Vol.2014-CSEC-65 No.31
Vol.2014-IOT-25 No.31
2014/5/23

SSL/TLSサーバにおける Forward Secrecyへの対応状況について （＋速報版 Heartbleed Bug 発覚後の状況変化）

須賀 祐治[1,a)]

概要：米国家安全保障局（NSA）による通信傍受に関する一連の報道に伴い，その対策方法として注目される Forward Secrecy について取り上げる．昨年末 SNS などの主要サイトにおいて SSL/TLS サーバの Forward Secrecy 対応が立て続けに進められており，広く普及しつつある．本稿では SSLyze 0.8 を用いてクローリングを行った主要サーバの Forward Secrecy への対応状況について報告する．さらに日本時間 2014 年 4 月 8 日未明に公開された HeartBleed bug についても急遽取り上げた．脆弱性公開前後のサーバ設定の変化について 1 週間観測した結果を速報する．

キーワード：(Perfect) Forward Secrecy, SSLyze, HeartBleed bug

SSL/TLS servers status survey about enabling forward secrecy (+ Rapid survey after the Heartbleed bug)

Yuji Suga[1,a)]

Abstract: This paper discusses forward secrecy functionality, which has come into the spotlight due to a series of news reports regarding theNSA. SSL/TLS server support for forward secrecy was introduced at major sites such as SNS at the end of last year, so servers enabling forward secrecy are increasing. This paper shows SSL/TLS servers status survey about enabling forward secrecy by crawling with SSLyze 0.8. And this paper treats new Heartbleed bug of OpenSSL, released in the early hours of April 8, 2014 Japan time and reports preliminary results of observation for one week for changes in server configuration before and after publication of this bug.

Keywords: (Perfect) Forward Secrecy, SSLyze, HeartBleed bug

1. はじめに

本稿では (Perfect) Forward Secrecy について取り上げる．Forward Secrecy は Diffie-Hellman などの鍵交換プロトコルを利用する手順において，そのセッションでしか利用しない一時的な鍵ペアを生成します．もし仮に，この一時的な公開鍵に対応する秘密鍵が漏えいしても，暗号通信が解読される範囲を一部に限定する効果が注目されている．一方で，毎回同じ公開鍵を用いて暗号化されているケースでは，一旦暗号化されたデータが広域ネットワークを通して伝播されている場合，暗号通信が何十年という単位で長期間に渡って記録され続けていて，将来のいつか秘密鍵が漏えいすることにより，過去に遡って復号可能な状態におかれることになる．

Forward Secrecy が注目されるきっかけとなったのは米国家安全保障局（NSA）による通信傍受に関する一連の報道である．昨年 9 月，米国国立標準技術研究所（NIST）

[1] 株式会社インターネットイニシアティブ
Internet Initiative Japan Inc., Jinbocho Mitsui Bldg. 1-105 Kandajinbo-cho, Chiyoda-ku, Tokyo, 101-0051, Japan
[a)] suga@iij.ad.jp

が策定した暗号アルゴリズムの一部に NSA によるバックドアがあり，解読される可能性があるとの報道がなされている．NIST では，意図的に脆弱な暗号を採用した可能性を否定する声明を発表し，擬似乱数生成アルゴリズム Dual_EC_DRBG について，利用しないことを推奨する勧告が発表された [1]．これを受けて米 EMC 社は，デフォルトで Dual_EC_DRBG が使用される設定になっていた暗号ライブラリ RSA BSAFE の顧客に対して，本アルゴリズムを使用しないように伝えている [2]．Dual_EC_DRBG は，擬似乱数出力の 32 バイトを入手すれば，それ以後生成される乱数列を特定可能という脆弱性 [3] が 2007 年の時点で発表されていたにも関わらず，暗号アルゴリズムの利用現場では使い続けられていた．攻撃者が擬似乱数生成モジュールを管理下に置くことで，通信時に用いられるあらゆる擬似乱数やそこから生成された暗号化用鍵に関する情報などを，リアルタイムに窃取する問題が考えられる．実際，擬似乱数生成モジュールのエントロピーが小さいことに起因する問題は，例えば Debian の OpenSSL における脆弱性などが知られている [4], [5]．この脆弱性は，例えば Debian の特定バージョンにおける OpenSSL を使って鍵生成を行った場合，極端に少ない鍵空間からしか秘密鍵を導出していない問題である．この問題については 2008 年に指摘があったにも関わらず，現在でもその脆弱な鍵を利用しているサイトが存在している．また Android 上の一部の bitcoin アプリケーションにおいても，乱数生成時の問題が報告されている [6], [7]．bitcoin で用いられている ECDSA 署名では，署名演算を行うたびに，毎回ランダムな乱数パラメータを生成する必要がある．このとき異なる署名生成時に同じパラメータを使用した場合，秘密鍵が漏えいすることが知られている [8]．

