

SSL プロトコル評価ツールの試作と各種 SSL 実装の評価

東角 芳樹* 武仲 正彦*

社会のさまざまなシーンで、インターネットが使用されている。このインターネットにおける商取引で、安全に商取引を行うための通信プロトコルが SSL や TLS である。今日では、SSL/TLS はインターネットの商取引だけにとどまらず組み込み機器のセキュアなプロトコルとしても、広く使用されている。本稿では、主に組み込み機器のテストを目的とした、SSL/TLS のテストセットを試作した。そして、本テストセットを用いて幾つかのブラウザの実装について実験し、組み込み機器の SSL サーバの実装上留意する点を明らかにする。

Making of SSL Protocol Evaluation Tools and evaluating various SSL protocol implementations

Yoshiki Higashikado* Masahiko Takenaka*

The Internet is used by various scenes in the society. In the transaction in the Internet, the communication protocol to transact it safely is SSL and TLS. SSL/TLS has been widely used as a protocol today the secure of not only the transaction of the Internet but also the built-in equipment. In this paper the test set of SSL/TLS to test the embedded device chiefly was made for trial purposes. And, the point noted in the SSL server of embedded equipment implement is clarified by experimenting on implements some browsers by using this test set.

1. はじめに

SSL/TLS は、今日のインターネットで最も広く使われているセキュリティプロトコルの一つである。SSL にはいくつかのバージョンが有り、現在はほとんど使われなくなっている SSLv1, SSLv2 をはじめとして、近年は、SSLv3 および TLSv1 が主に使われている。本稿では特に区別の必要のない限り、以後 TLS を含めて SSL と呼ぶこととする。SSL は、OSI の 7 階層では、トランスポート層上に位置するプロトコルである。SSL は、WEB ブラウザのセキュリティプロトコルのみならず、SSL を使用した VPN 接続というものがあり、比較的汎用的に使えるセキュリティプロトコルである。このため、近年では、各種の組み込み機器でも広く使われる様になってきている。TLS プロトコルは、IETF でインターネットの標準として RFC 規約化されているが、規約の記述が正常系を中心にかかれており、異常系に対する記述が少ないために RFC 規約のみでスクラッチから SSL プロトコルを実装することは困難である。このため、実際に異常系の SSL 実装は、各種実装によりまちまちになっていることが多い。そして、組み込みシステムでは、基本的にこれらの実装すべてを相手に正常に通信を行える必要があり、それぞれの実装の異常系との相性でフリーズするようなことは許されない。そこで、主に組み込みシステムをターゲットとして、SSL 実装の困難さを軽減することを目的に SSL プロトコルの正常系、異常系のテストを行うことのできるテストセットを試作した。本稿では、2 章で SSL/TLS プロトコルについて説明し、3 章で、今回制作した SSL/TLS テストセットについて説明し、4 章で各種の SSL 実装に本稿のテストツールを適用した結果、組み込みサーバの実装上どのような点に気をつける必要があるかについて述べ、5 章で 4 章の結果について考察する。6 章でまとめについて記述し、7 章で今後の課題について記述する。

2. SSL プロトコル

SSL は、当初、netscape 社のブラウザである netscape に実装された SSLver2 が 1994 年に発表されたものをはじめとして、その後、脆弱性が修正され、いくつかの機能が追加され、SSLver.3 が実装されている。また、インターネットの標準として、SSLver.3 をベースとして開発された TLS1.0 が RFC 2246[1]として公開されている。そして、後に脆弱性の対応で RFC4346[2]として、TLS 1.1 が規定されている。その後も脆弱性に対する対応がなされ現在では、RFC5246[3]として規定された TLS 1.2 が最新となっている。また、いずれの SSL もハンドシェイク時に使用されるプロトコルバージョン番号は、それぞれ TLS1.0 が 3.1, TLS1.2 が 3.2, TLS1.3 が 3.2 となっており、SSL ver3

* 株式会社 富士通研究所,

の後継として位置づけられている。

SSL は、下位プロトコルに TCP を用いてエンドポイント間で安全で信頼性のあるサービスを提供する様に設計されている。また、SSL は、単一のプロトコルで構成されているのではなく、図 1 の様に 2 層のプロトコルから構成されている[4]。

