コネクションレス型セキュアWWW方式の一考察

安田　仁、荻原　利彦、渡辺　徹
日本電信電話株式会社

あらまし:WWWにおけるセキュリティ手法として一般的なブラウザに実装されたＳＳＬ等のソケットレベルの認証・暗号技術を利用したシステムの構築が盛んである。しかし、近年アプリケーションレベルのセキュリティが要求されるようになってきており、これを提供する「セキュアWWWシステム」を開発したので報告する。また、本方式の端末APはブラウザから独立したローカルプロキシ方式を採用しており、ブラウザの種類やバージョンに非依存のシステム構築を可能としている。本方式によりWWW上の業務用APにおいてアプリケーションレベルのセキュアなシステムの構築が実現される。なお本稿では、本方式におけるローカルプロキシ方式でのセキュリティ処理の実用性について実験等により考察する。

A study of the connection-less secure communication over HTTP

Hitoshi YASUDA,　Toshihiko OGIHARA,　Tohru WATANABE
NIPPON　TELEGRAPH　AND　TELEPHONE　CORPORATION

ABSTRACT. The system using SSL mounted on Web browser as a security technique on WWW has been getting popular. Though, there has security problems in which can't be solved with SSL, we developed the "Secure WWW System" that realizes the security on WWW in the application level. The client application of this system can use the old version browsers that don't support SSL, because it applies the local proxy method which is independent of the browser.This system can be applied to more practical business applications on WWW. We have examined effectiveness for the service provided by the local proxy from several experiments.

1．はじめに

昨今のインターネットの普及を受けて、企業のインターネット接続が増加し、同時にそれまで専用線などの回線で構築されていた業務用ネットワークをインターネット上で再構築する場合が増えている。そのため、専用線ではあまり考えなくてもよかったセキュリティに関する問題についてインターネットというオープンなネットワークでは考えてシステム構築しなければならない。さらにHTTPプロトコルを利用するWWWシステムは、市販製品も多く技術革新も大変激しい。そのためWWWシステムに依存するセキュリティシステムについては、WWWソフトウェアのバージョン管理が煩雑になってくると考えられる。

さらに電子社員録や決裁処理システムなどのデータエントリ系の業務用APをWWWシステム上で構築しようとすると本人認証・否認防止の必要性が出てくる。そこで本稿では、既存システムへのセキュリティの導入とブラウザのバージョン等に依存しないセキュリティの構築手段として業務APに適用可能なセキュアWWWシステムを提案する。

2．ソケットレベルのセキュリティ

ここではSSL(参考文献[1]:Secure Sockets Layer)等のソケットレベルのセキュリティ手法を従来方式として取り上げる。

2．1．特徴

ソケットレベルのセキュリティ手法の特徴には以下のものが挙げられる。

①ブラウザとWWWサーバに実装

②データを隠蔽するための通信路のセキュリティ

③暗号鍵情報授受でネゴシエーションを行うコネクション型

2．2．課題

ソケットレベルのセキュリティ手法を使って実用的な業務用APを構築しようとすると以下のような課題がでてくる。

①適用できるブラウザが限定される。

利用するブラウザの種類・バージョンによって利用できないものも存在する。昨今の市販のブラウザは、数ヶ月毎にバージョンが上がっており、例えばＷｅｂサーバ側で意識しているＳＳＬのバージョンに合致したブラウザを利用するものとは限らない。また、ＳＳＬのバージョンが上がる場合や利用している暗号アルゴリズムの変更に対応する場合、利用されるＳＳＬとこれを実装しているブラウザが変更される可能性は大きい。これは、業務用システムを構築する上で、システムとして運用面・技術面でサポートしていくことは困難であることを示している。また、業務用システムでは市販のブラウザではなく専用APを用いて運用することも考えられ、その場合専用APにソケットレベルのセキュリティを組み込まなければならない。

②否認防止等アプリケーションレベルでのセキュリティには別方式が必要

ソケットレベルでのセキュリティは、通信路を流れるデータを第三者に見られても理解不可能なデータにする。しかし、いったんサーバ側においてＳＳＬで受けたデータを、人や業務用APに理解できるようなAPレベルのメッセージにするとそのデータに関するセキュリティ上の情報は失われてしまう。つまり、CGI-APにおいてはブラウザから送られてきた情報がセキュアな転送だったのか否かも認識できない。さらに、業務用APや企業間の契約処理システムなどは、通信時のセキュリティを確保することに加え、事後の証拠としてのセキュリティ上の情報が必要になってくる。つまり、ソケットレベルのセキュリティでは、通信相手本人が確かに送ってきたメッセージであるという証拠の保存ができない。