暗号アルゴリズムにバックドアを仕掛ける方法以外にも，NSA による通信傍受の仕組みが明らかになっている．NSA が米 Verizon に対し電話の通話記録の収集を求めていたこと，PRISM と呼ばれるインターネット上の動画・写真・電子メールなどのデータを監視するプログラムが，米国の主要なインターネット関連企業の協力のもと運用されていたことが昨年 6 月に報道された．更に 10 月には，NSA によるリアルタイム通信傍受プロジェクトの存在が暴露され，Yahoo!と Google のデータセンター間の通信を盗聴していたことが公開された．また，E-mail プロバイダである Lavabit は，FBI から秘密鍵供託を指示されたことから，利用者のプライバシーを保護することができなくなったと判断し，自らサービスを停止している．このようにセキュアプロトコルを利用することで通信を暗号化していても，通信内容を NSA や他の組織に傍受されている可能性があることが，具体的な事例を通して認識されつつある．昨年 11 月に開催された IETF-88 においても，Pervasive Surveillance（広域監視）がメイントピックとして取り上げられている [9], [10], [11]．Forward Secrecy は，その対策方法の一つとして取り上げられるようになっている．

1.1 Forward Secrecy の概要

上述の問題は，ストレージの暗号化など，コンテンツセキュリティの分野においてはすでに認識されている．一旦暗号化したデータがそのまま第三者に公開された時点で，ブルートフォース攻撃（総当たり攻撃）の対象となるためである．つまり，暗号化データの公開時点で対象データの解読を行うことが可能な状況下に置かれ，攻撃者は AES などの共通鍵暗号方式においてすべての鍵で復号を試みることができる．一方で，署名については長期保存署名として規格化されており，タイムスタンプと併用してある時点の検証情報に更に署名していく形式により，長期間に渡り検証を可能にしている．しかし，データ暗号化については長期保存技術が確立していない．

一方で SSL/TLS などのセキュア通信プロトコルにおいて，クライアントとサーバで確立したセキュアチャネルを流れるデータの完全性は，長期に渡って保証する必要がないため，長期署名の概念は必要ない．しかし，機密性については上記に示したように，暗号通信が何十年という単位で長期間に渡って記録され続けることで破られる可能性があるため，それを保証する仕組みが必要になっている．セキュアデータストレージにおいては，遠い将来に復号するような状況が想定されるため，暗号化に用いた鍵を適切に管理する事が要求されている．一方で，セキュア通信プロトコルを用いて一時的に暗号化する場合には，暗号化に用いた鍵を保存しておく必要はない点からも暗号鍵管理の観点においては大きく異なる．Forward Secrecy のアイデアは，この特徴の違いである「一時的に用いた鍵は捨ててもよい」ということを利用した方式である．

