Vol.2014-DPS-158 No.19
Vol.2014-CSEC-64 No.19
2014/3/6

SaaR：Sandbox as a Request の実装と評価

可児潤也[†1]　加藤岳久[†2]　間形文彦[†3]　勅使河原可海[†4]
佐々木良一[†4]　西垣正勝[†1]

近年，不正者によるサーバ攻撃が多発しており，我々はサーバ攻撃への対策として，SaaR（Sandbox as a Request）というコンセプトを提案した．本提案方式は，各クライアントからのサーバに対するリクエストごとに仮想サーバ（仮想マシンによって実装されるサーバ）をワンタイムで提供する方式となっている．正規のクライアントからのアクセスに対しても，不正なクライアントからのアクセスに対しても，サーバの「複製」がその都度サーバ内のサンドボックスの中に生成され，複製サーバのサービスがクライアントに提供される．複製された仮想サーバはクライアントからのリクエストに応じたサービスを終えた時点で使い捨てられる．もし不正者がサーバの脆弱性をつきサーバ内のデータの改ざんに成功したとしても，それは不正者に一時的に提供されたサーバの複製であり，サーバ本体は無傷を保つことになる．本稿では，Web サーバの形態で SaaR を実装し，実現可能性と適用範囲に関する評価と考察を行った．

Implementation and Evaluation of SaaR : Sandbox as a Request

JUNYA KANI[†1] TAKEHISA KATO[†2]
FUMIHIKO MAGATA[†3] YOSHIMI TESHIGAWARA[†4] RYOICHI SASAKI[†4]
MASAKATSU NISHIGAKI[†1]

We proposed a method that provides a disposable virtual server to each request for a real server from a client-user, as countermeasure of attack to server(s) by malicious users. Namely, the proposed scheme, Sandbox as a Request (SaaR), generates one-time virtual machine against each access request from any client-user, regardless of legitimate user or malicious user, and then creates a copy of a real server in the sandbox. The copied virtual server provides a service to each client-user, and is cleared out when it is finished providing service appropriate to the request by the user. Even if a malicious client-user succeeds in tampering data of the copied virtual server, the real server is working without fault. This paper implements this system in the form of Web-Server, evaluates and discusses about the feasibility and the applicability.

1. はじめに

Web サービスや組織の LAN 内など，あらゆるネットワーク上に，サーバ・クライアント方式のシステムが存在している．そのような中，不正者による Web サーバへの攻撃は後を絶たず，Web サーバの改ざんは深刻な被害[1]を引き起こしており，ドライブバイダウンロード攻撃[2][3]や水飲み場攻撃[4]といった攻撃にもつながっている．また，近年の標的型攻撃では，不正者は組織内に属する 1 台のクライアント端末を乗っ取った後に，LAN 内のローカルサーバ（例えば，共有プリンタや社内 DB など）に侵入し，そのサーバを踏み台にして組織内ネットワークへの感染を広げていく[5]．サーバは，ネットワーク上の全てのクライアントのリクエストに応じてサービスを提供する．そのため，不正者によってサーバが攻撃を受け，機能やデータが改ざんされてしまうと，その後サーバにアクセスする全てのクライアントが改ざんの影響を受けることになる．インターネットにおけるポータルサイトや大規模な組織のローカルサーバなど，クライアントからのアクセスが多いサーバほど，被害の影響は大きいものとなる．

サーバに対する攻撃を検知しようとする既存の研究や製品が多数存在しているが[6-12]，攻撃には未知の脆弱性が利用されることも多いため，攻撃を確実に検知することは難しい．そこで我々は，各クライアントからのサーバに対するリクエストごとに，仮想サーバ（仮想マシンによって実装されるサーバ）をワンタイムで提供する方式（SaaR : Sandbox as a Request）[13]を提案した．SaaR では，正規のクライアントからのアクセスに対しても，不正なクライアントからのアクセスに対しても，サーバの「複製」がその都度サーバ内のサンドボックスの中に生成され，仮想サーバ（複製サーバ）のサービスがクライアントに提供される．仮想サーバは，クライアントからのリクエストに応じたサービスを終えた時点で使い捨てられる．これによって，もし不正者がサーバの脆弱性をつきサーバ内のデータの改ざんに成功したとしても，それは不正者に一時的に提供されたサーバの複製であり，サーバ本体は無傷を保つことになる．したがって，それ以降の正規クライアントからのリクエストが入来した際に，新たにその時点でサーバ本体を複製することによって生成される仮想サーバも真正のままであり，正規クライアントが改ざんの被害を受ける

†1 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University
†2 情報処理推進機構
Information-technology Promotion Agency
†3 NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories
†4 東京電機大学未来科学部
School of Science and Technology for Future Life, Tokyo Denki University

ことはない．

本稿では，KVM（Kernel-based Virtual Machine）[14] を利用して，Web サーバの形態で SaaR を実装し，提案方式のパフォーマンス評価を行う．また，提案方式の適用範囲について調査する．

以降，2 章で SaaR のコンセプトについて述べる．3 章で SaaR の実装について説明し，4 章で SaaR の評価について詳述する．5 章で本研究の考察を述べた後，6 章で本論文をまとめる．

