Computer Security Symposium 2016
11 - 13 October 2016

MASQWARE: 通信プロトコルを偽装するマルウェア

長嶺 優大[1]　矢内 直人[1]　岡村 真吾[2]　藤原 融[1]

概要：HTTP などのプロトコル通信を任意のプロトコルに偽装して通信を行う方式としてフォーマット変形暗号（FTE）がある．これは必要以上に通信が制限された環境では有用な技術である一方，DPI によるネットワークトラフィックの監視をくぐり抜けるなど，マルウェアに利用されると甚大な被害を及ぼす可能性を持っている．本稿では，この潜在的な脅威の確認とその対策検討を見据えて、FTE を利用して既存の DPI をすり抜け外部と通信可能なマルウェアを提案する．

キーワード：マルウェア，偽装通信，フォーマット変形暗号，Deep Packet Inspection

Malware for Protocol Misidentification by utilizing Format-Transforming Encryption

Nagamine Yudai[1]　Yanai Naoto[1]　Okamura Shingo[2]　Fujiwara Toru[1]

Abstract: Format-transformation encryption (FTE) is an encryption scheme where a format of a communicated protocol is transformed to that of a different protocol. Alghough FTE enables users to avoid a strict network restriction, it may bring malware a profit to avoid check of network traffics such as the deep packet inspection (DPI). In this paper, we propose a new type of malware based on FTE in order to clarify a potential threat and its countermeasure.

Keywords: malware, Impersonation communication, Format-Transformation Encryption, Deep Packet Inspection

1. はじめに

1.1 序論

近年不正アクセスによる被害が急増し，対策が進められている．しかし，巧妙化する攻撃は施された対策を短期間で回避し，終わりの見えない状況が続いている．組織内の端末がマルウェアに感染し，機密データを外部に送信されたとき，流出を防ぐ技術の一つとして Deep Packet Inspection(DPI) がある．これは Intrusion Detection System(IDS) がパケットの送信元と送信先の IP・ポートを検査することに対し，DPI はパケット本体の中身まで検査することによってより詳細な検知を行う．DPI はパケットの送信元と送信先の IP アドレスとポートによりパケットの検知・遮断を行うシステムである IDS/Intrusion Prevention System(IPS) が行えない詳細な解析を補完し，組織内の重要データが外部へ漏洩することを防ぐ．一方で，DPI は一部国家において Youtube や Facebook といったサービスが国家によって制限するためにも使われている [1][2][3].

サービスの制限がある地域においては，DPI の検知を回避する技術を使うことで，禁止されたサービスを利用できる．DPI の回避技術の一つに The onion router(Tor)[4] がある．これは通信を複数のリレーエージェントと呼ばれる他のコンピュータを複数中継し，アクセス経路の出口以外では全て暗号化された通信内容を流す技術である．通信途中のパケットを取得しても内容及び本物の通信相手はわからないため，回避と同時に匿名性も保たれる特徴を持って

[1] 大阪大学
Osaka University, 1-5 Yamadaoka, Suita, Osaka, 565-0871, Japan
[2] 奈良工業高等専門学校
National Institute of Technology, Nara College, 22 Yata-cho, Yamatokoriyama, Nara, 6391080, Japan

いる．こうしたDPIの検知を回避する技術は複数開発され[1][5]，ネットワークの監視技術と回避技術が互いに発展している状況となっている．

回避技術は必要以上に通信が制限された環境では有用な技術である．一方で，DPIによるネットワークの監視をくぐり抜けることは，マルウェアに利用されると甚大な被害を及ぼす可能性を持っている．例としては，外部に用意されたサーバ(Command&Controlサーバ，C&Cサーバ)と通信を行うランサムウェアとスパイウェアが挙げられる．ランサムウェアは，ファイルの復号鍵をC&Cサーバへの送信やC&Cサーバからのコマンドを受け取る．スパイウェアは機密データをC&Cサーバに対して送信する．DPIが有効であればこれらの通信の検知や遮断が可能だが，回避技術で無効化された場合，甚大な被害が予想される．本稿ではこの潜在的な危険性の調査と対策を見据えて，マルウェアのプロトタイプを作成し，実験を行う．

