1U-6

SSL サーバの運用方式を反映したパフォーマンス計測ツールの実装

野出 利緒† 初谷 良輔†† 齋藤 孝道†
† 明治大学 †† 明治大学 大学院

1 はじめに

セキュリティプロトコルである SSL (Secure Socket Layer) やその後継にあたる TLS (Transport Layer Security) (以降, 併せて SSL) が広く利用されている [1, 2, 3]. しかしながら, SSL 通信は, 処理コストの高い多数の暗号処理を伴い, それを導入した Web サーバ (以降, SSL-Web) に SSL 通信が集中すると, サーバのパフォーマンスが急激に低下することがある. このような問題を事前に予測するために, SSL-Web サーバのパフォーマンス計測を行う必要がある. SSL 通信の負荷生成および計測を行う既存のツールには, httperf [4], JMeter [5], OpenSSL の s_time コマンド [6, 7] があるが, これらのツールでは SSL が提供する多様な認証モード, 鍵交換方式, バルク暗号方式などを全てサポートしていない.

そこで, 本論文では, 多様な SSL-Web サーバの運用形態を考慮した SSL のパフォーマンス計測ツールの提案と実装について示す.

2 提案ツール

2.1 概要と実装

提案ツールは, SSL が提供する多様な認証モード, 鍵交換アルゴリズム, バルク暗号アルゴリズム, 及び, SSL セッションの確立方法をサポートする. また, httperf や OpenSSL の s_time コマンドでは, パフォーマンス計測中に生成する HTTPS リクエストが一定であり, 本来, Web ブラウザ (ユーザ) から SSL-Web サーバに対して連続して送信される HTTPS リクエストを再現することができない. そこで, 提案ツールでは, Web ブラウザから Web サーバに対して送信される実際のリクエストを事前に記録し, これをシナリオとして負荷生成時に利用することで, SSL-Web サーバのパフォーマンス計測を実現することもできる.

提案ツールの実装は, Fedora Core 4(kernel-2.6-11) 上に gcc-4.0.0 (C 言語) と SSL 用の API として OpenSSL-0.9.8b を利用して実現した.

2.2 提案ツールの詳細

2.2.1 SSL の機能

提案ツール上では, SSL の認証モードとして, サーバ認証モード, クライアント認証モード*, 匿名認証モードに対応している. また, 提案ツールにおける SSL セッションの確立方法は, SSL セッションを新規に確立する Full Handshake と, 過去に生成した SSL セッションをもとに SSL セッションを再確立する Session Resumption の 2 つに対応している. Session Resumption を利用することで, 暗号処理の一部を省略して, SSL セッションを確立できるため, Full Handshake に比べてその処理コストは低い.

これら動作モードの切り替えは, テキストベースの設定ファイルで行う.

2.2.2 シナリオ生成

提案ツールでは, Web ブラウザが SSL-Web サーバに送信した HTTP リクエストをテキストファイル (以降, ScenarioFile) として保存し, これをシナリオとして実際のパフォーマンス計測時に利用することができる. これにより, Web ブラウザ (ユーザ) の挙動を考慮した負荷生成を実現する. 図 1 に, この機能の処理手順を示し, 以下で, その詳細を示す. ただし, 説明の手順は, 図中に割り当てた番号と対応している:

(1) Web ブラウザおよび実装した提案ツールの一部であるシナリオ生成プログラム (SCG: SCenario Generator) を Web ブラウザと同じマシン上で起動する. 次に, Web ブラウザから SSL-Web サーバにアクセスし, Web ページを取得する. このとき, Web ブラウザから発行された HTTP リクエストは, SCG を経由し, SSL-Web サーバに転送する. その際, この HTTP リクエストを SCG は, ScenarioFile として保存する.

(2) (1) にて作成した ScenarioFile を, 提案ツールの設定ファイルにインポートする. ただし, 複数の ScenarioFile をインポートすることもできる.

(3) インポートした ScenarioFile に保存された HTTP リクエストをもとに, 実際に HTTPS の負荷生成をし, その際のパフォーマンス計測を行う.

