APIフックによるSSL/TLS通信のDPIシステムの提案

三浦 良介 †　高野 祐輝 †　井上 朋哉 §
myu2@nict.go.jp　ytakano@wide.ad.jp　t-inoue@jaist.ac.jp

†国立研究開発法人，情報通信研究機構，サイバー攻撃検証研究室
923-1211　石川県能美市旭台 2-12
§国立大学法人，北陸先端科学技術大学院大学，高信頼ネットワークイノベーションセンター
923-1292　石川県能美市旭台 1-1

あらまし 企業内ネットワークのセキュリティ向上のため，従来のパケット監視だけでなく，SSL/TLS 通信の監視を行いたい要望がある．これを実現するための手法として SSL/TLS 通信をプロキシ経由で行い監視する方法があるが，これは，1 つのリクエストに対し SSL/TLS 通信処理が 2 回必要であり，計算の処理性能が必要になる．これを解決するため，本稿では，ソフトウェアで各端末毎に SSL/TLS 通信の監視を行い，その結果を集約するシステムを提案する．具体的には，企業内ネットワークに属する端末で SSL/TLS 通信の関数を API フックし，その結果を集約する．これによりソフトウェアで効率よく実現可能な SSL/TLS 通信の監視システムが可能になる．

A DPI System for the SSL/TLS connection using API hook

Ryosuke Miura †　Yuuki Takano †　Tomoya Inoue §
myu2@nict.go.jp　ytakano@wide.ad.jp　t-inoue@jaist.ac.jp

† National Institute of Information and Communications Technology 2-12,
Cybersecurity Research Center, Cyber Range Laboratory. Asahidai, Nomi, Ishikawa, 923-1211, Japan
§ Japan Advanced Institute of Science and Technology, Dependable Network Innovation Center
1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan

Abstract To improve security of enterprise networks, it is required that monitoring not only network packets but also communication data of SSL/TLS. SSL/TLS proxies are used to inspect such encrypted data, but the proxies need twice amount of calculation because data is encrypted by proxies and end-hosts. To solve this problem, we propose a system that can inspect communication data of SSL/TLS on each end-host and aggregate observed results. We adopted API hooking to inspect SSL/TLS connection on end-hosts in enterprise networks, and collects and aggregate the inspected results. Our system is fully software-based, and thus it can monitor inside SSL/TLS connections efficiently.

1　はじめに

業務ネットワーク (エンタープライズネットワーク) は，企業の複数拠点間を結び構築される．また，その形態の一つとしてクラウド環境と結ばれている場合もある．自宅や外出先からエンタープライズネットワークへ接続するためには VPN が経由される．エンタープライズネットワークは規模の増加により要求が増えるため複雑になる傾向がある．その結果，多様な接続方法など利便性の向上を求めた場合，情報セキュリティの維持は難しくなる．

企業が求められる事は，エンタープライズネットワークで情報セキュリティを維持する事である．多くのエンタープライズネットワークは，情報セキュリティを維持するため，アプライアンス製品のミドルボックスを導入し，アプリケーションレイヤーのパケット検査 (ディープ・パケット・インスペクション-DPI) を行っている．IDS/IPS，ファイアウォールなどは，DPI した結果を参照し，攻撃を示唆するパケットを検知した時，アラートを生じさせる事などが可能になる．

企業の従業員が企業外の第三者と知的財産を共有する場合，相手を識別し判断した上で機密情報を安全にやりとりする事が求められる．そのため，業務端末で機密情報のやりとりを行う時，真正性，完全性を有した暗号通信の SSL/TLS 通信 [8] が広く利用されている．しかしながら，この SSL/TLS 通信の DPI は，パケット取得をベースにしたシステムでは通信の中身が秘匿されているため行う事が出来ない．そのため，業務端末が SSL/TLS 通信を DPI する方法は，SSL/TLS 通信のプロキシ (SSL/TLS プロキシ) を利用する事である．SSL/TLS プロキシは業務端末がサーバと SSL/TLS 通信を行う場合，業務端末と SSL/TLS プロキシとの間，SSL/TLS プロキシとサーバとの間，それぞれの計 2 回の SSL/TLS 通信が行われる．SSL/TLS 通信の DPI は，SSL/TLS プロキシ上で行われる 2 つの SSL/TLS 通信の間で参照可能な平文やパケットと秘密鍵を用いることにより可能になる．