一般的なセキュア通信プロトコルやセキュアデータストレージでは，共通鍵暗号と公開鍵暗号の両方を用いるハイブリッド方式が使われている．実データの暗号化には共通鍵暗号が利用され，コンテンツ鍵（データ暗号化に用いられる鍵）は公開鍵暗号で暗号化されることが一般的である．例えば，AES 暗号で用いたコンテンツ鍵を，RSA 公開鍵で暗号化を行うという手順がその一例である．SSL/TLS では，鍵交換アルゴリズムとして RSA，DH_RSA，DHE_RSA などが定められている．鍵交換アルゴリズムは，コンテンツ鍵や MAC 用鍵を導出するための元データを安全に共有するために用いられる．例えば，鍵交換アルゴリズムとして RSA を選択した場合には，まず PreMasterSecret を安全に共有する．PreMasterSecret はクライアントが作成するランダムデータで，サーバ証明書に格納されている公開鍵を用いて暗号化することで，安全にサーバと共有することができる．この場合，RSA 暗号をサーバ認証と鍵交換の両方に利用していることを意味する．また DH_RSA では，

証明書に含まれている DH 公開鍵を用いて DH 鍵交換アルゴリズムにより安全なデータ共有を行う．

一方で DHE_RSA では，その都度異なる一時的な DH 公開鍵・秘密鍵の生成を行う．ここで，サーバ証明書に記載の公開鍵に対応する秘密鍵が後日漏えいしたケースを考える．鍵交換アルゴリズムが RSA または DH_RSA の場合，固定された公開鍵を用いて鍵交換を行っていたため，暗号化された通信内容を長期的に盗聴されていた場合，その記録からコンテンツ鍵の導出が可能であり，暗号通信データから当時の通信内容が暴露してしまう問題がある．一方で DHE_RSA では，セッションごとに DH 公開鍵・秘密鍵を生成しその都度捨ててしまうため，暗号通信データを復号するにはその通信に利用されていた DH 秘密鍵の解読が必要となる．もし仮に DH 秘密鍵の解読がされたとしても，対応する DH 公開鍵を用いて安全に導出したコンテンツ鍵で暗号化された通信のみが暴露されるのみで，被害を限定的にすることができる．図-1 は暗号化用鍵を導出するために利用された公開鍵の対応付けを表現している．これは公開鍵に対応する秘密鍵が解読された際に暴露される暗号化用鍵の対応付けについて表現しているとも言える．

DHE_RSA の利用は，使い捨ての公開鍵をその都度作成して暗号通信を行っていることを意味する．このとき，一時的に生成された DH 公開鍵はサーバによる RSA 署名により保証されており，DH 鍵交換アルゴリズムが潜在的に持つ中間者攻撃への弱さを防ぐことができる．TLS においては，1999 年に策定された RFC2246（TLS1.0）[12] から，Forward Secrecy が適用可能な CipherSuite が定められており，近年では DH の楕円曲線版である ECDH も利用されている．ECDH は 2006 年に策定された RFC4492 [13] より利用可能となっており，Forward Secrecy 対応の鍵交換アルゴリズムとしては ECDHE_*として記述されている．一般的には DH よりも ECDH が高速処理できるとされており，様々なクライアントやサーバにおいて ECDH の利用が広がっている．

1.2 Forward Secrecy の適用事例と注意点

2011 年から対応している Google [14] をはじめとして，2013 年には Facebook [15]，Twitter [16]，GitHub [17] などが ForwardSecrecy 対応または対応中であることを表明している．主要サイトの対応状況については EFF（Electronic Frontier Foundation）により随時更新されている [18]．これらの対応に呼応する形で，技術者向けに Apache+SSL などでの具体的な設定方法例も紹介されている [19]．しかし，Forward Secrecy を適用しても完全に Pervasive Surveillance を防ぐことはできないと認識されている．ひとつは，(EC)DH 鍵交換アルゴリズムについての潜在的問題である．サーバへの侵入などを通して RSA 秘密鍵が漏えいした，または RSA アルゴリズムが危殆化し