2. Sandbox as a Request

2.1 コンセプト

サーバの安全維持に対しては，OS やアプリケーションを常に最新の状態に保つことが第一に必要である．しかし，例えば独自のアプリケーションを利用している企業においては，OS の更新がアプリケーションの動作に弊害を与える場合があるなど，サーバ管理は一概に容易であるとは限らない．また近年では，未知の脆弱性を突く攻撃が少なくない．ゼロデイ攻撃に対してさえも効果が期待される防御策が肝要となる．

そこで我々は，各クライアントからのサーバに対するリクエストごとに，仮想サーバ（仮想マシンによって実装されるサーバ）をワンタイムで提供する方式（SaaR : Sandbox as a Request）[13]を提案した．SaaR では，クライアントからのサービスリクエスト単位でサンドボックスがあてがわれる．すなわち，正規のクライアントからのアクセスに対しても，不正なクライアントからのアクセスに対しても，サーバの「複製」がその都度サーバ上のサンドボックスの中に生成され，仮想サーバ（複製サーバ）のサービスがクライアントに提供される．仮想サーバは，クライアントからのリクエストに応じたサービスを終えた時点で使い捨てられる．

もし不正者がサーバの脆弱性を突きサーバ内のデータの改ざんに成功したとしても，それは不正者に一時的に提供されたサーバの複製である．複製サーバは，リクエストに対するレスポンスの終了とともに（感染したゲスト OS ごと）使い捨てられるため，サーバ本体は無傷を保つことになる．したがって，それ以降の正規クライアントからのリクエストが入来した際に，新たにその時点でサーバ本体を複製することによって生成される仮想サーバも真正のままであり，正規クライアントが改ざんの被害を受けることはない．ゲスト OS 越しにハイパーバイザ（ホスト OS）を直接攻撃されることがない限り，サーバは安全にサービスを提供し続けることが可能である．

インターネット上および LAN 内のあらゆるサーバに対して SaaR の導入が可能であるが，本稿では具体的な適用先として Web サーバを例に採って以降の検討を進めることとする．

2.2 前提条件

Web サーバに SaaR を適用するにあたっての 3 つの前提条件を記す．

（1）管理者へのなりすましが行えないこと：

SaaR は，脆弱性利用型攻撃に対する対策であるため，不正者が管理者になりすます攻撃については対象外となる．すなわち SaaR は，何らかのなりすまし対策技術との併用が必須となる．

（2）ホスト OS への攻撃が行われないこと：

SaaR は，ゲスト OS 上で稼働する Web サーバへの攻撃に対する対策であるため，ホスト OS への攻撃については対象外となる．一般的にシステムの下位レイアへの攻撃は難度が高く，実際，Web サーバ（Apache など）の脆弱性報告数と比べ，仮想マシン（VMware など）の脆弱性報告数が少ないことは周知である*．

（3）リードオンリ型の Web コンテンツであること：

Web サービスは，リードオンリ型（一般利用者はサービスを利用することのみが可能）とリードライト型（一般利用者自身が Web コンテンツを追記できる）に大別される．SaaR は，リードオンリ型 Web サーバを守る対策である．すなわち SaaR は，（SNS や掲示板に対する適用は難しいが）企業における製品紹介や官公庁における広報などの情報提供型の Web サービスが適用対象となる．企業や官公庁にとって，Web ページの改ざんは信用性の失墜を招きかねない大きな脅威であるため，情報提供サービスの保護を実現する SaaR の有効性は非常に大きい．

3. SaaR の実装

3.1 基本システム

まず，SaaR のコンセプトを最もシンプルに実装した場合の構成（図 1）について概説する．図 1 において，「リバースプロキシデーモン（RPD）」はホスト OS のプロセスであり，仮想サーバのライフサイクルの管理と仮想サーバ・クライアント間の通信を制御する．「Original VM（VO）」はホスト OS のディスク内に格納されている仮想サーバのオリジナルイメージであり，「VM 管理表」はホスト OS が仮想サーバ群の利用状況を管理するために用いるテーブルである．

① Web サーバにクライアント A（CA）からの HTTP リクエストが入来する．

* JVN 脆弱性対策情報データベース（http://jvndb.jvn.jp/）によると，2013 年 1 月から 12 月の間に公表された仮想マシンに関する脆弱性報告数は，それぞれ KVM に関するものが 21 件，VirtualBox に関するものが 2 件，VMware に関するものが 20 件であり，代表的な仮想マシンを合わせて 43 件であった．一方，Apache に関する脆弱性報告数は 88 件であり，代表的な Web サーバシステムである Apache のみでも，仮想マシンに比べて倍以上の脆弱性報告数があることが分かる．

② ホスト OS は，VO を A 用の仮想サーバ（VA）のディスク領域に複製する．
③ ホスト OS は，VA のゲスト OS をブートし，更に Web サービスを起動する．
④ RPD は，CA と物理サーバとの間の通信を CA と VA との間の通信としてディスパッチする．ホスト OS は，この情報を VM 管理表に記録する．
⑤ CA と VA の間に HTTP セッションが確立し，VA によって CA にサービスが提供される．
⑥ CA がサービスの利用を終えた時点（今回の実装では CA から 1 分以上応答がないとタイムアウト）で，HTTP セッションは終了し，VA は使い捨てられる（VA のメモリおよびディスク領域はすべて解放される）．