この SSL/TLS プロキシの手法は，SSL/TLS 通信を用いているが，SSL/TLS 通信を行うクライアントが，通信先のサーバを識別した上で安全に情報をやりとりすることが出来ない点が指摘されている [4] ．さらに，SSL/TLS プロキシは SSL/TLS 通信の処理が 2 回行われるため計算コストが高い．

業務端末が多い環境で SSL/TLS プロキシを配備する時，SSL/TLS 通信の多量なコネクションを扱うため，計算コストもまた必要となる．これを解決するために，アプライアンス製品が導入される事が多い [3]．しかしながら，この手のアプライアンス製品の SSL/TLS プロキシは高価であり，また，導入した場合は物理的なネットワークの結線が必要なだけでなく，論理ネットワークの構築が制約される．これらの問題を解決するためには，運用・導入コストが低い，柔軟な SSL/TLS 監視基盤が必要である．

本稿ではソフトウェアによる API フックを用いた SSL/TLS 通信の DPI システム (SSL/TLS-API) を提案する．この提案する手法は，SSL/TLS 通信の平文データを端末側でそのまま取得するため，SSL/TLS プロキシで行う必要があった 2 回の SSL/TLS 通信の計算は不要となる．また，SSL/TLS 通信を DPI をする際でも，業務端末はサーバとの間のみで SSL/TLS 通信の接続が行われることになる．さらに，IDS/IPS などと DPI した結果を共有するときネットワークポリシに応じた転送方法が選択可能になる．

2 目的と要件

本章では SSL/TLS-API の目的と要件を述べる．

2.1 目的

SSL/TLS-API は，ソフトウェアで実現する DPI システムが求められる．エンタープライズネットワークの管理者は，様々な形のネットワークを構築する必要性がある．また，構築したネットワークに対しポリシーも柔軟に選択出来る必要性がある．このネットワークポリシーの選択を実現するために，DPI を行う方法は，業務端末やクラウドを含めた端末で実行可能なソフトウェアでシステムを作る必要性がある．

また，今後クラウド上を含めた多くの端末で NFV (Network Function Virtualization)，GPGPU，FPGA 等が利用可能になる．これは，従来ソフトウェアで実現不可能だった計算が可能になる事，高速なパケット転送が可能になる事を意味する．その結果，既存のアプライアンス製品のみでしか実現出来なかった機能を，ソフトウェアで実現出来る可能性がある．ソフト

ウェアで DPI システムを作成することにより，上記の特徴を生かし連携する事が可能になる．

以上より，SSL/TLS-API は業務端末などで利用可能なソフトウェアで実現する DPI システムの構築を目的とする．

2.2 要件

SSL/TLS 通信の DPI を行う方法は，ソフトウェアで実現すること，計算量を減らすことの 2 つを実現するため，業務端末上で SSL/TLS 通信の API フックを用いることを想定した．まず，SSL/TLS 通信の API フックを行う時の要件を述べる．次にエンタープライズネットワークで求められる要件を述べる．また，DPI 結果を集約し連携する時の要件を述べる．

2.2.1 SSL/TLS 通信の API フック

業務端末のアプリケーションが SSL/TLS 通信を利用したとき，実装に応じた関数が呼び出される．DPI を行うために，この SSL/TLS 通信の関数をフックする必要がある．DPI を実現するためには，2 つの情報が必要になる．1 つ目は，パケットデータの中身であり，SSL/TLS 通信においては平文データである．2 つ目は，通信の識別情報であり，TCP/IP のヘッダに含まれる送信元，送信先情報となる．

1 つ目の情報を取得するためには，SSL/TLS 通信の関数フック時に平文データを取り扱う関数を探し出す必要があり，対象の関数の呼び出しが行われるとき，その引数から平文データを取得する必要がある．また，SSL/TLS 通信の DPI の機能のみが必要であり，平文データの改竄はしない必要がある．

2 つ目の情報を取得するためには，1 つ目の SSL/TLS 通信の関数フックで取得した平文データと通信元，通信先の情報を結びつける情報が必要がある．また，SSL/TLS 通信の暗号特性を維持するため，接続先や認証機構等は変えない必要がある．