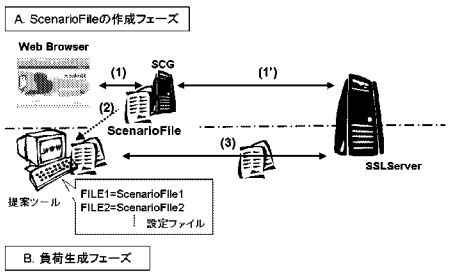

図 1: シナリオ生成と負荷生成の処理手順

2.3 擬似クライアントの生成方法

提案ツールでは, スレッドを用いて, 仮想的に複数のクライアントを生成する. ただし, 今回は, 同時に生成可能なスレッド数の上限を 100 としている. 100 以上の HTTPS リクエストを発行する場合は, 各スレッドが複数回処理を行う.

3 パフォーマンス計測

ここでは, 提案ツールを利用して, 既存の SSL-Web サーバのパフォーマンス計測を行う.

3.1 実験環境

図 2 にパフォーマンス計測に利用したシステム構成を示す. SSLClient は, 提案ツールが稼動するホストであり, サーバ側のネットワークに存在する SSLServer は, SSL-Web サーバとして Apache/mod_ssl が稼動するホストである. Load Balancer (負荷分散装置) は, SSLServer をスケールアウトした際のパフォーマンス計測を行うために用いる. また, 各ホストは互いに 1000 Mbps のレイヤ 2 スイッチで接続されている:

SSLClient
CPU: Pentium 4 (3.00 GHz)
RAM: 1024 MByte
OS: Kernel-2.6-11 (Fedora Core 4)

Load Balancer
Appliance: Array Networks TMX3000 [9]

† Rio NODE (ee37016@isc.meiji.ac.jp)
†† Ryosuke HATSUGAI (hatsugai@cs.meiji.ac.jp)
† Takamichi SAITO (saito@cs.meiji.ac.jp)
Meiji University(†)
Graduate School of Meiji University(††)
1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa, 214-8571, Japan.(†)(††)

* 相互認証モードとも呼ばれる.

SSLServer
CPU: XEON (3.20 GHz)
RAM: 1024 MByte
OS: Kernel-2.4-20 (RedHat Linux 9)
SSL Server: Apache-2.0.54/mod_ssl

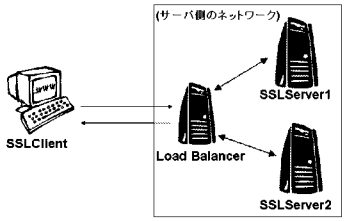

図 2: 計測実験に用いたシステムの構成

3.2 計測結果

パフォーマンス計測として, 実験 A と実験 B の 2 種類の実験を行った.

3.2.1 実験 A

この実験は, サーバ認証モードとクライアント認証モードのパフォーマンスを比較するための実験である. ここでは, サーバ認証モード, またはクライアント認証モードで稼動する SSLServer に対して, 新規の SSL コネクションを確立し, その後, 5Kbyte の html ファイルを取得し終えるまでの処理時間を計測する. リクエストする SSL コネクションの数は, 1000 から 5000 まで変動させ, また, 上述のとおり, SSLServer は, 1 台の場合と 2 台を並列稼動させ, それぞれの場合について計測する. ただし, 今回の実験で利用するクライアントの RSA 秘密鍵は, パスワードによる暗号化を施していないため, その秘密鍵を利用するために, 計測中にパスワードを入力する必要はない.

鍵交換アルゴリズムとして RSA (1024 bit), および, バルク暗号アルゴリズムとして 3DES (CBC モード), ハッシュアルゴリズムとして SHA1 を利用した場合のパフォーマンス結果を 図 3 に, RSA, RC4, SHA1 を利用した場合の結果を図 4 に示す. 図中の凡例にある *S*erverAuth はサーバ認証モードを, *C*lientAuth は, クライアント認証モードの利用を示しており, また, *1 と*2 は, SSLServer の台数を示している.

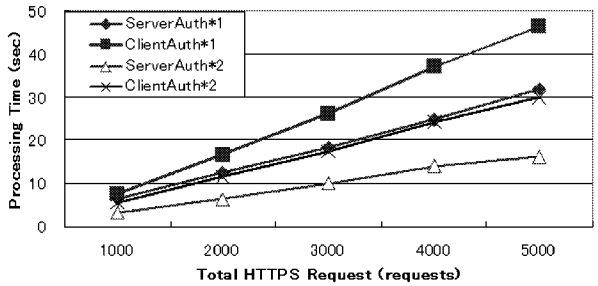

図 3: サーバ認証モードとクライアント認証モードの比較 (DES-CBC3-SHA1)