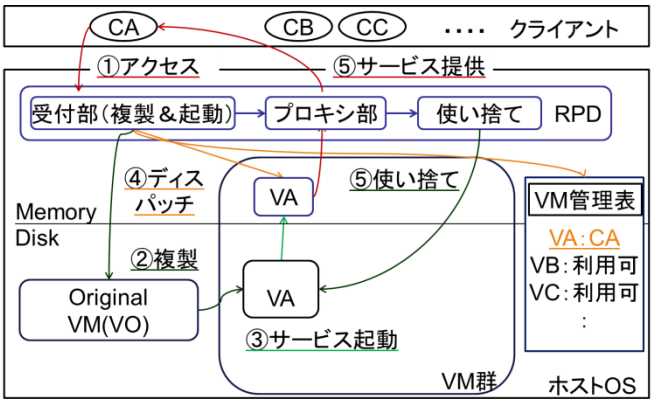

図 1　SaaR の実装（基本システム）

3.2 リクエストのホットスタンバイ

現在の仮想サーバ技術においては上記②および③に要する時間が大きい．Web サービスの業界では「Web サービスを提供する速度」が重要視されており，2009 年の調査[15]では Web ページの表示は 2 秒以下にするべきであるという指針も示されている．よって，サービス提供時間の効率的な削減を目指す．

クライアントからのリクエストの待機方法によって，SaaR のサービス提供時間の短縮が期待できる．具体的な待機方法としては以下の三つが挙げられる．

（α）コールドスタンバイ（図 1 の方法）：

リクエストのたびに，仮想サーバのディスクイメージを複製し（②），ゲスト OS をブートした上で Web サービスを起動する（③）．リクエストが入来してから仮想サーバを用意するためサービス開始までに多くの時間を要する一方，待機時のディスク消費，メモリ消費はない．

（β）ウォームスタンバイ：

待機時に仮想サーバのディスクイメージの複製（②）は済ませておき，リクエストの入来を受け，ゲスト OS のブートと Web サービスの起動を行う（③）．リクエストからサービス開始までの時間が幾分短縮される一方で，待機時もディスクを消費する．

（γ）ホットスタンバイ：

②と③を済ませた状態で仮想サーバを待機させておく．リクエストとともに Web サービスを提供することができるが，ディスクとメモリは常に消費されている状態となる．

最近では，外部記憶メディアは大容量・低価格が著しいためディスク容量の増加は大きな問題にはならない．また，物理メモリ容量を超えて膨大なメモリ空間の使用を可能にする仮想メモリ技術は既に一般的に利用されている．以上より，ホットスタンバイ方式（γ）が好ましいと言える．

3.3 スナップショットの利用

スナップショットとは，仮想マシンのある時刻におけるメモリイメージを圧縮して（ディスクに）保存しておく技術である[16][17] †．ゲスト OS をブートした上で Web サービスを起動した直後のメモリイメージをスナップショットとして（ディスク領域に）保存しておく．これによって，仮想マシンをクリーンな状態（ゲスト OS をブートした上で Web サービスを起動した直後の状態）に戻すことができる．これによって，図 1 の②と③に要する時間が短縮される．

3.4 リソースの効率的な利用

3.2 節でディスク容量，メモリ容量は問題にならないと説明したが，リソースの消費を抑えることができるに越したことはない．特に，ホットスタンバイ方式では，仮想サーバ 1 台当たりのリソース消費量が少ないほど，実マシン上により多数の仮想サーバを用意することができる．

メモリ消費量の効率化に関しては，Linux に用意されている KSM（Kernel SamePage Merging）と呼ばれる機能を利用することができる[18]．SaaR においては，全クライアントに対し同一の Web サービスを提供する（全クライアントが同一の仮想サーバを利用する）ため，メモリ上には重複するメモリページが多く存在することが予想できる．よって，KSM の導入効果は高く，メモリのオーバヘッドは低く抑えることができると期待される．

ディスク消費量の効率化に関しては，仮想マシンが有する，仮想マシンのオリジナルイメージから差分イメージを作成する機能（ディスクマージ機能と呼ぶ）を利用することができる[19]．ただし，メモリ情報であるスナップショットについてはディスクマージ機能による仮想サーバ間共有はできず，すべての仮想サーバに対して個別にスナップショットを持たせる必要がある．

† 我々の調査によると，ディスク容量 8GB の VM イメージから作成したスナップショットはおおよそ 500MB 前後のディスク容量で収まることが分かっている．ただし，VM イメージの内容によって，増減の可能性がある上に，スナップショットに対して書き込みを増やすほど容量は大きくなる．