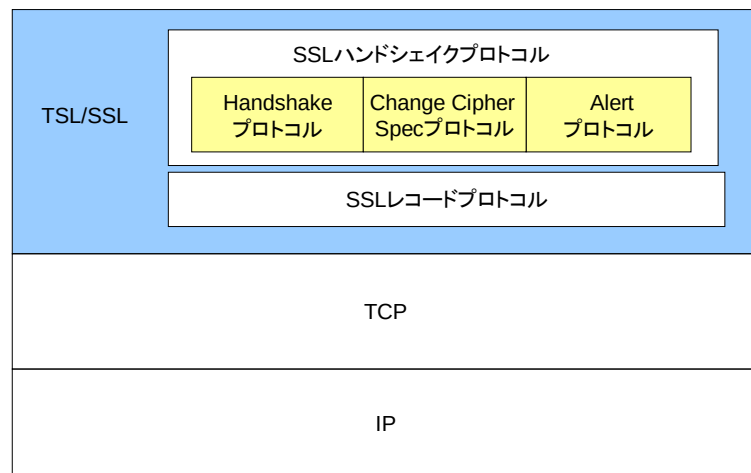


図 1 TSL/SSL プロトコル構造

・ハンドシェイクプロトコル

ハンドシェイクプロトコルは、サーバとクライアントの認証や暗号化に使用する共通鍵の生成、暗号通信の前準備を行う。また、ハンドシェイクプロトコルには、サーバ・クライアントそれぞれの証明書の交換を行う Handshake Protocol、暗号アルゴリズムを決定する Change Cipher Spec Protocol、警告情報を通知する Alert Protocol の 3 つのサブプロトコルから構成される。

・レコードプロトコル

レコードプロトコルは、データの送受信を行う。ハンドシェイクプロトコルで確立した鍵を使い、送受信するデータの暗号化、および、MAC(Message Authentication

Code)によるデータの完全性のチェックを行う。

以下に、SSL のハンドシェイクのシーケンス例を図 2 に示す。

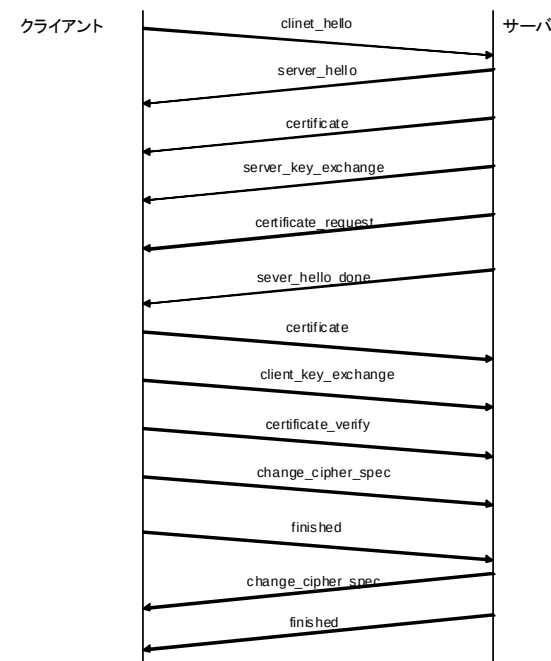


図 2.SSL の新規セッションのシーケンス

3. SSL テストセット

3.1 テストセットの作成の背景

各種 Web ブラウザに加えて、組み込み機器への SSL の普及により、ブラウザと SSL サーバとの組み合わせではない形態の SSL の実装を行うためには、実装をテストができる環境が必要である。しかし、従来は、脆弱性テストを目的とした SSL のテストセットが、Codonomicon 社[5]から市販されているのみであり、SSL サーバや SSL クライアントの実装時に必要となるテストを目的としたテストセットは、ほとんど存在しなかった。また、SSL のテストセットとして、OpenSSL の API を使ったものも存在した

