したがって DTLS を利用したアプリケーションプログラムの中で提案手法の動作確認を行うことを今後の課題とする。特に複数の攻撃者からの攻撃は擬似的に再現したものであるため、実際にサーバと通信のできる有効な IP アドレスを持つ攻撃者が複数存在する攻撃環境からの攻撃に対する提案手法の動作を検証する必要がある。

参考文献

- [1] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2." RFC 5246 (Standards Track), August. 2008.
- [2] E. Rescorla, N. Modadugu, "Datagram Transport Layer Security Version 1.2." RFC 6347 (Standards Track), January. 2012.
- [3] M. Handley, Ed., E. Rescorla, Ed., "Internet Denial-of-Service Considerations." RFC 4732 (Informational), November. 2006.
- [4] CERT Advisory CA-1996-21, "TCP SYN Flooding and IP Spoofing Attacks." 1996.
- [5] J. Lemon, "Resisting SYN flood DoS attacks with a SYN Cache," Proceedings of the BSD Conference 2002 on BSD Conference, BSDC' 02, pp. 89-98, USENIX Association, 2002.
- [6] D. J. Bernstein, "SYN cookies." 1996. <http://cr.yp.to/syncookies.html>.
- [7] D. Ito, Y. Izumi, S. Saito et al., "TCP セッション管理による DoS 耐性の考察," 電子情報通信学会技術研究報告. IA, インターネットアーキテクチャ, 2001.
- [8] T. Darmohray, R. Oliver, "Hot Spares" For DoS Attacks," USENIX SAGE, 2000.
- [9] A. Juels, J. Brainard, "Client puzzles: a cryptographic defense against connection depletion attacks," Proceedings of NDSS Symposium, pp. 151-165, 1999.
- [10] D. Dean, "Using client puzzles to protect TLS," Proceedings of the 10th conference on USENIX Security Symposium, Vol. 10, 2001.
- [11] D. Holmes, "Mitigating DDoS Attacks with F5 Technology." 2013. <https://f5.com/resources/white-papers/mitigating-ddos-attacks-with-f5-technology>.
- [12] M. Tiloca, C. Gehrmann, and L. Seitz, "On improving resistance to Denial of Service and key provisioning scalability of the DTLS handshake," International Journal of Information Security, Springer, 2016.
- [13] M. Yassine, A. Ezzati, and M. Belaisaoui, "DoS Attacks Analysis and Improvement in DTLS Protocol for Internet of Things," ACM International Conference on Big Data and Advanced Wireless Technologies, 2016.
- [14] "OpenSSL." <https://www.openssl.org/>.
- [15] "Man page of QUQUE." https://linuxjm.osdn.jp/html/LDP_man-pages/man3/queue.3.html.
- [16] "libart." <https://github.com/armon/libart>.
- [17] "IPv6 Address Generator." <https://www.randomlists.com/ip-addresses>.