また，業務端末で流れる SSL/TLS 通信は，情報セキュリティを向上させるためであり，必要に応じてネットワークの管理者が DPI を行いたくない場合が存在すると考えられる．例えば，銀行の暗証番号などが挙げられる．このため，SSL/TLS 通信の関数フックを行う時，平文データを取得するかどうかを判別する機構が必要になる．

2.2.2 エンタープライズネットワーク

エンタープライズネットワークは，1,000-10,000 台の中規模の業務端末を想定する．ネットワークの構成は基幹システムや業務端末が置かれるネットワーク等複数に分かれ，ネットワークポリシーによりそれぞれの接続可否が設定される．一方，ネットワーク規模が大きな場合，ミスを防ぐ目的でネットワークポリシーを必要な限り単純にする傾向があり，例外的なネットワークポリシーを少なくする必要性がある．

また，一般的な業務端末のネットワークは，業務遂行のみに利用するネットワークだけが存在する．さらに，エンタープライズネットワークで流れる情報は，企業の知財を含めた秘匿情報が想定される．以上より，業務端末で DPI を行った場合に，その結果を基幹システムネットワークに置かれた IDS/IPS 等と共有する事を実現するためには，ネットワーク管理者の要求レベルに応じた転送方法が必要である．

2.2.3 DPI 結果の分散処理

基幹システム内において，DPI 結果の情報を IDS/IPS，コンテンツフィルタ等と連携する事が想定される．DPI 結果の情報の利用は，その DPI した情報を単純に 1 対 1 で転送，参照する方法もあるが，これを 1 対複数で実現し，効率よく利用する場合も想定される．例えば，DPI 結果をコンテンツフィルターのためにパターンマッチングを行うとき，転送された単一のマシンで行うのではなく，より処理性能を求め，複数のマシンで分散し実行するなどが挙げられる．これを行うためには，DPI 結果を分散して共有する機能が必要となる．

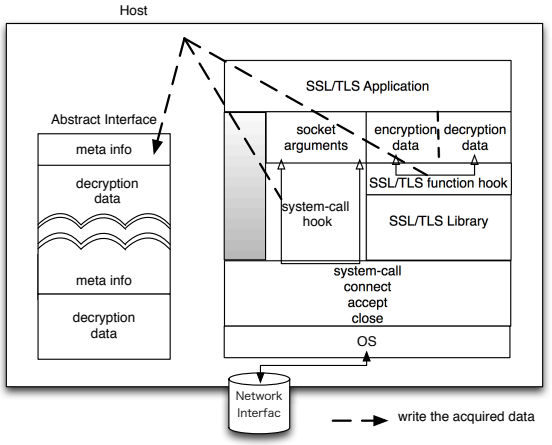

図 1: SSL/TLS Capture Plane

3 設計

図 1 と図 2 に概要を示す．SSL/TLS-API の DPI システムは，SSL/TLS Capture Plane と SSL/TLS Forward Plane の 2 つから構成される．

SSL/TLS Capture Plane は，アプリケーションが SSL/TLS 通信を行う時に利用する SSL/TLS 関数 API をフックすることで，関数 API の引数の情報から，復号データを取得する．さらに，復号データの情報とメタヘッダを抽象化インターフェースに書き込む．メタヘッダは，復号データの長さや SSL/TLS 通信で利用した TCP のフロー情報が含まれる．抽象化インターフェースは，Unix Domain Socket や名前付きパイプを利用する．

SSL/TLS Forward Plane は，2 つのシステムから構成される．1 つ目は，抽象化インターフェースに書き込まれたデータを読み込み，SSL/TLS 通信の DPI 結果を共有するホストへ 1 対 1 で転送を行う仕組みである．2 つ目は，1 つ目の 1 対 1 で転送を行うのではなく，メタヘッダに書き込まれたフロー情報を識別子とし DPI 結果を共有するホストへ 1 対 N で転送を行う仕組みである．ここで N は 1 以上である．