1.2 貢献

本稿の貢献は，潜在的な危険性の検査ツールとしてプロトコル擬態通信を行うマルウェア(masqware)の作成，及びその有効性の検証として，ローカルホスト上にサーバを用意し，擬態通信が行われていることを確認したことである．外部サーバと通信をするマルウェアはスパイウェアも考えられるが，本稿では特に近年被害が急増しているランサムウェアとして実装する．作成するマルウェアのプロトタイプの動作は，二段階とする．まず，感染端末のファイルを暗号化する．次に，復号鍵をC&Cサーバへプロトコル擬態を施して送信する．masqwareは一度特権でインストールされてしまうとその端末から動作を確認することができず，かつ秘密裏に外部と通信するRootkit型を想定している．DPIに守られたシステム内でその動作を全うするマルウェアは現在確認されていない．

プロトコル擬態を行うマルウェアを作成するには，プロトコル擬態通信の実現，マルウェアの隠密性の確保といった課題を解決することが必要であり，それぞれ以下のアプローチで解決する．

プロトコル擬態通信の実現: ランサムウェアやスパイウェアといったマルウェアは，攻撃者が外部に用意したサーバと通信を行う必要がある．このサーバは，情報収集や遠隔操作に必要不可欠である．サーバとマルウェアが通信を行うためには，コネクションを確立し，互いに暗号化した情報の送受信，正しく復号することで情報を取り出すといった一連の動作が必要となる．ネットワーク監視のもとでこの課題を解決するために，プロトコルのフォーマットを変形し任意のフォーマットの暗号文を得られるフォーマット変形暗号(FTE)[6][7]を用いる．さらに、総合的な擬態通信を行うために、通信内容の統計的な傾向を利用者自身でプログラミングでき，かつFTEをサポートしている，擬態通信フレームワークのMarionette[8]を用いる．MarionetteはPythonとCによって書かれており，比較的小規模のプログラムとなっているため，その改変も行いやすい．

単一フォルダでの作成: マルウェア本体が感染対象にデフォルトでインストールされていないソフトウェア(PythonやRuby，追加のDLLファイルなど)を必要とする実装であった場合，必要なソフトウェアをバックグラウンドでインストール，その後実行の流れとなる．このような動作すると，より感染を気づかれやすくなる．例としては，インストールの権限を与えるウィンドウが開く，CPU使用率や通信量の増大，イベントログに痕跡をより多く残すことが考えられる．従って，起動には追加のファイルを求めないことが望ましい．これらの問題を解決するために，必要なファイルはマルウェア本体も含めて大きすぎない程度(数十メガバイト)の単一フォルダにまとめる，Windowsで確実に動作するようにマルウェア本体は実行形式ファイルの形で作成する．また，ユーザに求める行動は一度のダブルクリックのみとする．

本稿の構成は以下のようになっている．2章では脅威モデル及び要素技術について示す．3章では提案するmasqwareの設計方針とその具体的な構成について解説する．4章に実装したランサムウェアを用いて行った実験とその結果を記述し，それらについて考察する．5章では本稿に関連する研究を挙げる．最後に，6章に本稿のまとめと今後の展望を述べる．

2. 要件定義

2.1 前提条件

本稿で対象とする内容はマルウェアに感染した際の挙動の擬態であり，実際の感染のさせ方については議論しない．つまり，被害者はあらかじめ本マルウェアに感染しているものとする．対象のPCには，WindowsXP以降にデフォルトでインストールされるファイルが揃っているとする．加えて，偽装通信をするために，感染端末はネットワークに接続されている必要がある．加えて，感染端末のあるネットワークはDPIシステムによって，外部との通信に機密ファイルがやり取りされているか，不適切なサイトへ接続していないかなどを監視しているとする．対象にウイルス対策ソフトが導入されていた場合，シグネチャ登録による検知は，回避が比較的容易であることから対象外とする。しかし，プログラムの動作のヒューリスティック監視によって検知される可能性があるが，本稿の目的から離れるため議論しない．C&Cサーバは実際に活動しているランサムウェアを参考にコマンドの送信及び感染端末からの情報の受信を主な機能とする．