Vol.2014-DPS-158 No.19
Vol.2014-CSEC-64 No.19
2014/3/6

3.5 Web コンテンツの更新

SaaR で Web サービスを提供するにあたっては，正規の管理者によって Web コンテンツが更新された際には，すべての仮想サーバにその変更を適用する必要がある．これに対して，OS や Apache などが利用するシステム系コンテンツと Web コンテンツを分離し，Web コンテンツについては NFS（Network File System）[20]を利用した管理を行う．具体的には，ホスト OS で Web コンテンツのオリジナルデータとなる「オリジナルコンテンツ（CO）」を保有しておき，それぞれの仮想サーバではこのオリジナルコンテンツ（CO）をリードオンリ形式でマウントする．これによって，サーバ管理者が管理者権限でホスト OS の持つオリジナルコンテンツ（CO）を更新すれば，仮想サーバの Web コンテンツも自動的に更新が適用されることになる．

3.6 実装システム

3.1 節（図 1）の基本方式に対し，3.2～3.5 節の改良を追加した方式を，図 2 を用いて説明する．図 2 において，「Original Contents（CO）」，「Original VM（VO）」，「Original Snapshot（SO）」はホスト OS のディスク内に格納されている仮想サーバのオリジナルイメージであり，CO はクライアントに提供される Web コンテンツに関するデータ，VO は Web コンテンツ以外のデータ，SO はスナップショットのデータである．

① 仮想サーバ VA～VN のそれぞれのディスク領域に VO および SO を複製した上で，すべての仮想サーバをホットスタンバイさせる．仮想サーバを立ち上げるにあたり，ディスクマージ機能によって差分イメージのみが複製され，CO については NFS マウントを行う．

② Web サーバにクライアント A（CA）からの HTTP リクエストが入来する．

③ RPD は，CA と物理サーバとの間の通信を CA と VA との間の通信としてディスパッチする．ホスト OS は，この情報を VM 管理表に記録する．

④ CA と VA の間に HTTP セッションが確立し，VA によって CA にサービスが提供される．

⑤ CA がサービスの利用を終えた時点（今回の実装では CA から 1 分以上応答がないとタイムアウト）で，HTTP セッションは終了し，VA は使い捨てられる．同時に，VA に対しスナップショットの復元を行い，Web サービスをホットスタンバイさせる．（④のサービス提供の間に，①の時点で VA のディスク領域に複製されたオリジナルイメージ自体が改ざんされている可能性があるため，VA のメモリおよびファイルは一旦すべてクリアした上で，再度①を実行する．）

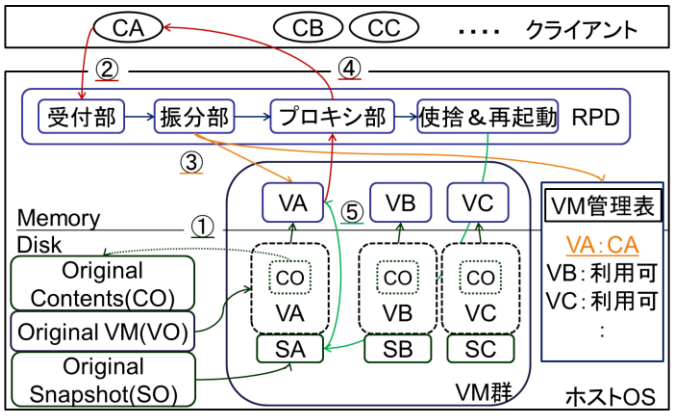

図 2　SaaR の実装（改良方式）

3.7 実装端末のスペック

実装に用いた端末の性能は以下の通りである．

- 実装端末：Ubuntu 13.04 (32 bit), Core-i7-CPU860 (2.8 GHz), 7.9 GB (RAM)
- 仮想環境：KVM（qemu-system-x86_64 , QEMU emulator version 1.4.0）
- 仮想マシン：Linux debian 3.2.0-4-686-pae #1 Debian 3.2.51-1 i686 GNU/Linux, core2duo, 1.0 GB (RAM) , Apache 2.2.22
- Web コンテンツ：文字 100 文字程度（216 byte）と画像 1 枚（2.0 MB）

オンラインショップサーバなどの超高負荷の Web サーバにおいてはハイスペックな物理端末を複数台連携して稼働させていることが一般的であるが，SaaR の適用先と考えられる情報提供型のサーバ（2.2 節前提条件（3））においては，物理端末 1 台構成の Web サーバが一般的である．このため本稿では，物理端末 1 台の上で SaaR を実装した．また，今回の実装に使用した物理端末はハイスペック機ではないが，本機におけるパフォーマンス低下の「割合」から SaaR 導入時のオーバヘッドを測ることができると考えている．

4. SaaR の評価

本稿では，SaaR のパフォーマンスに関する評価と適用範囲に関する評価を行う．パフォーマンス評価においては，レスポンスタイムやメモリ消費などの観点から，SaaR のオーバヘッドを測る．適用範囲の評価としては，2.2 節の前提条件（3）に挙げたような SaaR の適用先となり得る Web サイトの割合（上場企業のサイトの中で，リードオンリなコンテンツを提供するサイトがどの程度存在しているか）を調査する．