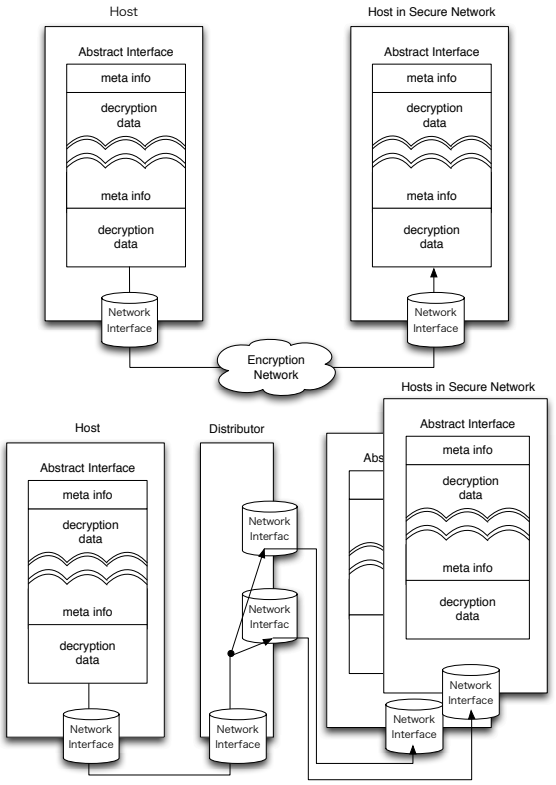

図 2: SSL/TLS Forward Plane

3.1 SSL/TLS Capture Plane

SSL/TLS Capture Plane はアプリケーションが SSL/TLS 通信を行うとき，SSL/TLS 関数の API を含む動的ライブラリをフックすることで行う．また TCP のフロー情報を得るため，SSL/TLS Capture Plane ではソケット関連の API もフックする．図 3 に関数 API フックをしたときに取得した情報の流れを示す．また，SSL/TLS-API の動作定義を行うフォーマットを図 4 に示す．さらに，SSL/TLS-API の抽象化インターフェースへ書き込むメタヘッダ- 復号データの構造を図 5 に示す．

図 4 の global はシステム全体に反映される設定である．インターフェースを表す if は SSL/TLS 関数を読み込んだ結果を書き込むインターフェースを定義する．exclude はドメインのリストを表し，正規表現でマッチしたドメインへ SSL/TLS 通信へ接続した時，if へは書き込まない指定を行う．アプリケーション毎に個別設定を行うためにバイナリが置かれたパスを表す

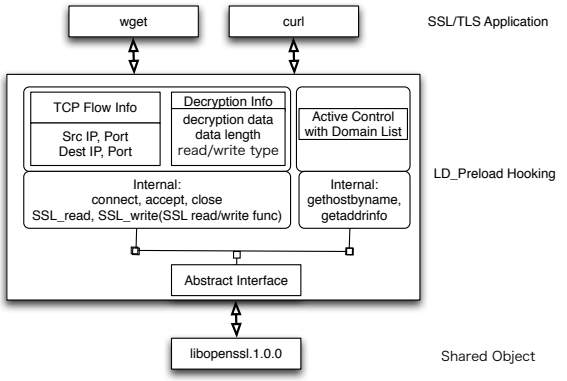

図 3: SSL/TLS-API と関数 API フックの流れ

```
global:
  if = /var/lib/ssl
wget:
  path = /usr/bin/wget
  if = /var/lib/ssl_wget
  exclude = .*\.bank\.co.\.jp
curl:
  path = /usr/bin/curl
  if = /var/lib/ssl_curl
```

図 4: 設定ファイル (例)

path を識別子として設定の上書きが出来る．

図 3 の connect，accept，close のシステムコールをフックすることで，ソケットのファイルディスクリプタを監視する．API フックを行った関数が，監視されたソケットのファイルディスクリプタを経由して，TCP コネクションの接続情報を取得する．取得した接続情報を，TCP のフロー接続情報として利用する．また SSL/TLS 関数をフックし，復号データの読み込み，書き込みが行われたとき，その関数内部でソケットのファイルディスクリプタと結びつきがあるかを確認する．ソケットのファイルディスクリプタの結びつきが SSL/TLS 関数の内部で確認できた時，かつ，exclude で除外されていない時，if で指定されたインターフェースへ書き込む．