が、OpenSSL の API をそのまま使ったものでは、パラメタ異常のような細かいテストを行うことができなかった。通常、SSL/TLS を実装するためには、そのプロトコル規約である RFC を理解する必要があるが、TLS に関する RFC では異常系の動作に関する部分の記述が少なく、実装上必要となる情報が乏しい。そこで、一般的にはオープンソースで提供されている SSL 機能、たとえば OpenSSL 等を移植することで、問題は解決するのであるが、組み込みシステムでは、メモリや CPU パワーなどの制限から、必要以上の機能を持ったプログラムを作成するよりも、できるだけ軽い実装を行う傾向にある。このため、実際に SSL のインプリメントを行う際には、OpenSSL 等のオープンソースを参考にして、異常系の振る舞いを調査することになる。しかし、SSL ライブラリの挙動を公開ソースをベースに始めから調査し開発を行うことは、開発者にとって負担が大きいものとなっていた。そこで、開発者の負担軽減を目的として、既存の実装の調査と組み込み向けの SSL のシステム開発に向けたテストセットの整備として、本稿のテストセットを開発した。

3.2 SSL/TLS テストセットの要件

SSL テストプログラムは、SSL プロトコルの実装のテストを行うために次の要件を考慮して実装されている。

- (1) SSL の各ハンドシェイクメッセージのシーケンスを自由にかつ容易に組み立てられること
- (2) 送受信されるメッセージのパラメタを柔軟に変更できること
- (3) テストで送受信した各シーケンスを後から確認できること

テストセットの実装では、(1)については、テストに必要な正常なシーケンスや異常な順番のシーケンスを送受信させることができ、かつ、複雑な記述をする必要がないように、UNIX のシェルスクリプトをベースとした、シェル・コマンドの組み合わせとして記述できるようにした。このことは、技術的には、各シーケンスが個別のコマンドで与えられることから、通信の継続性を保証するためにデーモンを介して通信を行う様に工夫する必要があったが、シェルスクリプト化することによりテストシーケンスの作成を容易にすることができた。(2)については、パラメタをシェル・コマンドに与える引数とすることで、パラメタの記述を容易にできるようにしている。(3)については、テストセットをターゲットシステムに適用した場合に、正常なシーケンスで送受信できたか、また、異常があった場合は、どのシーケンスで異常があったのかを正確に把握できる様に、各シーケンスにタイムスタンプを付け、送受信の方向を示すシーケンスチャートの作成と詳細のログが作成できるようにしている。

3.3 SSL テストプログラムの構成

SSL テストプログラムの構成を図 3 に示す。

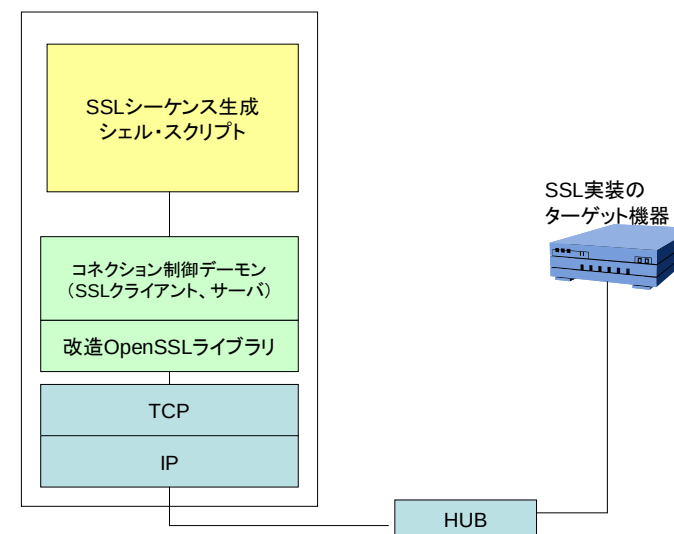


図 3. SSL テストセットの構成

本テストセットは、上位層から SSL シーケンスの生成のためのコマンド・スクリプト、にクライアント・サーバ機能で共通のコネクション制御デーモン、その下に OpenSSL を改造した SSL ライブラリからなる。改造 OpenSSL ライブラリは、SSL のレコードプロトコルやハンドシェイクプロトコルのパラメタ等をコネクション制御デーモンから変更できるように OpenSSL のライブラリの機能を改造して実装している。そして、シーケンスを生成する部分は、シェルスクリプトのコマンドとして実行するため、1 つ 1 つのハンドシェイク・シーケンス毎にコマンドが終了することになる。しかし、SSL のセッションを維持する必要があることから、SSL サーバ・クライアントのコネクション制御デーモンを置き、このデーモンと各シーケンスを生成する各コマンドが通信を行うことで、セッションを維持しながら任意のハンドシェイク・シーケンスを生成し、送受信することを可能としている。