Vol.2014-DPS-158 No.19
Vol.2014-CSEC-64 No.19
2014/3/6

4.1 パフォーマンス評価

4.1.1 レスポンスタイム

SaaR を導入した Web サーバの速度評価を行う．クライアント端末を用意して，「クライアント端末が Web サーバにリクエストを送信してから，クライアント端末がレスポンスを受信するまでにかかった時間」を測定する．クライアント端末のスペックは，Windows 7 SP1 (32 bit)，Core-i5-2540M (2.6 GHz)，4.0 GB (RAM)，FireFox 26.0 であり，速度の測定には FireFox のアドオン Firebug 1.12.6[21] 用いた．SaaR（スナップショット，ホットスタンバイ）のレスポンスタイムを T1 とする．比較のために，SaaR（スナップショット，ウォームスタンバイ）のレスポンスタイム（T2）と SaaR を導入していない状態でのレスポンスタイム（T0）も計測した．コールドスタンバイはウォームスタンバイ以上に時間がかかるため，T2 の計測までで十分と判断した．評価結果を表 1 に示す．結果は 10 回の計測を行った平均値と標準偏差である．

表 1　レスポンスタイム

	T0	T1	T2
平均値（秒）	0.008	0.025	7.799
標準偏差（秒）	0.006	0.006	1.092

表 1 より，ホットスタンバイ（T1）であれば，SaaR を導入していない Web サーバのレスポンスタイム（T0）からの速度劣化は若干見られるものの，人間が負担に感じるほどの速度低下は見られないことが分かった．

4.1.2 負荷テスト

SaaR（スナップショット，ホットスタンバイ）を導入した Web サーバに対して，負荷テストを行う．スケーラビリティ（台数効果）を測るために，（C）仮想サーバ 1 台構成，（D）仮想サーバ 10 台構成，（E）仮想サーバ 20 台構成のそれぞれに対して負荷テストを行う．また，比較のために，（A）SaaR も仮想マシンも利用しない Web サーバ，（B）SaaR は導入していないが，1 台の仮想マシン上で Web サービスを稼働している Web サーバに対しても負荷テストを実施する．

負荷テストには，ベンチマークツール「httperf」[22]を用いた．クライアント端末を用意して，クライアント端末側の httperf から Web サーバに対して，接続完了回数が 1000 回に至るまで接続試行を実施する．その際 1 秒間に 50 回，100 回，200 回，500 回，1000 回の連続接続試行を実施し，それぞれのオーバヘッドを評価する．クライアント端末は，Windows 7 SP1 (32 bit)，Core-i5-2540M (2.6 GHz)，4.0 GB (RAM) の VirtualBox 上の Ubuntu12.04LTS (32 bit)，Core-i7-2600K (3.4GHz)，1.0 GB (RAM)を用いた．

「接続完了までにかかった時間」と「リプライの受信成功回数」を用いて評価する．結果をそれぞれ図 5，図 6 に示す．図 5 の縦軸は接続完了までにかかった時間，図 6 の縦軸はリプライの受信成功回数，両グラフの横軸は 1 秒間の接続試行回数である．

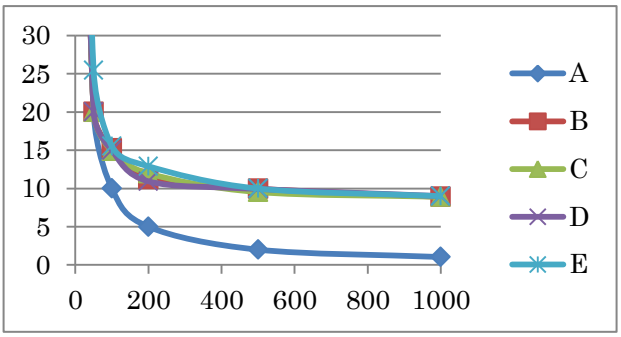

図 3　接続完了までにかかった時間

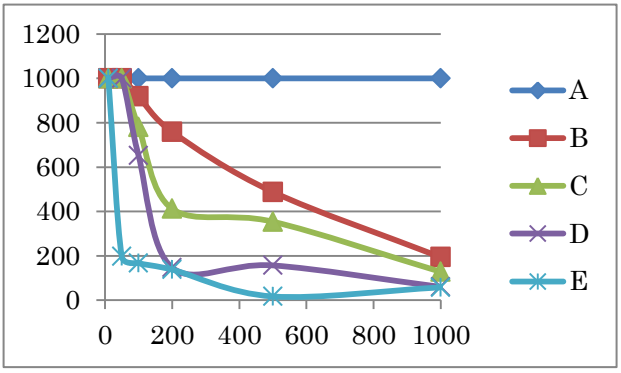

図 4　リプライ受信数

図 3 を見ると，（A）のグラフは 1 秒間の接続試行回数に対して，200 回であれば 5 秒，1000 回であれば 1 秒のように遅延なく 1000 回の接続が完了していることが分かる．それに対して，仮想サーバを利用する場合（B~E）は，1 秒あたり 100 回の接続試行を超えたあたりから数秒の遅延が発生しており，接続完了までにある一定のオーバヘッドがかかっていることが分かる．仮想サーバの数を増やしても接続完了までにかかった時間がほとんど変化していないことから，仮想マシンのネットワーク機構にボトルネックが生じていると推測できる．