図 5 の type はメタヘッダの種別を表す．rw

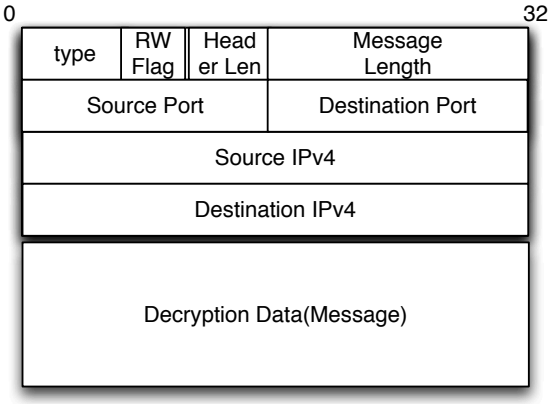

図 5: メタヘッダと復号データ

は SSL/TLS 関数の復号データの読み込み，書き込みフラグを示す．Header-Length はメタヘッダの長さ，Message-Length は復号データの Decryption Data の長さをそれぞれ示す．Source Port，Source IPv4，Destination Port，Destination IPv4 は SSL/TLS 通信で接続したソース，ディスティネーションの IP とポート番号をそれぞれ示す．SSL/TLS 関数の呼び出し毎に図 5 のデータが if へ書き込まれるタイミングとなる．

SSL/TLS 関数をフックすることで，読み込み，書き込みの復号データを取得し DPI を実現している．その結果，SSL/TLS プロキシでは，SSL/TLS 通信のリクエスト毎に 2 回の SSL/TLS 通信の暗号計算が必要であったが，この暗号計算コストは不要となる．また，SSL/TLS 関数のフックを行う時，関数フックの呼び出し後に，本来の関数呼び出しの結果と同一にするよう注意することで，業務端末の SSL/TLS 通信は，サーバの証明書を含めた識別が可能になり安全な通信が可能になる．

以上より，業務端末上で SSL/TLS 通信の DPI を行う時，エンタープライズネットワークの管理者は，ポリシーに応じアプリケーションの選択と DPI の有無が可能となる．

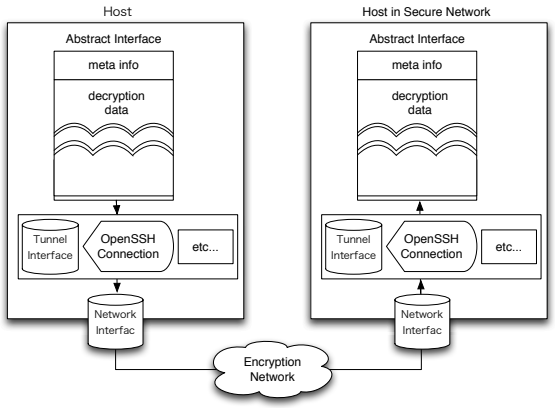

図 6: Secure Foward

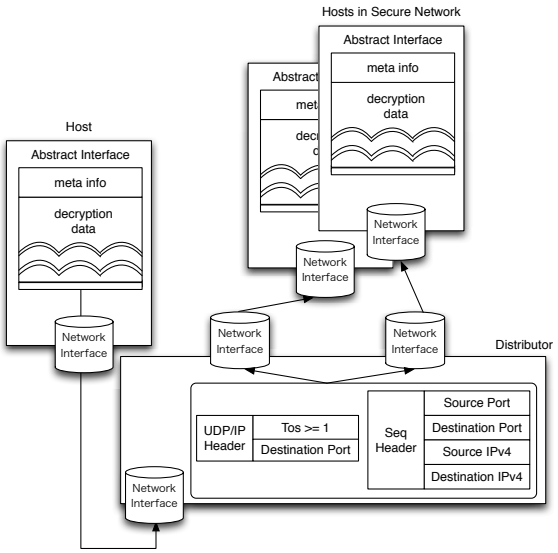

図 7: Distributed Forward

3.2 SSL/TLS Forward Plane

SSL/TLS Forward Plane は SSL/TLS Capture Plane で抽象化インターフェースに書き込まれた情報をファイアウォールや IDS/IPS と共有するため，ホスト間で転送するための機構である．SSL/TLS Foward Plane は，Secure Forard と Distributed Forward で構成される．図 6 ， 7 に概要を示す．

Secure Foward は VPN 暗号化等を経由し，送信元のインターフェースを送信先の抽象化インターフェースへ安全に転送するための機構である．具体的な送信方法は，OpenSSH6.7 からサポートされた Unix Domain Socket のリモートへの転送 (図 8) や，OpenVPN などの VPN 暗号化経由で抽象化インターフェースをコピーする事を想定しており，送信元と送信先の抽象化インターフェースが 1 対 1 でコピーされる仕組みである．これは，一般的に広く知られた暗号化通信のファイル転送共有の仕組みと同様と考え，詳細は省く．Secure Foward はエンタープライズネットワークの業務端末が業務を遂行するためのネットワークしか存在しない場合でも，セキュリティを確保したネットワークのホスト転送が必要に応じて可能になる機構である．