2.2 脅威モデル

本稿では，DPIによって外部との通信が保護された環境

において，DPI をすり抜け，外部サーバと通信を行うマルウェアを作成する．本稿の趣旨は外部ネットワークとの境界で情報漏えいを防ぐエンドポイントにおけるセキュリティ機能の検証である．このためには感染端末自体の監視と，ネットワークの監視の二つを想定する必要がある．端末自体の監視の回避として，感染後は自身を unlink することで，特権ユーザに対してさえもプロセス参照を防ぐ．これは特権ユーザの権限で動作する rootkit を参考とする．ネットワークの監視回避には，プロトコル偽装した通信により DPI システム，及び感染端末のプロセス監視から逃れることが必要となる．プロセスの unlink については，HTTP(S) 通信への偽装では，Internet Explore などの端末に既にインストールされているブラウザからの通信に偽装することも考える．以上を行うことにより，組織内の端末に感染し，長時間潜伏しながら情報を収集，送信しつつも検知を困難にすることが見込まれる．

3. 要素技術

本章では Masqware の設計にあたり必要となる要素技術について紹介する．

3.1 フォーマット変形暗号 (FTE)

FTE は入力として平文 (M) とフォーマット (F)，そして鍵 (K) を与え，出力としてフォーマットに沿った暗号文を得る暗号方式である．鍵は FTE のメッセージを始めに AES で暗号化するために与える．これは AES の仕様となっている．フォーマットの記述には正規表現を利用でき，ユーザ自身が柔軟なフォーマットを作成できる．フォーマットにプロトコルを模したものを与えれば，暗号化されたデータは HTTP(S) や SSH，FTP など任意のプロトコルに擬態することができる．よって，DPI の検知ルールの記述に正規表現を用いた DPI(regular expressions based DPI, regexDPI) の検知ルールと同じ記述が可能となる．この特性を利用すると，検知ルールの設定ファイルがわかれば，そのルールで暗号化することで，DPI の検知を容易に回避することができる．Kevin らの研究 [6] によると，実際に FTE-powered Tor Browser Bundle を用いて，中国の様々なサービスを禁止する DPI システムの The Great Firewall of China(CFG) を回避し，中国では禁止されているサービスの Youtube，Facebook へのアクセスが成功している．

3.2 Marionette

DPI などのアプリケーション層でのフィルタリング対策として，パケットペイロードの難読化が提唱されている．その目的はプロトコル隠蔽であり，アプローチとしてはメッセージのランダム化，トンネル化，暗号文フォーマットを利用した擬態暗号化の大きく三つに分けられる．各方法は対象となる環境やフィルタによって威力は限定的である．具体的には，ランダム化はデータ部が全く意味をなさないため，検知が良いうである．トンネル化はプロトコルのホワイトリストの設定により無効化される．擬態化は，HTTP の GET メソッドの後には”200 OK”や”404 Not Found”といったレスポンスが来る，といった通信におけるパケットの流れの統計である，ステートフルプロトコルセマンティクスを調べることで検知できる．Marionette はこれらのプロトコル隠蔽手法を組み合わせることで，より高度な難読化通信を提供する．従来では困難であった，ステートフルプロトコルセマンティクスの擬態をサポートしている．ユーザは擬態対象のプロトコルの状態 (HTTP であれば GET，POST，Not Found など)，その状態になりうる確率，またプロトコルのフォーマットを操作できる．記述には，専用の Domain-Specific Language(DSL) を用いて，比較的容易な開発及びテストが可能である．プロトコル擬態の機能はプラグインによって任意の暗号方式を使用できる．現在の実装では，サーバとクライアントプログラム側で Marionette がプロキシとして動作し，通信を行う．