3.4 シェルスクリプトについて

SSL テストセットでは、あらかじめ通常の SSL のハンドシェイクのシーケンスを記述した雛形のシェルスクリプトが下記のように用意されている。

```

if [ "$SARGNUM" -eq 0 ]; then
    # 使い方を表示
    usage
else
    # 引数チェック
    arg_check
    # Ctrl-C捕捉
    trap 'if test $SSLPID -ne 0; then kill -TERM $SSLPID; fi; kill -TERM $$_INT' INT
    # 試験開始
    while [ "$RETV" = 0 ]
    do
        case "$STATUS" in
            init)
                do_init
                ;;
            start)
                do_teststart
                ;;
            rcv)
                do_rcv
                ;;
            serverhello)
                do_serverhello
                ;;
            serverhellodone)
                do_serverhellodone
                ;;
            servercertificate)
                do_servercertificate
                ;;
            serverkeyexchange)
                do_serverkeyexchange
                ;;
            certificaterequest)
                do_certificaterequest
                ;;
            serverchangeipherspec)
                do_serverchangeipherspec
                ;;
            serverfinished)
                do_serverfinished
                ;;
        esac
    done
    do_end
    do_dispresult
fi
exit $RETV
end)
#do_end
break
;;
alert)
#do_end
break
;;
data)
do_data
;;
timeout)
#do_end
break
;;
disconnect)
echo "クライアントが切
断されました。再接続回数 $RECONNECT
0 ]; then
STATUS="rcv"
RECONNECT="expr $RECONNECT - 1"
else
RETV=1
fi
*)
echo $0 " Error :
Status : "$STATUS" Last Packet : "$SOLD_STATUS
STATUS="start"
break
;;
esac
done
do_end
do_dispresult
fi
exit $RETV

```

図 4. 雛形シェルスクリプト（一部分を抜粋）

テストシーケンスを新たに作成する場合には、この雛形のスクリプトコピーして使用する。たとえば送信メッセージの内容を変更する場合は、このスクリプトの後半部にある下線部の”ServerHello”を “ServerHelloDone”に変更すればよい。

```

do_serverhello()
{
    # ServerHello 送信
    $SSLTESTCMD -server -test send ServerHello -port $SSL_PORT
    RETV=$?
    set_status $RETV
}

```

このように、あらかじめ準備されている雛形ファイルをベースに変更を加えることで、比較的容易に任意のハンドシェイク・シーケンスを生成することができる。

4. 各種 Web ブラウザの挙動について

今回作成した、SSL のテストセットを用いることによって、各種ブラウザにサーバ側からさまざまなシーケンスを流した場合に、どのような応答が帰るか、あるいは帰らないかの挙動を知ることができる。これにより、組み込み機器に Web サーバなどを自前で実装する場合、接続できるシステムの挙動に合わせて接続できるような実装とする必要がある。

4.1 正常系の SSL サーバのシーケンス

正常系の実験に使用したサーバのシーケンスは、SSL の最も基本的なもので、クライアントから SSL でセッション確立後、1 レコードサーバ側から送信して終了するという単純なものである。以下に実験に使用したシーケンスを示す。