③コンテンツの必要な部分のみの暗号化によるスループット向上等が図れない。

暗号化・復号化は多倍長演算の処理が入るため、通常のインセキュアの通信に比べＣＰＵに掛かる負担が大きい。テキストの数キロバイト程度の通信や高速演算が可能なＣＰＵを搭載している場合では、問題にならないかもしれないが、動画などの大量のストリームデータや非力なＣＰＵを搭載している携帯端末などでは少なからずストレスを感じることになるだろう。動画などのストリームデータをＳＳＬ上で送る場合など実時間性を要求されるデータでは暗号化・復号化によって実時間性が失われてしまう可能性がある。ＭＰＥＧなどで圧縮をされた動画データならば、基本フレームのみ暗号化して差分データについては暗号化しなくても構わないといった場合でも、ＳＳＬでは同一コンテンツにおいてこのような部分的な暗号化は困難である。

3．コネクションレス型セキュアWWW方式の概要

本方式のシステム構成を図1に示す。

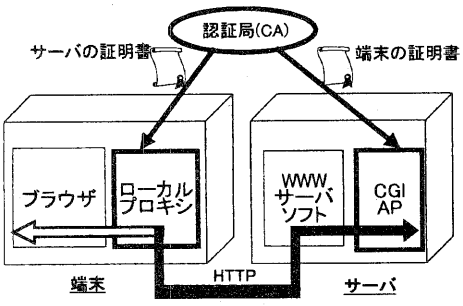


図1. コネクションレス型セキュアWWW方式

本システムは、端末側にローカルプロキシを、サーバ側はCGIを設置する。

a)端末環境

端末側においてブラウザのプロキシの設定に「localhost」と設定しローカルプロキシを指定する。また、ＬＡＮ環境から外部ネットワークのWWWサーバの使用を考え、外部ネットワークへのプロキシの指定は、ローカルプロキシの設定で行えるようになっている。こうしてブラウザからのＨＴＴＰプロトコルによる通信は、すべてローカルプロキシを介して行うことになる。

b)ユーザへの通知

ＳＳＬでは、セキュアなコネクションが確立した場合、鍵のアイコンによってユーザに知らせるが、本方式の場合では、コネクションレス型であるためセキュアな通信が行われた後にローカルプロキシのAP上とタスクトレイの信号機の形をしたアイコンによってユーザに通知する。セキュアな通信では青信号が、インセキュアの通信では黄信号が表示される。万が一、受信したデータの電子署名検証処理で改ざんなどが検出された場合には赤信号を表示する。さらに、過去のアクセスについてもローカルプロキシでセキュアかインセキュアかをＵＲＬと共に記憶しているので端末側で過去に遡ってサーバのメッセージ認証を確認することが可能となっている。

c)サーバ環境

サーバ側のセキュリティに関する処理はCGIにて行われる。セキュリティ処理を行うこのCGIは、業務用CGI-APと連携することが可能であり、単独でもコンテンツの暗号化が行える構成となっている。本システムでは、端末側のローカルプロキシからサーバ側のCGIまでの範囲で暗号化・電子署名が施されたメッセージが流れ、セキュリティが確保される。

4．本方式の特徴

①いろいろなブラウザの利用が可能

本方式の端末側のセキュリティに関する処理は、ブラウザから独立したローカルプロキシ方式のアプリケーションで行う。ローカルプロキシ方式を採用することによってブラウザに依存せず古いバージョンのブラウザや業務用に作られた専用APでも利用可能である。さらに将来的にも新しいバージョンのブラウザにも適用可能であり、セキュリティの処理に変更を行う場合にも、このローカルプロキシのみに変更を行うだけでブラウザには変更が必要ない。また、既存のWWWシステムに、セキュリティの機能を後から付加する場合や既存のWWWシステムとの共存もブラウザを変更することなく行える。

②WWWサーバソフトを選ばない

サーバ側でのセキュリティに関する処理はＣＧＩで行っているため、WWWサーバソフトに依存しない。これは以下のようなメリットがある.

・既存のWWWサーバへの導入コスト低減

・プラットフォームを選ばない

・ほとんどの市販のWWWサーバソフトに利用可能

③本人認証を意識したＣＧＩ処理が可能

メッセージが誰から来たものかということのセキュリティ的な意味での保証を得た後で処理の方法を決定することが可能になる。

従来方式では、安全な通信路を確保しその上でIDとパスワードの照合によってアクセス制御が行われる。本方式では、パスワード等に代表されるユーザ毎の秘密情報を管理する必要がない。ユーザ毎の秘密情報は、第三者に漏洩しないように細心の注意を払って管理しなければならず、これは運用コストの面でのデメリットとなる。ユーザ毎に発行される公開鍵の証明書に基づいてアクセスを制御することによってユーザ毎の秘密情報の管理が不要となっている。