4. Masqware

本節ではプロトコル擬態通信を行う masqware について，その設計方針，及び構成について述べる．

4.1 設計方針

擬態通信: 本稿では，任意のプロトコルに擬態した暗号文を得るために，任意のフォーマットの暗号文を得られるフォーマット変形暗号 (FTE)[6][7] を用いる．また，偽装したデータを使った暗号通信には，総合的な擬態通信を行うフレームワークの Marionette を用いる．Marionette を使う利点は，暗号化・復号に任意の暗号技術の導入が可能であるため，FTE の導入・拡張が容易だからである．Mrionette はサーバとクライアントプログラムがある．利用方法は，まず C&C サーバ側を，通常の HTTP などを受け付けるサーバとして実装する．この際，偽装通信を考慮せずに実装し，任意のポート上で稼働させる．そして，サーバが稼働しているポートと同じポート上で Marionette のサーバプログラムを起動する．以降は，サーバへの通信は偽装通信の場合は Marionette へ，それ以外の通信は通常通りに処理される．一方，マルウェアに感染する端末では，クライアントプログラムをサービスが動いていない任意のポート上で稼働させる．可動が成功すれば，そのポートへの通信は全て偽装通信が行われる．マルウェア側では，クライアントプログラムが稼働しているポートに対して，通常の HTTP などのパケットを送信するだけでよい．偽装通信はクライアントプログラム越しに，通信先の Marionette サーバプログラムと行われる．

単一フォルダでの作成: 単一フォルダでの作成は，依存

ファイル一つに集めることで行う．具体的には，稼動するための設定ファイルとマルウェア本体が利用する DLL ファイルである．DLL ファイルといったバイナリは ntdll.dll といったどの Windows でも持っているファイルは含めない．サイズを削減するために，起動に必要な設定ファイルはその内容をソースコード内に埋め込んでからコンパイルする．この手法は必要なファイルを含める性質上，フォルダサイズが大きくなるが，ある程度大規模なソフトウェアのインストーラは数百 MB から数 GB のものがあるため，有効なサイズであると考えられる．

ランサムウェアの仕様:　ランサムウェアとしての機能実装には，FTE や Marionette が Python で実装されていることを踏まえ，Python の機能を用いる．本稿では，Python ライブラリの Crypt.Cipher による AES 128bit CBC モードで暗号化を行う．暗号化の対象は，My (Document/Downloads...etc) といった，ユーザ個人のファイルが置かれるフォルダを対象とする．全ファイルの暗号化後，復号鍵を C&C サーバへ送信する．現在の実装では，ファイルの復号に必要な秘密鍵 key，及び CBC モードで用いる初期化ベクトル (Initial Vector，IV) の二つを送信する．送信形式は，”http:<serverIP>?key=<value>&iv=<value>”，という URL 形式を取り，アクセスしたページ URL が鍵情報となる．

2.1 節で述べたとおり，C&C サーバの主な機能はデータの授受，及びマルウェアへのコマンドの送信である．これは実際に活動しているランサムウェアを参考にする．具体的なコマンドの内容，受信するパケットのフォーマットなどについては今後の課題としたい．現在の実装では，コマンドを送信する，及び受け取った GET メソッドに対してレスポンスは送り返さない．アクセスされた URL をサーバプログラムを起動した端末上で表示する機能のみを持つとする．

4.2　構成

masqware のプロトタイプ実装は，大きく分けて三つで構成されている．まず，Marionette のクライアントプログラムである marionette_client.exe，全体を統括する機能を持つ masqware.exe，二つの実行ファイルが動作するために必要な設定ファイルである．masqware.exe の機能は marionette_client.exe の起動，ファイルの暗号化，復号鍵の送信である．設定ファイルには，Marionette が動作するポートや接続先サーバの IP が記述されている．marionete_client.exe と設定ファイルは，masqware.exe に埋め込むことで，masqware.exe 一つに統合するできる．しかし，本稿では，暗号化のフォーマットの変更など，実験用のパラメータを容易に変更するために敢えて分割し，それらをまとめたフォルダをマルウェア本体とする．

プロトタイプ実装の masqware.exe の動作は三段階となっている．感染対象の PC で masqware.exe が起動されると，まず内部で marionette_client.exe を実行する．これにより，予め指定されたポート上で FTE 通信を行うためのプロキシが稼動する．次に，PC のユーザ自身のファイルが保存されているフォルダを検索し，順に暗号化していく．最後に，暗号化が終了すると Marionette プロキシ越しに，復号鍵情報を含む GET メソッドを送信し，脅迫文を表示した後に終了する．送信したパケットは，DPI では擬態したプロトコルとして判定されているため，HTTP が禁止されている環境でも DPI をすり抜け，HTTP 通信を行うことができると考えられる．