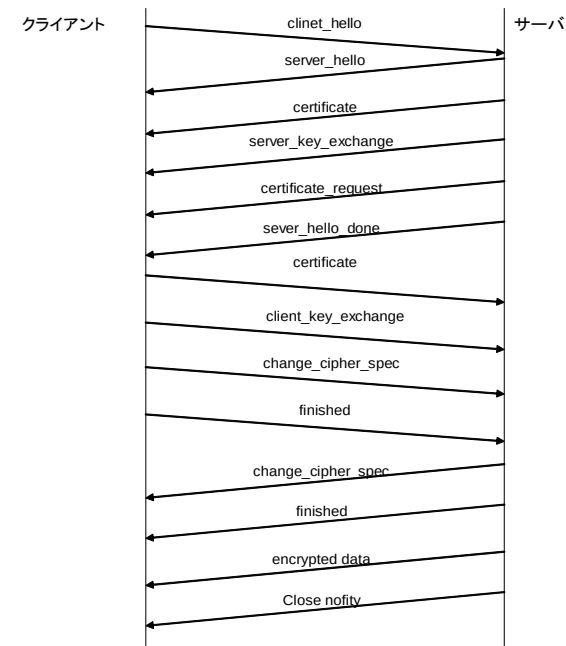


図 5. 正常系テストシーケンス

4.1.1 Internet explorer の場合の正常シーケンス

Internet Explorer の場合のシーケンスは以下のとおりであった。

SERVER	CLIENT
<---	ClientHello 01: log/server/20080515192010/R001_01
-ServerHello -->	02: log/server/20080515192010/S001_02
-Certificate -->	03: log/server/20080515192010/S002_03
-CertificateRequest	04: log/server/20080515192010/S003_04
-ServerHelloDone>	05: log/server/20080515192010/S004_05
<---	ClientHello 06: log/server/20080515192010/R002_06
-ServerHello -->	07: log/server/20080515192010/S005_07
-Certificate -->	08: log/server/20080515192010/S006_08
-CertificateRequest	09: log/server/20080515192010/S007_09
-ServerHelloDone>	10: log/server/20080515192010/S008_10
<---	Certificate 11: log/server/20080515192010/R003_11 **MULTI **
<---	ChangeCipherSpec 12: log/server/20080515192010/R004_12
<---	Finished 13: log/server/20080515192010/R005_13_DEC** ENCRYPTED **
-ChangeCipherSpec>	14: log/server/20080515192010/S009_14
-Finished -->	15: log/server/20080515192010/S010_15_DEC** ENCRYPTED **
<---	ApplicationData 16: log/server/20080515192010/R006_16_DEC** ENCRYPTED **
-ApplicationData>	17: log/server/20080515192010/S011_17_DEC** ENCRYPTED **
-AlertMessage -->	18: log/server/20080515192010/S012_18_DEC** ENCRYPTED **
close_notify	

図 6. Internet Explorer8のシーケンス

Internet Explorer の version 6,7,8 の各ブラウザについてテストしたが、特にシーケンス上で差異は、認められなかった。Internet Explorer では、最初の ClientHello が、常には送信され、一旦、ServerHelloDone の後に、再度 ClientHello が送信される。2 回目の ClientHello 以降のシーケンスは一般的な SSLv3 のシーケンスどおり続き、AlertMessage (close_notify) で終了する。Internet Explorer と接続する SSL サーバは、ClientHello から ServerHelloDone のシーケンスが 2 回繰り返されるというハンドリングを行わなければ

ならないことがわかる。

4.1.2 FireFix の場合の正常シーケンス

FireFox3 系の場合の正常シーケンスは以下のとおりである。

SERVER	CLIENT
<---	ClientHello 01: log/server/20080519114702/R001_01
-ServerHello --->	02: log/server/20080519114702/S001_02
-Certificate --->	03: log/server/20080519114702/S002_03
-ServerKeyExchange --->	04: log/server/20080519114702/S003_04
-CertificateRequest --->	05: log/server/20080519114702/S004_05
-ServerHelloDone --->	06: log/server/20080519114702/S005_06
<---	Certificate 07: log/server/20080519114702/R002_07 **MULTI **
<---	ChangeCipherSpec 08: log/server/20080519114702/R003_08
<---	Finished 09: log/server/20080519114702/R004_09_DEC** ENCRYPTED **
-ChangeCipherSpec --->	10: log/server/20080519114702/S006_10
-Finished --->	11: log/server/20080519114702/S007_11_DEC** ENCRYPTED **
<---	ApplicationData 12: log/server/20080519114702/R005_12_DEC** ENCRYPTED **
-ApplicationData --->	13: log/server/20080519114702/S008_13_DEC** ENCRYPTED **
-AlertMessage --->	14: log/server/20080519114702/S009_14_DEC** ENCRYPTED **
close_notify	