一方，図 4 を見ると，リプライの受信成功回数は，SaaR の導入の有無や仮想サーバの台数によって大きく差が見られる．仮想サーバ 20 台を起動する SaaR（E）では，1 秒間の接続試行回数が 100 回を超えたあたりから，（A）の 10 分の 1 以下の受信成功回数となってしまっていることが分かった．

しかし，例えば静岡大学の Web ページ（トップページ）に対する 2012 年度の平均リクエスト数は(多いときでも)，1 秒間に 7 件である．この程度のアクセス頻度の Web サーバであれば，SaaR を導入した場合もほぼ十分なパフォーマ

Vol.2014-DPS-158 No.19
Vol.2014-CSEC-64 No.19
2014/3/6

ンスが期待される．

4.1.3 メモリ使用量

SaaR を導入した Web サーバにおいて，KSM を利用することによって，2 台，5 台，10 台の仮想サーバを同時に動作させたとき，どの程度メモリを削減できるかを測定した．結果を図 5 に示す．横軸は仮想サーバの台数を示し，縦軸は左側が仮想サーバ 1 台あたりの削減量，右側が物理端末全体での削減量となっている．

図 5 を見ると，仮想サーバ 10 台の場合で約 3GB のメモリの削減が見られ，1 台あたり約 300MB のメモリを節約することができていることが分かった．今回の実装では，仮想サーバ 1 台に対して，1GB のメモリを割り当てていたことから，1/3 程度のメモリ削減が達成されている．また，仮想サーバの台数が増えるほど，メモリ削減の効果は増加傾向にある．

Dense Ship [23]では，メモリの共通化の行えなかったページを圧縮することによって，メモリ使用量を抑えることに成功している．同様な仕組みを SaaR に取り入れることによって，メモリ使用量を更に削減できる可能性がある．また，メモリスキャンのオーバヘッドや多大なメモリ容量を使い続けることによる消費電力の増加に関しても今後の検討が必要である．

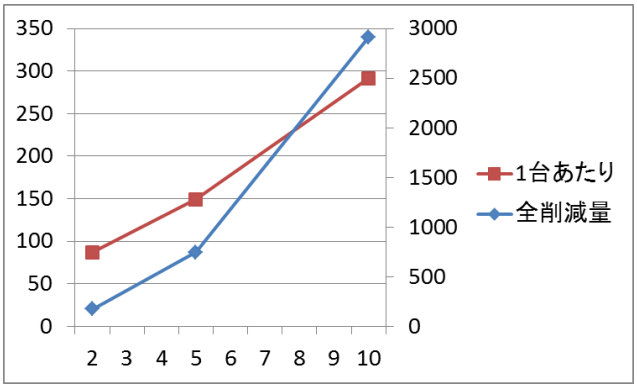

図 5　KSM によるメモリ削減量

4.1.4 仮想サーバの台数

SaaR では，クライアントからのリクエストごとに仮想サーバをあてがうため，ホットスタンバイで待機させておくことのできる仮想サーバの台数分しか，リクエストを受理することができない．(仮想サーバの台数分以上のリクエストが入来した場合は，その後のリクエストは待たされる．) すなわち，それぞれの仮想サーバは「あるリクエストを受理した後，そのリクエストに対するサービスを終え，ホットスタンバイの状態に戻るまで」の間，次のリクエストを受理することができない．

これを逆に考えると，1 台の仮想サーバが「あるリクエストを受理した後，そのリクエストに対するサービスを終え，ホットスタンバイの状態に戻るまで」の間に入来するリクエストの和 N_{VM} に相当する数の仮想サーバが並列に存在すれば，すべてのリクエストに対応することができるはずである．

よって，Web サーバに入来するリクエストの平均アクセス数 N_{ACCESS}，1 回当たりのサービス提供にかかる平均所要時間 $T_{SERVICE}$，仮想サーバの使い捨てとリブートにかかる平均時間 $T_{STANDBY}$ から，必要な仮想サーバの台数 N_{VM} を算出することができる．具体的には，

$$N_{VM}＝N_{ACCESS}×(T_{SERVICE}＋T_{STANDBY}) \quad (1)$$

となる．

例えば，静岡大学の Web ページ（トップページ）においては，2012 年度実績で（アクセスの多いときでも）毎秒の平均リクエスト数が 7 件，サービス提供にかかる時間が 3 秒であった．4.1.1 節の実験結果からスナップショットを利用した際のリブートにかかる時間（T2）は，おおよそ 8 秒であった．よって，式(1)より最低限必要な仮想サーバの台数は 77 台と分かる．