Distributed Foward は，SSL/TLS Capture Plane または Secure Forward で転送された抽象化インターフェースから読み込み，その情報を，SSL/TLS Capture Plane のメタヘッダに記述されているソース，ディスティネーションの IP，ポート番号のフロー情報で分割し，複数のホストへ転送を行う．これは SSL/TLS 通信の DPI 結果をフローで識別した上で，複数ホストで共有を行う仕組みである．

抽象化インターフェースに書き込まれた情報を UDP パケットで分割して転送を行う．図 9 に UDP パケットのフォーマットを示す．フォーマットの type は識別子を，Pad はパディングを，Header Len はヘッダーの長さを，Sequence Number はシーケンス番号を示す．また，Source，Desitnation の IPv4，Port は，それぞれソース，ディスティネーションの IP，ポート番号を示す．フォーマットに含まれるシーケンス番号を元に，送信先では分割されたパケットの再構成を行う．もし送信先でパケットのタイムアウトを確認した時や，シーケンス番号が含まれないパケットが存在した場合は，送信元へ再送を促す．分割するパケットのサイズは，送信先の経路上における MTU 以下を想定する．また，UDP パケットを送信する時に IP header の tos field は 1 以上を付与する．フロー毎に送信先ホストを分割し転送を行うために，Distributor は IP ヘッダーの tos field，UDP のポート番号を参照し，ソース，ディスティネーションの IP，Port を元に送信先ホストを決定する．

Distributed Forward は，SSL/TLS Capture

```
-- Host  --
# /var/lib/unix_domain_sock を作成
$ ls -l /var/lib/unix_domain_sock
srwxrwxr-x 1 m m 0 Aug 1 23:13 /var
  /lib/unix_domain_sock=

-- Host in Secure Network --
$ ssh -R /var/lib/unix_domain_sock:
  /var/lib/unix_domain_sock \
  host
```

図 8: OpenSSH による抽象化インターフェースの共有 (例)

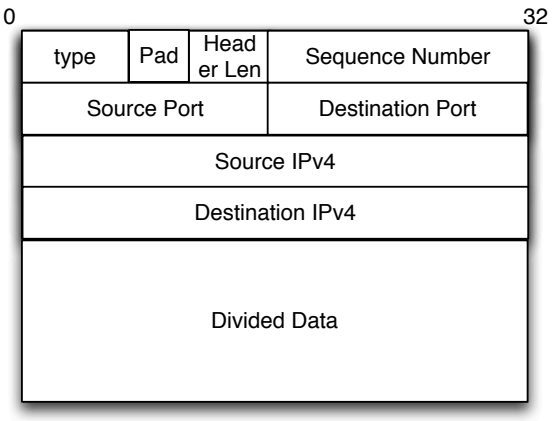

図 9: ホスト分割するための UDP パケットのフォーマット

Plane で読み込まれた SSL/TLS 通信の情報を，効率よく分散し複数のホストで参照するための仕組みである．これは，DPI と連携する場合に計算負荷が高い IDS ルールやコンテンツ参照フィルタを動作させる事を想定している．単独のホストではなく複数ホストに分割することで，CPU，ネットワークの帯域を活用することが可能になる．

Secure Forward による転送は，暗号通信を利用した場合，送信元，送信先のそれぞれで計算コストがかかる．SSL/TLS Capture Plane では，アプリケーションやルールに基づき出力インターフェースを変更出来るため，必要に応じて変更するか，または，ポリシーに基づきネットワーク構成を作る事が可能である．

以上より，エンタープライズネットワークの管理者は，ホスト端末上で SSL/TLS 通信の DPI した結果を，Secure Foward により安全に，Distributed Foward により，高速に多量に転送が可能になる．

4 関連研究

DPI 基盤として，SF-TAP [9] が提案されている．SF-TAP は Cell Incubator と Flow Abstractor から構成される高速な L7 解析器である．パケットキャプチャした結果を抽象化インターフェースで読み込む事が可能であり，本稿で提案した SSL/TLS-API も統合を目指している．