図 7. FireFox Ver3の正常シーケンス

FireFox3 では、通常の SSL のクライアント・サーバのシーケンスをハンドリングできればよいことがわかる。

4.2 異常シーケンス場合のテスト

異常系のテストでは、Alert プロトコルにより、状態を返してくることが想定される場合のケースを中心に幾つかのテスト項目をピックアップしテストを実施した。そして、テストツールを用いて、次の異常系のシーケンスを作成し、Internet Explorer, FireFox に適用した。

- ・シーケンス異常

- (1)ServerHello 送信を ServerCertificate にする
- (2)ServerCerfiticate 送信を ServerHelloDone にする
- (3)ServerHelloDone 送信を CangeCipherSpec にする
- (4)ChangeCipherSpec 送信を Finished にする
- (5)Finished 送信を ApplicationData にする

- ・パラメータの異常

- (1)ServerHello の server_version の値を存在しない 5 にする
- (2)ServerHello の gmt_time を 1 時間過去にする
- (3)ServerHello の random_bytes を 0 にする
- (4)ServerHello の CipherSuites の値の異常(0xff)

なお、異常系のシーケンステストであるが、通常、SSL/TLS の下の層は、TCP であるため、サーバの実装を間違わない限り上記のシーケンス異常のパターンはほとんど発生しない。しかし、RFC 上これらのシーケンスに対するエラー処理が明確に規定されていないため各ブラウザでの結果を把握しておくことは、組み込み機器の SSL サーバの実装上重要であると考えられるためである。なお、パラメータ異常の 4 項目については、(1)は、クライアントのリクエストに含まれる Version と異なる値がサーバから返された時の挙動、(2)は、サーバが組み込み機器であった場合時刻が適当でない可能性があり、通知された時刻が現在時刻よりずれていた場合の挙動、(3)は、組み込み機器で乱数を発生させる機能がない場合に 0 を入れた時の挙動、(4)は、通常使われない暗号スイートが指定された場合の挙動について調査するために項目を選択したものである。

4.2.1 シーケンス異常の場合

- ・シーケンス異常の場合の各ブラウザでのテスト結果を以下に示す。

	(1)	(2)	(3)	(4)	(5)
IE6,7,8	コネクション切断	コネクション切断	コネクション切断	コネクション切断	コネクション切断
FF3	Alert unexpected message	コネクション切断	Alert unexpected message	コネクション切断	正常終了

表 1. シーケンス異常の場合

シーケンス異常については、IE 系はすべてコネクションが切断される。これに対し、FireFox では、一部のエラーを Alert で返していることがわかる。なお、FireFox3 の(5)の正常終了は、動作的にはバグの可能性はある。

4.2.2 パラメータ異常の場合

- ・パラメータ異常の場合の各ブラウザでのテスト結果を以下に示す。

	(1)	(2)	(3)	(4)
IE6,7,8	コネクション切断	正常終了	正常終了	コネクション切断
FF3	Alert protocol version	正常終了	正常終了	Alert handshake failure

表 2. パラメータ異常の場合

パラメータ異常については、明らかにパラメータ異常で通信が継続できないと考えられる(1)(4)では IE 系はコネクション切断、FireFox では、一部のエラーを状況に応じた正しい Alert で返していることがわかる。