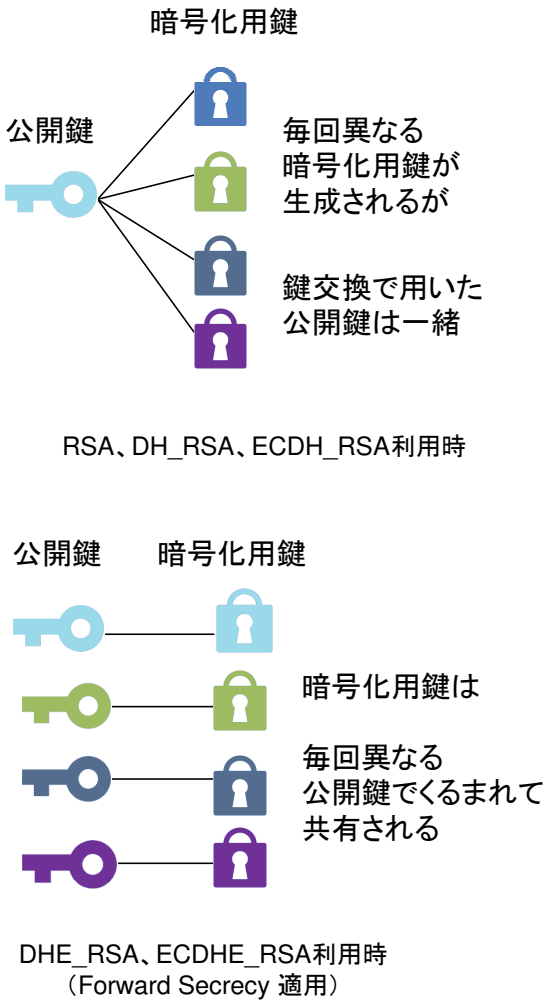

図 1 TLS における Forward Secrecy 不適用・適用の違い

た時点で，Forward Secrecy に対応し使い捨て公開鍵を利用していたとしても，それ以降の暗号通信が漏えいする可能性がある．例えば TLS における鍵交換アルゴリズムとして DHE_RSA を利用する場合を考えると，クライアントはサーバから送られてきた一時的な DH 公開鍵の正当性を確認するが，クライアントとサーバに割り込んだ攻撃者は RSA 秘密鍵を入手しているため，サーバから送信する DH 公開鍵を書き換えることができ，中間者攻撃が可能となる．そのため RSA 鍵が危殆化した場合，それ以降の暗号通信は解読可能となる．この場合，RSA 鍵ペアとサーバ証明書の更新により対策する必要がある．

2 点目は擬似乱数生成モジュールの問題である．乱数生成モジュールは，多くの場合，暗号ライブラリを通して利用されており，その暗号ライブラリの認定制度としては，CMVP, JCMVP があるため，これらを活用することで安全に利用できると考えられている．しかし，前述した RSA

BSAFE は政府機関のお墨付きを得ている暗号ライブラリだったにも関わらず，問題があったことが露呈している．この事例は，暗号ライブラリとして安全が確認されていても，擬似乱数生成モジュールに入力する Seed や Nonce が短いなど，利用上の問題が発生していないかについて気を付ける必要があることを示唆している．TLS を利用する前にサーバは RSA 鍵ペアを生成しサーバ証明書を作成するが，その際にはサーバが擬似乱数生成モジュールを用いて素数を生成する必要がある．ここで前述した Debian OpenSSL 問題のように十分なエントロピーを確保しなければいけない．TLS プロトコルはクライアントとサーバそれぞれで Hello を送信することから始まる．ClientHello および ServerHello のメッセージ送信においてそれぞれ 28 バイトのランダムデータを含む random を生成する．さらに鍵交換アルゴリズムとして RSA を選択した場合には，46 バイトのランダムデータを含む PreMasterSecret を ClientKeyExchange メッセージを通して安全に共有する．また DHE や ECDHE を利用する場合には一時的な DH 鍵または ECDH 鍵を生成して鍵交換アルゴリズムを実行し PreMasterSecret を共有する．このとき一時的な（EC）DH 秘密鍵の選択範囲が少ないと，解読される危険性が増すことが知られている．最後に ClientHello.random，ServerHello.random，PreMasterSecret の 3 つのランダムデータから MasterSecret を算出し，MasterSecret から，クライアントとサーバそれぞれの MAC 鍵，コンテンツ鍵，（CBC 暗号モード利用時の）初期ベクトル（IV）を導出することで安全な通信を行うことができる．TLS では複数のランダムデータを利用して暗号化用鍵を導出する構造を持つ設計がなされているが，安全に擬似乱数生成ができない，つまり生成される値に偏りがあるケースにおいては TLS が安全に利用できなくなる可能性があることに注意せねばならない．