4.2 適用範囲の評価

4.2.1 評価の概要

SaaR の適用先となり得るサイトの割合（上場企業のサイトの中で，リードオンリなコンテンツを提供するサイトがどの程度存在しているか）を調査する（図 5）．

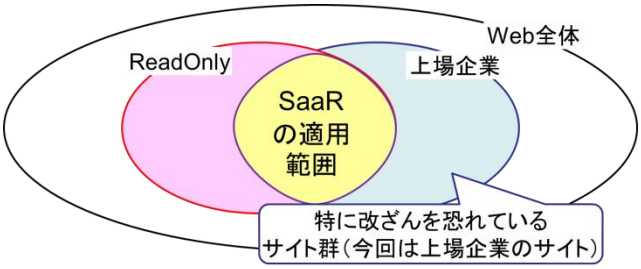

図 6　SaaR の適用範囲

今回の調査では，上場企業の Web サイトからランダムに 70 社を選び，目視にて Web サイトのサービスがリードオンリであるかの確認を行った‡．ここでのリードオンリの定義は，「(ある) ユーザが書き込んだ情報を（他の）ユーザが見ない」こととする．例えば，企業情報の提示などの Web コンテンツのみを提供している Web サイトは「リードオンリ」であり，掲示板への書き込み（SNS 等のユーザの投稿情報）や Amazon などの商品レビューなどを含む Web サイトは「リードライト」である．サイト内検索は Web コンテンツ自体がアップデートされるわけではないので「リードオンリ」である．「お問い合わせフォーム」等の Web

‡ 上場企業の Web サイト一覧より，EXCEL の RAND 関数を用いて，70 社を選択した．また，目視については，同一サイトを二名ずつで確認することによってダブルチェックを行った．

ページを通じて管理者に質問を送ることができる Web サービスも，質問内容が管理者宛にメールで送信されるものは，Web コンテンツ自体がアップデートされるわけではないので「リードオンリ」である．ただし，今回の調査では，Web サイトに登録した上で会員専用のページまでを確認することまでは行っていない．このため，会員専用ページが含まれていた場合には（実際にはリードライト型の Web コンテンツがなかったとしても）「リードライト」型のサイトであると判断している．上場企業数は 2014 年 2 月 3 日現在，3418 社である．70 社の無作為抽出によって，統計調査の信頼区間をおおよそ 20%に収めることができる．

調査結果を表 2 に示す．リードオンリ型サイトは 70 社中 49 社であった．母比率 p に対して，標本比率$\bar{p}$ と調査対象数n から，式(2)より，統計学的に母集団の比率は，95%の確からしさで 59.3%～80.7%の間にあることが分かる．すなわち，SaaR の導入による効果が期待される上場企業は全体の約 7 割に達する．

$$\bar{p} - 1.96 \times \sqrt{\frac{\bar{p} \times (1-\bar{p})}{n}} < p < \bar{p} + 1.96 \times \sqrt{\frac{\bar{p} \times (1-\bar{p})}{n}} \quad (2)$$

表 2　適用範囲の調査結果

リードオンリ型サイト	会員専用サイト（今回はリードライト型サイトに含める）	リードライト型サイト
49 社	18 社	3 社

5. 考察

5.1 安全性検証

SaaR に脆弱性利用型攻撃を仕掛けることによって，SaaR の安全性を確認する．今回は，以下の流れで検証を行った．①脆弱性（CVE-2013-2251）を持つ仮想サーバ（Apache Tomcat 7.0.50, JDK 1.6.0_27, Apache Struts 2.0.9）を用意し，クライアントからのリクエストが発生したら常に当該仮想サーバがリクエストを受理するように設定しておく．②クライアント端末から，当該仮想サーバに対し（CVE-2013-2251 の）攻撃を仕掛ける．③クライアント端末からのアクセスを終了する．④再度，クライアントからリクエストを発信し，仮想サーバにこのリクエストを受理させる．⑤④の時点の仮想サーバには②の攻撃の痕跡がないことを確認する．検証の結果，③の時点で攻撃を受けた仮想サーバは使い捨てられ，④の時点の仮想サーバには②の攻撃の痕跡がないことが確認された．

ただし，サーバに対する攻撃方法は多種多様であるため，これだけの検証によって SaaR における完全な安全性が確認できたわけではない．SaaR の安全性の証明は今後の重要な課題である．

5.2 ログの記録

SaaR では，仮想サーバがクライアントのリクエストごとに使い捨てられる．これは，不正者による攻撃から Web サーバを守るというメリットを産む一方で，不正者の活動の証拠が消滅するというデメリットをはらむ．すなわち，不正者が SaaR の仮想サーバを踏み台にして外部への攻撃を行なった場合，不正者が仮想サーバとのセッションを終了した瞬間に，当該仮想サーバは自動的にクリアされて，不正アクセスの証拠も抹消されることになる．

そこで，SaaR を実現する上では，各仮想サーバのログをホスト側にて管理する仕組み（ログ集中管理サーバ）を取り入れることが必要になるだろう．既存研究として，仮想マシンからのログを安全にホスト側で管理する仕組みが提案されている[24]．このような方式と組み合わせることによって，SaaR においても各仮想サーバによるログを安全にホスト側に蓄えておく事が可能となる．

6. 今後の課題とまとめ

本稿では，サーバへの攻撃に対する対策として提案した SaaR（Sandbox as a Request）を，Web サーバの形態で実装を行い，そのパフォーマンスと適用範囲に関する評価を行った．

SaaR は，各クライアントからのリクエストのたびに，仮想サーバをワンタイムで提供する方式であるため，レスポンスタイムの低下が懸念される．また，リクエストの数だけ仮想サーバが同時に立ち上がるため，オーバヘッドが甚大となる可能性がある．スナップショット機能を利用したホットスタンバイ方式によってレスポンスタイムの維持を達成し，KSM，ディスクマージ機能，NFS を利用することによってメモリとディスクの使用量のオーバヘッドを抑制した．ただし，メモリに関しては更なるオーバヘッドの削減が必要となる．