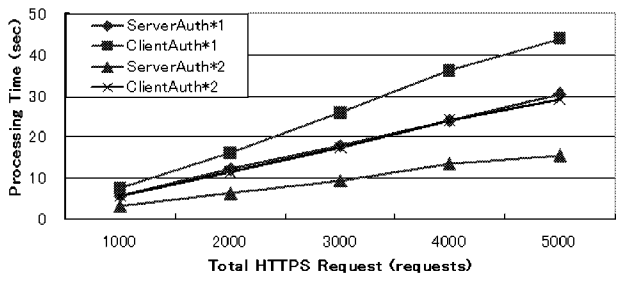

図 4: サーバ認証モードとクライアント認証モードの比較 (RC4-SHA1)

図 3 と図 4 のとおり, SSLServer の台数が 1 台で SSL コネクション数が 5000 の場合で, 処理時間は, サーバ認証モードを利用した場合, 約 31 秒, 同様にクライアント認証モードは, 46 秒の処理時間が掛かった. SSLServer を 2 台とした場合では, サーバ認証モードでは約 16 秒, クライアント認証モードでは, 約 29 秒となった. SSLServer が 1 台の時と, 2 台の時両方で, クライアント認証モードよりサーバ認証モードの方が 10 秒以上短い時間で処理が終了する. これにより, サーバ認証モードに比べて, クライアント認証モードの利用は, サーバにより高い負荷をかけることがわかる.

3.2.2 実験 B

この実験 B は, Session Resumption の効果を確認するための実験である. 計測は, SSL コネクション数を 1800 から 4200 まで変動させ, その際に 5Kbyte の html ファイルを取得するのに掛かった処理時間とする. 鍵交換アルゴリズムとして RSA, バルク暗号アルゴリズムとして 3DES(CBC モード), ハッシュアルゴリズムとして SHA1 を利用した. この実験 B では, Session Resumption の効果を確認するため, SSL-Web サーバと確立する SSL コネクション数の 2/3 は, Sessinon Resumption によるものである. 実験 B の結果を図 5 に示す. 図中の実線は Session Resumption を利用していない場合, 破線は Session Resumption を利用した場合の計測結果を示している.

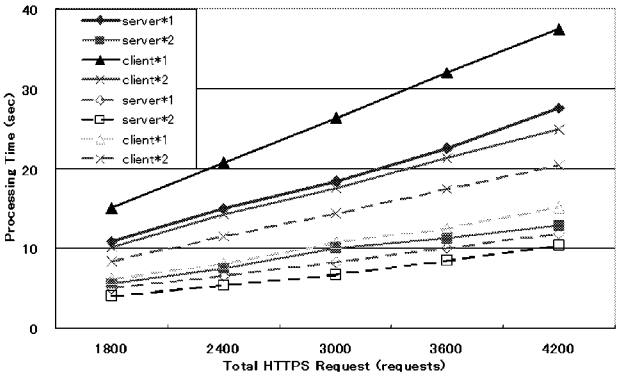

図 5: Session Resumption の有効性検証

図 5 より SSL コネクション数が 4200 の場合, SSL-Web サーバが 1 台のときにはサーバ認証モードとクライアント認証モードの両方で, 処理時間が 15 秒以上短縮されたことがわかる. これにより, Session Resumption の利用がサーバのパフォーマンス向上につながることがわかる.

4 まとめ

本論文では, SSL-Web サーバの運用形態を考慮したパフォーマンス計測ツールの提案と実装を示した. また, それを利用して Apache/mod_ssl サーバの評価を行った. 今後の課題として, シナリオ生成機能を利用したパフォーマンス計測がある. また, 計測結果の GUI 化など, より詳細な出力値を得ることができるようにすること, さらに, より大規模な SSL-Web システムのパフォーマンス計測を行うために, 提案ツールのクラスタ化などがある.

参考文献

[1] Alan O. Philip Locher, and Paul C. Kaltorn, "The SSL Procotol Version 3.0 draft"
[2] Tim Dierks and Christopher Allen, "RFC2246: The TLS Protocol Version 1.0"
[3] Eric Rescorla 著, 齋藤孝道, 鬼頭利之, 古森貞 監訳, マスタリング TCP/IP SSL/TLS 編
[4] http://www.hpl.hp.com/reserch/linux/httperf/
[5] http://jakarta.apache.org/
[6] http://www.openssl.org/
[7] John Viege ,Matt Messier ,Pravir Chandra 共著, 齋藤孝道 監訳, OpenSSL
[8] http://www.apache.jp/
[9] ArrayNetworks, http://www.arraynetworks.net/