L7 の DPI ライブラリとして，nDPI [6] が存在する．nDPI を用いて SSL/TLS 通信を DPI する時，SSL/TLS プロキシと同じ手法を用いている．また，幾つかのアプライアンス製品 [3] だけでなく，ソフトウェアベースで実現された mitmproxy[1] などにおいても同様の手法である．

SSL/TLS プロキシは中間者攻撃を用いて実現されており，その問題点を Jeff ら [4] が指摘している．例えば，SSL/TLS 通信の DPI は可能になるが，SSL/TLS 通信が 2 つに分割される事，SSL/TLS プロキシは攻撃者から標的にされる事，クライアントから SSL/TLS 通信をサーバへ行った場合，サーバ証明書を検証出来ない事などの問題点を挙げている．

Justine ら [7] は，暗号化をした状態で DPI を可能にする提案を行っている．しかしながら，実用的な IDS のルールセット (3,000 個) を用いた場合，暗号通信の初期接続時間が数百秒程度かかるため，いまだ実用化には難しいと考える．

Adam ら [2] は SSL/TLS のサーバ証明書の検証に関する脆弱性を SSL/TLS 関数の動的リンクをフックし機能拡張することで解決する手法を提案している．この手法は Ubuntu で広く

利用されるアプリケーションの内，94 % がバイナリを変更することなく動作する事が可能と報告している．また，SSL/TLS ライブラリは OpenSSL，GnuTLS，JSSE の一部分と幅広い実装が行われている．

ANDR ら [5] は，Windows XP 上で，Internet Explore と Firefox のブラウザでやりとりされる通信情報を DLL injection を行うことで取得している．これは SSL/TLS 通信上のものも含まれたものが報告されている．

5 まとめと今後の課題

本稿では，SSL/TLS 関数をフックすることで SSL/TLS 通信の DPI を行う仕組みについて述べた．SSL/TLS 関数をフックする手法を用いることにより，1 回の SSL/TLS 通信で 2 回 SSL/TLS 通信が必要な SSL/TLS プロキシより計算量は低くなる．また業務端末で SSL/TLS 関数を透過的に実行する結果，SSL/TLS プロキシで指摘された業務端末がサーバ証明書を識別することが出来ない問題が解決可能になる．エンタプライズネットワークの管理者は業務端末で SSL/TLS 通信の DPI を制御可能であり，ネットワークポリシーに合せた構築が可能になる．

今後の課題として SSL/TLS 関数は，プラットフォーム，オペレーティングシステム，アプリケーション毎に呼び出し方法が異なるため，それぞれに対応し API フックを実装する必要がある．また，SSL/TLS 関数の利用方法は，動的リンクだけでなく静的リンク等のアプリケーションバイナリに埋め込まれる場合が想定されるため，これも課題である．さらに，本稿は SSL/TLS 通信のみを対象にしたが，UDP を用いた DTLS に関しても検討する必要がある．

現在，本稿で提案したソフトウェア API による DPI システムを実装中である．今後は，SSL/TLS 関数の API フックを用いた DPI システムを検証し評価を行う予定である．

参考文献

[1] Aldo Cortesi. mitmproxy. http://mitmproxy.org/index.html.

[2] Bates, Adam and Pletcher, Joe and Nichols, Tyler and Hollembaek, Braden and Tian, Dave and Butler, Kevin RB and Alkhelaifi, Abdulrahman. Securing SSL Certificate Verification through Dynamic Linking. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 394–405. ACM, 2014.

[3] A10 Networks Inc. Thunder adc. http://www.a10networks.co.jp/products/thunderseries/thunder-adc.html.

[4] Jarmoc, Jeff and Unit, Dell SecureWorks Counter Threat. SSL/TLS interception proxies and transitive trust. *Black Hat Europe*, 2012.

[5] Nordbø, André. Man-in-the-browser to retrieve content of SSL connections.

[6] ntop. ndpi. http://www.ntop.org/ products/deep-packet-inspection/ndpi/.

[7] Justine Sherry, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. BlindBox: Deep Packet Inspection over Encrypted Traffic. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pp. 213–226, New York, NY, USA, 2015. ACM.

[8] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568.

[9] Yuuki Takano and Ryosuke Miura and Shingo Yasuda and Kunio Akashi and Tomoya Inoue. SF-TAP: Scalable and Flexible Traffic Analysis Platform Running on Commodity Hardware. In *29th Large Installation System Administration Conference (LISA15)*. USENIX Association, November 2015.