2. 観測環境とその結果

地方自治体および大学のサイトについて上記脆弱性の対策状況についてクローリングすることで SSL/TLS の設定状況を把握する先行研究がある [20]．これらの研究では，THC-SSL-DOS 対策, RFC5746 対策, 証明書の受け入れ対策, RSA 鍵長対策, CRIME 攻撃対策について調査対象となっていたが，実際に利用されるアルゴリズムについての調査は対象となっていなかった．

今回の調査対象は以下の通りである．今回クローリングに際し利用したソースはすべて Alexa [21] 提供のリストから抽出したものである．なおクローリングは 2014 年 4 月 14 日に行われたものを集計している．

- Alexa top sites の上位 20000 サイト
- .jp ドメイン 17988 サイト
- ある国の政府機関ドメイン 152 サイト
- .gov ドメイン 809 サイト

ここで SSL/TLS 接続が確立したとしても共用サーバの利用など意図せず SSL を有効にしているケースが見受けられるため，サーバ証明書の FQDN マッチングが OK なもののみを取り上げた． これは通常のブラウザにおいてエラーを起こさないように設定されており，実際に SSL/TLS が利用されていると考えられるサーバのみを調査対象とすることで，より現実的な状況把握を行うことを目指した．公開鍵証明書の大規模収集という観点では，EFF SSL Observatory[22] や PsQs [23],RwWr[24] などの調査が存在する．このクローリング方式においては IP アドレスベースの調査のためテストサイトなど実際に利用されていない証明書を収集してしまうデメリットがある．実際 Heninger らの調査 [23] においては 60%以上のサイトがほかのサイトと秘密鍵ペアを意図せず共有しているという調査結果が報告されており，これは実際に正しく運用されていないサイトをカウントしている点や，同じ FQDN に対して複数の IP アドレスが割り振られている点などの事情をうまく汲み取れていないと考えられる．

結果として本稿では以下の SSL/TLS サーバ (重複あり) について調査を行っている．

- Alexa top sites [A20K] 6835 サイト
- .jp ドメイン [.jp] 5668 サイト
- ある国の政府機関ドメイン [某] 78 サイト
- .gov ドメイン [.gov] 307 サイト

それぞれのサーバ群に対して Forward Secrecy への対応状況を調査している．付随的に SSL/TLS バージョン対応状況および脆弱なアルゴリズムを受け入れているか，ついても報告を行う．

2.1 SSL/TLS バージョン対応状況

（総数）	A20K 6835	.jp 5668	某国 78	.gov 307
SSL2.0	378	1380	6	8
SSL3.0	6516	5672	77	290
TLS1.0	6574	5669	76	302
TLS1.1	3536	834	17	119
TLS1.2	3771	965	27	125

表 1 SSL/TLS バージョン対応状況

SSL3.0 と TLS1.0 サポートがどのカテゴリにおいてもほとんど対応していることが分かる．脆弱であると認識されている SSL2.0 も特に .jp ドメインで受け入れていることも読み取れる．また， Alexa top sites の上位 20000 サイトでは半数以上が TLS1.1/1.2 への対応がなされているが，.jp ドメインにおいては対応が遅れていることが分かる．