④否認防止

メッセージ毎にセキュア化状態で保存可能であり、ユーザの電子署名によりサーバ側で受け取った情報の送信元が特定できる。蓄積されたデータのセキュリティは任意時点で検証できる。

⑤コネクションレス型のセキュリティ

コンテンツの選択的なセキュリティ処理が可能となり、実時間性を必要とするデータなどに効果的に適用できる。サーバ上のＣＧＩは、端末側のローカルプロキシとのネゴシエーション無しに任意にセキュア化されたメッセージを送信可能である。これにより帯域見合でセキュア化処理を行ったり、インセキュア処理を行ったりと柔軟な処理が行える。

5．実現方式

①HTTP+MOSS

ブラウザとWWWサーバの間で交換されるHTTPプロトコルのメッセージをMOSS（参考文献[2]:RFC1848: MIME Object Security Services）でカプセル化している。MOSSはコネクションレスで利用できる。また、MOSS化された情報はCGIにおいて、セキュアな状態で保存しておくことが可能である。このようなプロトコルメッセージのカプセル化を行うことで、これまでのアクセス方法を維持したままセキュリティを確保することができる。

②セキュリティ処理の契機

a) 端末からサーバへ

ローカルプロキシでは、セキュア化処理を行うべきＣＧＩのＵＲＬのリストを持っており、そこへGETメソッドもしくはPOSTメソッドのアクセスが行われた場合に下記に示すセキュアなアクセスを行う。通常のWWWをベースとした業務用APは、CGIを使ったGETメソッドとPOSTメソッドを利用しているため、本システムでは、これに対応している。

ユーザの利便性を高めるためこのURLのリストは、「*」を使ってURLの一部を省略可能で、URLの全ての文字を記述しなくても対応可能とした。

このURLのリストでのセキュア化の判断と後で述べるセキュアな受信のための処理によって、ネゴシエーションを必要せずコネクションレスで任意時点からのアクセスが可能となっている。

（例：ブックマークからのアクセス）

b) サーバから端末へ

サーバから端末にMOSS形式のセキュアなメッセージが送られてきた場合、ローカルプロキシはMIMEヘッダのContent-TypeからMOSS化されたデータか否かを判断して、復号化・署名検証の処理を行う。

③GET（セキュアな受信）

一般的には WWW サーバからの情報取得に用いられる。このためサーバからの情報をクライアント向けに CGI が MOSS 化する必要がある。暗号を行う場合には、送信相手を特定し相手の公開鍵を CA（認証局）から取得しなければならない。ここで GET メソッドを利用する WWW のアクセスでは、サーバにおいて端末側の鍵情報が必要になる。本方式では、端末側からセキュア WWW システムにアクセスする場合に、エンティティヘッダの拡張領域を介して端末側の公開鍵情報を送信する（図2）。サーバ側では、このヘッダを CGI が解釈して端末側の公開鍵を取得して暗号化を行う。なお、この GET メソッドにおいて本方式ではアクセスする URL は平文でネットワークを流れる。よって、URL 自体に秘密情報を含めることはできない。秘密情報を端末から送出する場合は、次に述べる POST メソッドを用いる。

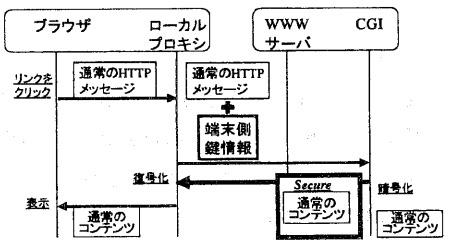


図2. GET:動作（セキュアな受信）

④POST（セキュアな送信）

アンケートなどのデータ入力フォームを使って端末からサーバにデータ送信を行うのが POST メソッドである。GET メソッドとは異なり、クライアントからサーバへの送出データにおけるセキュリティが重要である。これは、データエントリ系の業務 AP に多く用いられるものである。このセキュアなデータ送信の場合、端末側においてサーバ側の鍵情報が必要になる。本方式では、フォームの中の hidden 項目としてサーバの鍵情報を埋め込むことでこれを実現している(図3)。