図 1 にプロトタイプ実装における masqware の構成を示す．masqware の masqware.exe は marionette_client.exe を含む形で実装するが，本稿では単一フォルダを masqware として定義しているため，masqware.exe と marionette_client.exe を含む設計となっている．これらの実行ファイルを統合することは今後の課題としたい．C&C サーバの構成は，偽装通信を行うプロキシ marionette_server.exe とマルウェアからのデータの受信やコマンドの送信を行う server，そしてサーバと偽装通信を行うための設定ファイル群となっている．

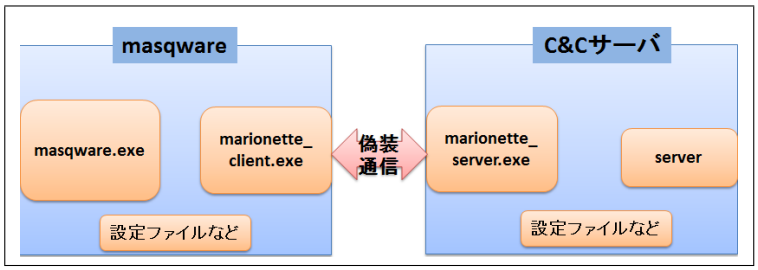

図 1　masqware 構成

5.　評価

本章では，4 節で述べた FTE 通信を行うマルウェアのプロトタイプとしてランサムウェアを実装し，その通信内容の検証を行う．その後今後の課題について述べる．

5.1　実験内容

実験は windows7 上でローカルホストにサーバを立てて行う．サーバと同じ待受ポート上で marionette サーバを動作させる．サーバは HTTP の GET・POST メソッドで接続があると，その接続要求が行われた URL をプロンプトに表示するようになっている．次に，サーバ・Marionette サーバが動作している状態で，masqware.exe をローカルホスト内で実行する．masqware.exe はランサムウェアとして，実験用に指定したファイルを暗号化する．4.1 節で述べたように，プロトタイプの暗号化方式には AES 128bit の CBC モードを用いる．暗号化が終了す

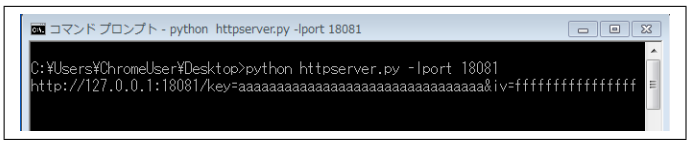

図 2　送信された鍵情報の確認

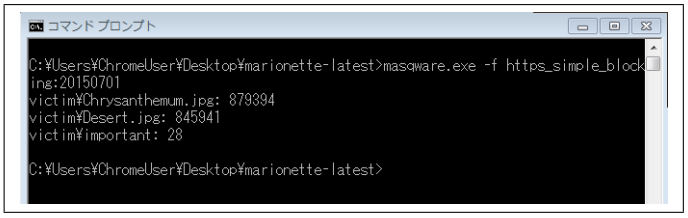

図 3　masqware 起動画面

ると，復号鍵と IV を FTE で暗号化した情報としてサーバへ送信する．送信の成否はサーバ側のプロンプト上に情報が表示されるかで確認する．FTE 通信の確認には，Windows のループバックアドレスへのパケットを収集するプログラム RawCap[9] を用いてパケットを pcapng 形式で収集し，Wireshark[10] で通信内容を確認する．実験では，HTTP の GET メソッドを”https:www.iana.org”への HTTPS 通信に擬態する https_simple_blocking.mar を用いた．これは Marionette プログラムに含まれているフォーマットの一つである．

実験に使用した環境を表 1 に示す．尚，Wireshark は 2.0.3，RawCap はバージョン 0.1.5.0 を使用した．

表 1　実験環境

CPU	Memory	OS
Intel Core i5 M540 2.53GHz	1GB	Windows7 Enterprise SP1

5.2　実験結果

図 2，図 3 はそれぞれ，鍵情報が C&C サーバに鍵情報が送信されてきたことの確認，及び masqware.exe を端末から起動した場合のプロンプトである．そして，図 4 は HTTP の GET メソッドが HTTPS に擬態した結果である．画像は RawCap で取得したパケットを Wireshark で閲覧したものである．masqware.exe から鍵情報のデータが送信されてきていることが確認できる．鍵情報は簡単のため，復号鍵が 0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa，IV が 0xffffffffffffffff としている．