2.2 脆弱なアルゴリズムの受け入れ状況

（総数）	A20K 6835	.jp 5668	某国 78	.gov 307
輸出規制用暗号の利用	1120	2028	11	78
脆弱な共通鍵暗号の利用	1302	3248	22	81

表 2 脆弱なアルゴリズムの受け入れ状況

DES(鍵長 56 ビット) 以下のアルゴリズムを含む CipherSuites を受け入れるサーバを「脆弱な共通鍵暗号の利用」としてカウントした．そのうち，暗号化鍵を 40 ビットに制限されている CipherSuites を受け入れるサーバを「輸出規制用暗号の利用」としてカウントしている．驚くべきことに.jp ドメインにおいては 56 ビット以下の暗号アルゴリズムを 57%のサーバが受理しており，しかも輸出規制時代に利用されていた 40 ビット暗号の利用を 35%のサーバが現在も受け入れている状況が明らかになった．

2.3 Forward Secrecy 対応状況

（総数）	A20K 6835	.jp 5668	某国 78	.gov 307
EXP-EDH-RSA-DES-CBC-SHA	365	1644	10	11
EDH-RSA-DES-CBC-SHA	640	2803	19	13
DHE-RSA-AES256-SHA	2515	3879	40	72
DHE-RSA-CAMELLIA256-SHA	1394	1254	19	29
DHE-RSA-AES128-SHA256	922	557	11	17
DHE-RSA-AES128-SHA	2614	3906	41	72
DHE-RSA-CAMELLIA128-SHA	1391	1254	19	29
DHE-RSA-SEED-SHA	659	979	14	23
EDH-RSA-DES-CBC3-SHA	2413	3897	41	70
DHE-RSA-AES256-GCM-SHA384	923	556	11	17
DHE-RSA-AES128-GCM-SHA256	921	555	11	17
ECDHE-RSA-AES256-SHA	2081	373	6	89
ECDHE-RSA-AES128-SHA	2079	374	6	90
ECDHE-RSA-AES256-GCM-SHA384	1573	105	3	7
ECDHE-RSA-AES128-GCM-SHA256	1614	109	3	7
ECDHE-RSA-DES-CBC3-SHA	1686	126	3	22
ECDHE-RSA-RC4-SHA	1448	117	1	7

表 3 SSL/TLS バージョン対応状況

前節でも指摘したように Forward Secrecy 対応の CipherSuites の中でも輸出規制用に準備されていた EXP-EDH-RSA-DES-CBC-SHA，EDH-RSA-DES-CBC-SHA はまだ多くのサーバで受理されていることが分かる．また.jp ドメインでは依然として DHE-*が利用されているが，TLS バージョンの移行と同じように Alexa top sites の上位 20000 サイトでは ECDHE-* への移行が進められていることが分かる．

3. Heartbleed Bug

日本時間 2014 年 4 月 8 日未明に公開された HeartBleed bug [25] について急遽取り上げる．この脆弱性は OpenSSL 1.0.1 から 1.0.1f および 1.0.2beta1 において Heartbeat メッセージの処理において境界チェックの問題があり，OpenSSL が動作しているマシンのメモリ情報を取得可能な状態にあったことが公表されている．RFC6520 で策定されている Heartbeat プロトコルはリクエスト-レスポンス型の 2-way で完結する簡易なプロトコルで SSL/TLS の Record レイヤ，つまりハンドシェイクと同じレイヤで送受信される．SSL/TLS のハンドシェイク中にいきなり Heartbeat リクエストが送られてしまうと，それを受け取った OpenSSL ライブラリは境界チェックのバグを誘発しメモリ領域のデータをレスポンスしてしまう．このとき OpenSSL では受け取った Heartbeat リクエストと同じ長さのレスポンスを返すように実装されており，Heartbeat プロトコルの仕様では Length 情報として 2 バイトが割り当てられているため最大で 0xFFFF バイト分のメモリ情報が漏洩することが確認されている．