適用範囲に関する調査では，SaaR は上場企業の内，7 割程度の Web サイトに対して適用可能であることが分かった．

しかし，SaaR には未だ複数の課題が残されており，今後は，データベースのライトバックに関する検討と実装，安全性の評価，Web サーバ以外の SaaR の適用についての検討を進めていきたいと考えている．

謝辞　SaaR の実装にあたって，ご協力頂いた富士通研究所の西口直樹様，岡山大学の山内利宏先生，立命館大学の毛利公一先生，その他調査や研究にご協力して下さった皆様に，謹んで感謝の意をここに表します．

参考文献

1) Web サイト改ざんに関する注意喚起，JPCERT/CC, <https://www.jpcert.or.jp/at/2013/at130027.html>(参照 2014/02/10).

2) Provos, N., McNamee, D., Mavrommatis, P., Wang, K. and Modadugu, N. : The Ghost In The Browser Analysis of Web-based Malware, Proc. HotBots'07, Usenix, pp.4-4 (2007).
3) JM Hipolito: Stolen FTP Credentials Key to Gumblar Attack, TrendLabs Malware Blog, Jun 2009, <http://blog.trendmicro.com/stolen-ftp-credentials-key-to-gumblar-attack/>(参照 2014/02/10)
4) Symantec： ウェブサイトセキュリティ脅威レポート 2013 Part1, White Paper (2013).
5) 特定非営利活動法人 日本セキュリティ監査協会，APT による攻撃対策と情報セキュリティ監査研究会： APT 対策入門，株式会社インプレス R&D(2012).
6) 秋山満昭，佐藤一道，岩村誠，伊藤光恭：Gumblar の長期観測による分析，電子情報通信学会技術研究報告，IA，インターネットアーキテクチャ Vol.110，No.78，pp.69-74 (2010/06/10).
7) 井上大介:サイバー攻撃観測網について, ITU ジャーナル, Vol.43, No.4 ， pp.12-14(2013/04).
8) 上松晴信，名坂康平，酒井崇裕，西垣正勝：相補的な Web 感染型マルウェア検知方式の提案，情報処理学会研究報告，2011-CSEC-52-53, pp.1-6 (2011/03/03).
9) 山田正弘，森永正信，海野由紀，鳥居悟，武仲正彦：組織内ネットワークにおける標的型攻撃の検知方式，情報処理学会研究報告，2013-CSEC-62-53，pp.1-6, (2013/07/11).
10) isAdmin ‐ Web コンテンツ／アプリケーションの改ざん検知／自動復旧, J-SYS, <http://www.jsys.co.jp/solution/isadmin.html/>(参照 2014/02/10)
11) WebS@T，株式会社ネットワールド, <https://www.kaizankenchi.jp/>(参照 2014/02/10)
12) Web サイトの「変更」と「改ざん」を見極める検知システム「WebS@T」に迫る，クラウド Watch ，<http://cloud.watch.impress.co.jp/epw/cda/security/2008/08/11/13520.html>(参照 2014/02/10)
13) 可児潤也，小林真也，加藤岳久，間形文彦，勅使河原可海，佐々木良一, 西垣正勝:SaaR:Sandbox as a Request の提案, CSS2013 (2013/10).
14) A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori.: “kvm: the linux virtual machine monitor”, In Proceedings of the Linux Symposium, Vol. 1, pp. 225–230, 2007.
15) Forester Research: eCommerce Web Site Performance Today, WhitePaper (2009/08/17)
16) Understanding and exploiting snapshot technology for data protection, Part 1: Snapshot technology overview, <https://www.ibm.com/developerworks/tivoli/library/t-snaptsm1/> (参照 2014/02/12)
17) Understanding virtual machine snapshots in VMware ESXi and ESX, <http://kb.vmware.com/selfservice/microsites/search.do?cmd=displayKC&externalId=1015180> (参照 2014/02/12)
18) A.Arcangeli: "Increasing memory density by using KSM", Linux Symposium, 2011
19) VMware: “Storage Considerations for VMware Horizon View 5.2”, WhitePaper (2013/05/24)
20) Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, Bob Lyon (1985). "Design and Implementation of the Sun Network Filesystem". USENIX.
21) FireBug, <http://getfirebug.com/> (参照 2014/02/12)
22) Welcome to the httperf homepage, <http://www.hpl.hp.com/research/linux/httperf/> (参照 2014/02/12)
23) 川古谷裕平，岩村誠，伊藤光恭 : Dense Ship: サーバ型ハニーポット用仮想マシンモニタ, 電子情報通信学会技術研究報告, ICSS, 情報通信システムセキュリティ ， Vol.111， No.82, pp.63-68(2011/06/09).
24) 安藤類央，橋本正樹，山内利宏：仮想化技術による安全なファイルアクセスログ外部保存機構，情報処理学会論文誌， Vol.54 No.2 pp.1-11(2013/02)