5. 考察

- ・テストセットについて

異常系のプロトコルをテストするためには、正常な応答が返らない前提でテストセットができていなければならない。このため、今回作成してテストセットでは、異

常系で応答が帰らない場合、指定された時間で処理を打ち切るためのタイムアウトの仕組みを入れ込むことでこの問題を解決している。

- ・ 各種 SSL の実装について

組み込み系の Web サーバをインプリメントする際、通信の相手システム(ここでは、Web ブラウザ)の通信シーケンスで独特の挙動をハンドリングできるようにすることは、不特定のシステムとの接続が必要であるシステムでは、必須のテスト項目である。このため、SSL のサーバ実装者は、まず最もメジャーな Web ブラウザである Internet Explorer の実装がどのようになっているかを知る必要がある。そこで、正常系のシーケンスにおいては、Internet Explorer は、ClientHello リクエストから ServerHelloDone までのシーケンスを 2 度繰り返していることがわかる。このため、Internet Explorer に接続する必要のある SSL の実装では、この繰り返しのハンドリングできる実装になっている必要がある。次に、異常系のシーケンスであるが、本稿で用いたメッセージの順番の間違いのシーケンスは、SSL/TLS の下位プロトコルが TCP であることから順序性とメッセージ内容は保障されているため通常は起こりえない。しかし、SSL/TLS に関する RFC や Netscape 社の規約は、正常時のシーケンスには、詳しく触れられているものの、異常時の振る舞いについては、どのような種類の Alert プロトコルメッセージがあるのか程度の記述しかなく、どういったシーケンスに陥ったときに、どのようなエラーを返したらよいかの記述がほとんどない。このため、異常となるシーケンスの場合は、各ブラウザ作成者のインプリメントマターとなっている。したがって、各ブラウザの終了の仕方はまちまちであるために、サーバの実装者はそれぞれのブラウザの挙動をハンドリングできるように実装する必要がある。まず、IE の場合の異常シーケンスでは、テストセットによる実験の結果、Alert メッセージを返さずにいきなりコネクションを切断することを前提とした、サーバの実装が必要である。また、FireFox をクライアントにする必要のあるサーバの実装では、結果が Alert 状態となることが普通であり、Alert 後、最終的にはコネクションを切断することを前提とした実装とする必要がある。FireFox を接続相手とするサーバの場合は、異常なシーケンスであった場合には、基本的に何がおかしかったかを Alert プロトコルで返すため、サーバは、コネクションが突然切断された際の理由を知るために有効なログを残すことが可能である。次にパラメータエラー異常の場合のテストについてであるが、(2)の ServerHello の `gmt_unix_time` で表される時刻を過去の時刻とするテストでは双方のブラウザとも正常終了しているがこれは、組み込み機器などの場合、タイマーがずれていることは十分考えられるため、多少の時刻のずれでは問題が発生しないことを示している。このためサーバの実装では、それほど現在時刻にシビアになる必要がないことを示している。また、(3)の ServerHello の `random_bytes` を 0 にするテストについては、各ブラウザとも正常終了しているが、これは、組み込み機器的には乱数生成は困難

な問題を含むので、セキュリティ的な問題は別とすれば、値が 0 となっても正常終了させるようになっているので、サーバの実装では、乱数生成の `random` の部分は、それほどシビアな値を入れる必要がないことを示していることが分かる。

6. まとめ

本稿では、組み込み機器実装の検証のための SSL/TLS テストセットを作成し、テストセットで実行可能なテストについて示した。そして、本プログラムの機能を用いて、いくつかの特徴的な正常・異常なシーケンスを含む SSL 通信行い、各種ブラウザの挙動について実行結果を示すことで、組み込み機器の SSL 実装に当たっての考慮すべき項目についての知見を得ることができた。近年、各種の組み込み機器でも SSL がサポートされるようになってきているが、規約のベースとなっている Netscape 社のドキュメントや RFC を見ても異常系についての記載が少ないため、本稿のテストセットを用いることで、実際の Web ブラウザ、および、本稿では触れなかったが、テストセットの SSL クライアント機能を活用し SSL サーバや SSL クライアントの実装を比較的簡単に確認することができることが期待できる。

7. 今後の課題

本稿では、作成したテストプログラムのうち主に SSL クライアント(Web ブラウザ)のテストを行うために、サーバ部分機能について、各種ブラウザでの実験結果とブラウザの挙動について示した。今後は、残りのクライアント部分について、各種 SSL サーバの実装の挙動を明らかにすること、および、脆弱性のテストセットについてのスクリプトを充実していけばよいものと考えている。

参考文献

- [1] T.Dierks 他, “The TLS Protocol Version 1.0”, ”RFC2246 , <http://www.ietf.org/rfc/rfc2246.txt>
- [2] T.Dierks 他, “The TLS Protocol Version 1.1”, ”RFC4346 , <http://www.ietf.org/rfc/rfc4346.txt>
- [3] T.Dierks 他, “The TLS Protocol Version 1.2”, ”RFC5246 , <http://www.ietf.org/rfc/rfc5246.txt>
- [4] E.Rescorla 著 斉藤孝道ら訳, 「マスタリング TPC/IP SSL/TLS 編」, オーム社, 2004
- [5] Codenomicon defensics, “Codenomicon TLS Server Test Tool”, <http://www.codenomicon.com/products/tls-server.shtml>