既に OpenSSL 1.0.1g でパッチが公開されており 1.0.1g にアップデートする，もしくは Heartbeat を無効にするコンパイルフラグをつけて再インストールを行うことで問題が解消される．パッチが与えられているため，他の脆弱性と同様に最新版に保つ対策のみでよいと考えがちであるが，本脆弱性により他ユーザの ID/Password や Cookie が見られてしまう，サーバ証明書の秘密鍵が抜き取られてしまうなどの被害が実験で指摘されている．1.0.1 が公開されてから 2 年が経過していること，メモリ情報を搾取された際にログが残らないことから，早急な対応が必要であると認識されていた．特にサーバの秘密鍵が漏れている可能性があり，その情報を悪用して本物と判断される偽サーバの構築や，Forward Secrecy を持つ CipherSuites を使っていない場合には過去に記録された暗号通信が復号されてしまう．そのため複数の認証機関では秘密鍵を作り直しサーバ証明書を再発行することが推奨されていた．

3.1 証明書の再発行状況

2014 年 3 月 28 日から 30 日に収集された .jp ドメインの 17988 サイト，4 月 2 日から 6 日にかけて収集された Alexa top sites の上位 20000 サイトの SSL/TLS 接続情報を用いて証明書再発行の状況を調査した．

OpenSSL Security Advisory [26] が公開された 4 日 8 日朝 4 時半頃の 36 時間後から 24 時間ごとに上記サーバの証明書を取得して，HeartBleed bug の公開前の証明書との比較を行った．以下の表では，4 日 8 日朝 4 時半よりも後に発行されたと考えられる証明書の数を列挙していく．括弧内の数字は，そのうち有効期限に変化が無いが異なる証明書に差し替えられた数を示している．これも今回の問題への対応として証明書を差し換えた場合に相当すると考えられるためカウントしており，他の観測情報とは異なる結果となっていることに注意する．

脆弱性公開の 36 時間後にはすでに多くのサイトで秘密鍵漏洩のリスクが理解され証明書の再発行が行われていることが分かる． 特に Alexa top sites の上位に位置するサーバ群ではその対処の迅速さが目立ち，36 時間後には

（総数）	A20K 6835	.jp 5668
脆弱性公開 36 時間後	501 (163)	108 (43)
脆弱性公開 60 時間後	695 (192)	156 (53)
脆弱性公開 84 時間後	844 (215)	177 (55)
脆弱性公開 108 時間後	879 (221)	182 (55)
脆弱性公開 132 時間後	932 (224)	220 (55)

表 4 証明書の再発行状況

7.3%，60 時間後には 10.1%，84 時間後には 12.3% のサーバで安全な状態に改善された．なお，証明書の差し替えが行われたサイトに本脆弱性の問題があったかどうかについては不正アクセス禁止法などに抵触すると考えられるため調査を行っていない．

4. まとめ

Alexa top sites を用いて SSL/TLS サーバのクローリングを行い Forward Secrecy への対応状況について調査した結果を報告した．現時点では RSA アルゴリズムでは 2048 ビットの利用が主流になりつつあるが DHE-* CipherSuites ではいまだに DH1024 が利用されている．本稿には掲載できなかったが，発表時には DH アルゴリズム鍵長についての考察も報告できる見込みである．

さらに Heartbleed Bug での観測情報についても報告を行った．TLS1.1/1.2 のサポートを行うなど，常に最新のライブラリを利用する「意識の高い」サーバが脆弱な状態に置かれていたという皮肉な事例であったとも言える．一方でエンバグしたライブラリがあったことが，サーバ管理者の意識を高め，安全なバージョン，アルゴリズムへの移行が進んだとも言える．おそらく後世にも残る本事例について，いまだ決着がついていない状況であるが，本稿を目にすることで OpenSSL のバージョンアップのきっかけとなれば幸いである．