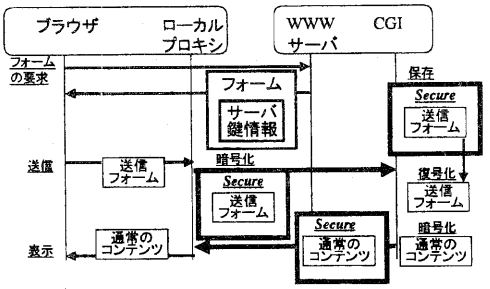


図3. POST:動作（セキュアな送信）

なお、サーバ側の鍵情報が埋め込まれたフォームを前述した GET メソッドで取得すれば、クライアント側でのサーバ検証も行える。サーバを信用して運用しているフォーム文に hidden 項目として埋め込まれたサーバ側の鍵情報が公開情報であるため、一般の LAN 環境ではフォーム文をインセキュアで取得しても問題無い。また、POST メソッドのリクエストに対するサーバからクライアントへの返却情報に関しては、CGI においてセキュア化するか否かは任意である。クライアントから送られてくるセキュアな POST メソッドにはクライアント側の鍵情報が含まれており、CGI は必要に応じてセキュアなレスポンスをクライアントへ返却可能である。また、RFC1867（参考文献[3]）に則って、ブラウザ上のフォームを使ってファイルのアップロードもセキュアに行うことができる。この場合は、ブラウザからローカルプロキシに送出されるメッセージの形式が、通常の「x-www-form-urlencoded」形式ではなく「multipart/form-data」形式になっており、ローカルプロキシではこれを解釈して鍵情報を取得している。

6．考察

本方式の場合、インセキュアの通常のWWW へのアクセスもこのローカルプロキシを経由することになる。ここでローカルプロキシのオーバヘッドが処理の負担に及ぼす影響を、実験を行い測定した(図4)。

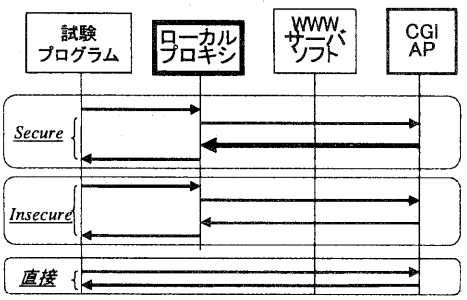
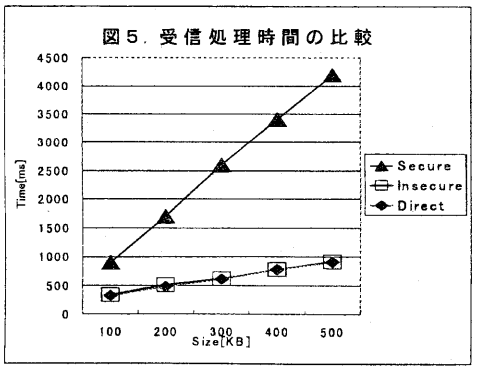


図4. 受信処理時間測定

コンテンツの表示時間は、コンテンツの種類に依存するものである。よって今回使用した試験プログラムは、ブラウザの代わりになるものであるがコンテンツの表示時間は計測対象外とし受信時間を測定するものである。

セキュアでのアクセスの時間とインセキュアでのアクセスの時間そしてローカルプロキシを使用しない場合の時間を測定した結果が図5である。このグラフからインセキュアでのアクセスとローカルプロキシを使用せずに直接アクセスした場合での差はほとんどなく、ローカルプロキシのオーバヘッドはないといえる。また、セキュアの場合とインセキュアの場合を比較すると使用している暗号アルゴリズムの処理性能及び MOSS フォーマットにおける BASE64 化によるデータ増大がグラフの差に反映された形となった。以上の実験の結果からローカルプロキシの介在オーバヘッドは実用上問題ないと考えられる。



7. まとめ

本システムでは、AP レベルでのセキュリティ確保により、CGI-AP における否認防止等を実現した。本方式では、GET と POST の2つのメソッドについてセキュア化の手法を提供しているが、WWW ベースの業務 AP のほとんどが GET と POST の2つのメソッドを利用した CGI であり、この2つのメソッドのセキュア化手法で十分である。また、MOSS の適用と鍵情報授受方法を工夫することにより WWW 技術がデータエントリ系業務 AP のプラットフォームとして有効に利用可能となった。今後は、SMTP・POP などのメールプロトコルについても本方式のようなローカルプロキシ方式が適用可能か検討を進める。

参考文献

[1]Alan O. Freie, Philip Karlto, Paul C. Kocher,"The SSL Protocol ver3.0" http://home.netscape.com/eng/ssl3/ssl-toc.html,March 1996

[2]S. Crocker , N. Freed , J. Galvin, S. Murphy ," MIME Object Security Services",RFC1848, October 1995

[3]E. Nebel, L. Masinter ,"Form-based File Upload in HTML"RFC1867, November 1995

[4]Berners-Lee, T., Fielding, R., and H. Frystyk,"Hypertext Transfer Protocol -- HTTP/1.0",RFC 1945, May 1996.

[5]Berners-Lee, T., Fielding, R., and H. Frystyk,"Hypertext Transfer Protocol -- HTTP/1.1" RFC 2068, January 1997.