本プロトタイプ実装の masqware.exe は，引数により擬態するプロトコルを指定できる実装となっている．実行後，暗号化対象フォルダ victim 内のファイルを暗号化し，各ファイルの暗号化が終了すると暗号化後のファイルサイズを出力する．

図 4 より，暗号化されていることが確認できる．データ部は正規表現で，`"\x17\x03\x03\x01\xbe.*$"`という

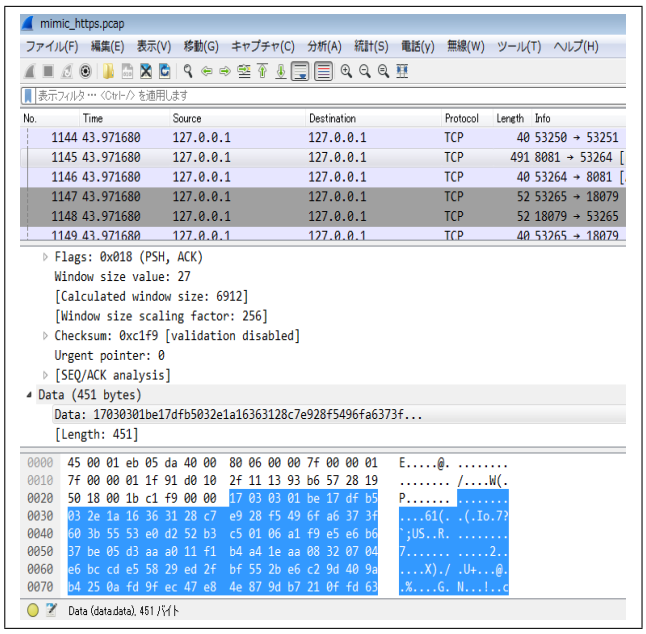

図 4　HTTPS 擬態パケットのキャプチャ結果

フォーマットで暗号化されている．4，5 バイト目 (0x01be) は，TLS 通信におけるデータ長を表し，その後に実際の暗号文が続いている．正しく TLS 擬態できた場合，Wireshark の Protool 列に”TLSv1”と出るはずであるが，”TCP”として認識されている．これはつまり，擬態が不完全であると考えられる．同様の問題は，HTTP に擬態するフォーマットにおいても確認されている．従って，HTTPS への擬態フォーマットには何らかの修正が必要であると考えられる．修正については今後の課題としたい．

5.3　今後の課題

5.1，5.2 節で行った実験では，簡単のためサーバやプロキシプログラム (Marionette) を全てローカルホスト上で動作させ，実験を行った．今後の課題はより実際の攻撃対象に近い実験環境として，小規模のネットワークを構築し，遠隔で動作しているサーバ・プロキシに対しての通信や regexDPI によって通信が制限された環境において DPI のすり抜け実験が考えられる．

本稿で偽装通信を行うために用いた，ネットワークの高度な偽装通信を提供するフレームワークである，Marionette に付属しているフォーマットは，プロトコルのヘッダのみを偽装する実装となっている．よって，HTTP 通信であれば，メッセージの長さが格納される Content-Length を検証すれば検知できる．また，Content-Type 部に暗号文を埋め込むフォーマットもある．このフォーマットを用いた場合，通常ではありえない内容が記述される．よって，検知は容易である．以上のように，付属しているフォーマットは単純なため，フォーマットの改善は必要不可欠であると考えられる．同様にプロトコルセマンティクスの改善も必要となる．また，現在の実装ではフォルダが本体となっ

ているが，一つの実行形式ファイルに統合することで，より簡単な動作での感染も期待できる．

本稿で偽装暗号化，及びその通信に用いた Marionette は，その設計上アプリケーション層での暗号化を目的としている．従って，より低い層の擬態は想定していない．しかし，低い層での擬態が可能であれば DPI 以外の検知システムもすり抜けることが可能であると考えられる．例としては，ICMP への偽装がある．よって，Marionette の設計を改良し，IP 層などのより低い層での擬態機能の開発を検討する．