参考文献

[1] NIST, "SUPPLEMENTAL ITL BULLETIN FOR SEPTEMBER 2013", http://csrc.nist.gov/publications/nistbul/itlbul2013_09_supplemental.pdf
[2] ArsTechica, "Stop using NSA-influenced code in our products, RSA tells customers", http://arstechnica.com/security/2013/09/stop-using-nsa-influencecode-in-our-product-rsa-tells-customers/?comments=1&post=25330407#comment-25330407
[3] Dan Shumow, Niels Ferguson, "On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng", Rump session in CRYPTO2007, http://rump2007.cr.yp.to/15-shumow.pdf
[4] Debian Security Advisory, "DSA-1571-1 openssl – predictable random number generator", http://www.debian.org/security/2008/dsa-1571
[5] IIJ, IIR Vol.17, "1.4.1 SSL/TLS, SSH で利用されている公開鍵の多くが他のサイトと秘密鍵を共有している問題, http://www.iij.ad.jp/development/iir/pdf/iir_vol17.pdf
[6] bitcoin.org, "Android Security Vulnerability", http://bitcoin.org/en/alert/2013-08-11-android
[7] Joppe W. Bos, J. Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, Eric Wustrow, "Elliptic Curve Cryptography in Practice", https://eprint.iacr.org/2013/734
[8] 須賀, "Bitcoin の ECDSA 署名生成時にポカしたら現金搾取される", 4A1-3, SCIS2014.
[9] IETF Blog, "We Will Strengthen the Internet", http://www.ietf.org/blog/2013/11/we-will-strengthen-the-internet/
[10] CA Security COUNCIL Blog, "IETF 88 - Pervasive Surveillance, https://casecurity.org/2013/11/26/ietf-88-pervasive-surveillance/
[11] IETF 88 Technical Plenary: Hardening The Internet, https://www.youtube.com/watch?v=oV71hhEpQ20
[12] The TLS Protocol Version 1.0http://www.ietf.org/rfc/rfc2246.txt
[13] Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security(TLS), http://www.ietf.org/rfc/rfc4492.txt
[14] Google Online Security Blog, "Protecting data for the long term with forward secrecy", http://googleonlinesecurity.blogspot.jp/2011/11/protecting-data-for-longterm-with.html
[15] Facebook Engineering, "Secure browsing by default", https://www.facebook.com/notes/facebook-engineering/secure-browsing-by-default/10151590414803920
[16] The Twitter Engineering Blog, "Forward Secrecy at Twitter", https://blog.twitter.com/2013/forward-secrecy-at-twitter-0
[17] The GitHub Blog, "Introducing Forward Secrecy and Authenticated Encryption Ciphers", https://github.com/blog/1727-introducing-forward-secrecy-andauthenticated-encryption-ciphers
[18] Electronic Frontier Foundation, "UPDATE: Encrypt the Web Report: Who's Doing What", https://www.eff.org/deeplinks/2013/11/encrypt-web-reportwhos-doing-what#crypto-chart
[19] Qualys Community, "Configuring Apache, Nginx, and OpenSSL for Forward Secrecy", https://community.qualys.com/blogs/securitylabs/2013/08/05/configuringapache-nginx-and-openssl-for-forward-secrecy
[20] Yuji Suga, SSL/TLS status survey in Japan - transitioning against the renegotiation vulnerability and short RSA key length problem, The 7th Asia Joint Conference on Information Security (AsiaJCIS 2012).
[21] http://www.alexa.com/topsites
[22] Electronic Frontier Foundation, The EFF SSL Observatory, https://www.eff.org/observatory
[23] Nadia Heninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", USENIX Security'12.
[24] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, Christophe Wachter "Public Keys", CRYPTO2012.
[25] The Heartbleed Bug, http://heartbleed.com/
[26] OpenSSL Security Advisory [07 Apr 2014], "TLS heartbeat read overrun (CVE-2014-0160)", https://www.openssl.org/news/secadv_20140407.txt