6. 関連研究

最後に，本稿に関連した研究について紹介する．

6.1 rootkit

rootkit は持続性を持ち，感染した端末にステルスでアクセスするようデザインされたマルウェアである．動作は防御用の監視が行われているレベルよりも低いところで動作させ，検知されなくする．rootkit 型のマルウェアの研究として，ROP や Jump Oriented Programming を用いて既存のセキュリティ機構を回避する，Vogl らの Data-only Malware[11] がある．ROP などの自身のコードを持たない攻撃は，実行コードのアドレスが離れ，追跡が困難になりやすい．この問題を解決するために，Graziano らの ROPMEMU[12] が開発されている．これは ROP を行うマルウェアから，全自動で ret 命令によるジャンプを除いたバイナリを生成する解析補助ツールである．他に解析補助ツールとして dream++[13] がある．これは Hex-Ray のデコンパイラを比較対象に，より可読性の高いデコンパイラである．実装としては，Hex-Ray の解析用プログラムである IDA Pro のディスアセンブル結果を用いて，数学的な論理演算や多重ループを削減することで，可読性の高いソースコードを生成する．

6.2 Deep Packet Inspection

Tamer らの調査 [14] では，DPI システム実装における課題とアルゴリズム，ソフトウェア，ハードウェアでの実装によるスループットの比較を行っている．DPI システムで主に考えるべき二つの要素とは，大量のデータ処理に耐えうるメモリアクセス率を効率化するデータ構造の作成，及び膨大なシグネチャマッチングを処理するための，高スループットなアルゴリズムの作成である．前者のデータ構造の作成については，シグネチャを固定長のハッシュ値として保存することで総サイズを縮小する bloom filter や Content Addressable Memory(CAM) が挙げられる．後者の一致作業の高速化は，Boyer Moore(BM) 法，Knuth Morris Pratt(KMP) 法が挙げられている．Finite State Machine(FSM) を用いた DPI システムは，フィルターのルールの記述に regular expression を使えるため，非常に高い柔軟性を持った検知が可能である．しかしその反面，柔軟性を実現するために Deterministic Finite Automata(DFA) のサイズが肥大化しがちである．この問題を改善するために，DFA を改良し，メモリアクセスの削減による高速化を行った構造として D^2FA がある．この中でも，正規表現を使ってフィルタを操作することができる DPI を regular expression based DPI(regexDPI) と呼び，その柔軟性から注目されている．

6.3 DPI 回避技術

DPI は不正な通信を検知するために有効な手段であるが，同時に国家によって，国民のアクセスが制限されてしまう事例も発生している．DPI により，特定のサービスやプロトコルが禁止された地域において，禁止されたサービスを利用するための手法としては，The onion router(Tor) がある．しかし，これに対しての攻撃や妨害手法が提案されている [1]．DPI をすり抜ける方法として，ランダム化，トンネル化，擬態化がある．ランダム化は obfs4[15], ScrambleSuit[16], Dust[17] が提案されている．これは DPI においては，正当なプロトコルのリストを作成し，それのみを許可する white listing により防止できる．トンネル化の手法の実装としては，FreeWave[18]，Facet[19] がある．これらは通信をしている両者間以外では通信内容が一切わからないようになっているが，パケットとしては不自然なため，比較的簡単に防止される．擬態化については次節で述べる．

6.4 プロトコル擬態化

擬態化は，本来の通信内容を他のプロトコルに偽装する，暗号通信の亜種である．擬態化の方法は，プロトコル自体を擬態化するフォーマット変形暗号 (FTE)[6]，StegoTorus[20] がある．FTE は平文と任意のフォーマットを与え，暗号化することで，そのフォーマットに沿った形で暗号化する手法である．Kevin[6] の研究では，FTE を使った通信で，中国で使われている DPI システムである Great Firewall をすり抜けることが確認されている．また，同じ著者により FTE 暗号を提供する API が提供されている [7]．単純にフォーマットを変換して送信するだけでは，トラフィック上でプロトコルに偏りが生まれて検知しやすくなる．この対策として，レスポンスの内容をより現実的な振る舞いをするようユーザが操作できる Marionette[8] がある．StegoTorus ではヘッダに擬態データを埋め込み，ステガノグラフィで HTTP に見せる手法を取られている．DPI では擬態しているプロトコルを禁止することにより対策ができるが，対象のサービスによっては禁止できない場合もある．

7. まとめ

本稿では，プロトコル擬態通信を行うマルウェアの潜在的な危険性を確認するために，プロトコル擬態するマルウェアである masqware のプロトタイプを作成し，偽装通信が行われていることを確認した．プロトコル偽装の暗号化に FTE，暗号通信の実現には Marionette を用いた．プロトタイプはランサムウェアとして実装し，暗号化が終わると復号鍵をサーバへと送信する．送信は Marionette プロキシを仲介したプロトコル擬態通信を行った．今後は，最初に Wireshark で偽装したいプロトコルとして認識されるようフォーマットを改良する．その後，現実的な環境においての攻撃についての考察を行うために，regexDPI が動作している実環境での実験，及び偽装できる範囲を拡張するために Marionette の改良を行う．

参考文献

[1] Philipp Winter and Stefan Lindskog. How the great firewall of china is blocking tor. In *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet, Bellevue, WA, USA, August 6, 2012*, 2012.

[2] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. Internet censorship in iran: A first look. In *Proceedings of the 3rd USENIX Workshop on Free and Open Communications on the Internet, Washington, D.C., USA, August 13, 2013*, 2013.

[3] Map: Here are the countries that block facebook, twitter, and youtube. http://www.motherjones.com/politics/2014/03/turkey-facebook-youtube-twitter-blocked.

[4] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004.

[5] Daiyuu Nobori and Yasushi Shinjo. Vpn gate: A volunteer-organized public vpn relay system with blocking resistance for bypassing government censorship firewalls. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation*, pages 229–241, Seattle, WA, 2014.

[6] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Protocol misidentification made easy with format-transforming encryption. In *2013 ACM SIGSAC Conference on Computer and Communications Security, Berlin, Germany, November 4-8, In Proceedings of 2013*, pages 61–72, 2013.

[7] Daniel Luchaup, Kevin P. Dyer, Somesh Jha, Thomas Ristenpart, and Thomas Shrimpton. Libfte: A toolkit for constructing practical, format-abiding encryption schemes. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 877–891, 2014.

[8] Kevin P. Dyer, Scott E. Coull, and Thomas Shrimpton. Marionette: A programmable network traffic obfuscation system. In *Proceedings of the 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 367–382, 2015.

[9] Rawcap. http://www.netresec.com/?page=RawCap.

[10] Wireshark. https://www.wireshark.org/.

[11] Sebastian Vogl, Jonas Pfoh, Thomas Kittel, and Claudia Eckert. Persistent data-only malware: Function hooks without code. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.

[12] Mariano Graziano, Davide Balzarotti, and Alain Zidouemba. ROPMEMU: A framework for the analysis of complex code-reuse attacks. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, China, May 30 - June 3, 2016*, pages 47–58, 2016.

[13] Khaled Yakdan, Sebastian Eschweiler, Elmar Gerhards-Padilla, and Matthew Smith. No more gotos: Decompilation using pattern-independent control-flow structuring and semantic-preserving transformations. In *Proceedings of the 22nd Annual Network and Distributed System Security Symposium, San Diego, California, USA, February 8-11, 2015*, 2015.

[14] Tamer AbuHmed, Abedelaziz Mohaisen, and DaeHun Nyang. A survey on deep packet inspection for intrusion detection systems. *CoRR*, abs/0803.0037, 2008.

[15] Tor project. obfsproxy. https://www.torproject.org/projects/obfsproxy.html.en.

[16] Philipp Winter, Tobias Pulls, and Jürgen Fuß. Scramblesuit: a polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, Berlin, Germany, November 4, 2013*, pages 213–224, 2013.

[17] Brandon Wiley. Dust: A blocking-resistant internet transport protocol. technical report. School of Information, University of Texas at Austin, 2011.

[18] Amir Houmansadr, Thomas J. Riedl, Nikita Borisov, and Andrew C. Singer. I want my voice to be heard: IP over voice-over-ip for unobservable censorship circumvention. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[19] Shuai Li, Mike Schliep, and Nick Hopper. Facet: Streaming over videoconferencing for censorship circumvention. In *Proceedings of the 12th Workshop on Privacy in the Electronic Society*, November 2014.

[20] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. Stegotorus: a camouflage proxy for the tor anonymity system. In *Proceedings of the ACM Conference on Computer and Communications Security, Raleigh, NC, USA, October 16-18, 2012*, pages 109–120, 2012.