OUassister: 仮想マシンのオフラインアップデート機構

塩田裕司　光来健一

長期間使用していなかった仮想マシンはソフトウェアに脆弱性が見つかっていることが多い。従来は再開後にセキュリティアップデートを行っていたが、この方法では、アップデート中にネットワーク経由で攻撃を受ける可能性がある。そこで、停止している仮想マシンのアップデートを行うオフラインアップデートが提案されているが、シャットダウンした仮想マシンにしか適用することができなかった。本論文では、サスペンドした仮想マシンのオフラインアップデートを可能にする OUassister を提案する。OUassister はオフライン時にはアップデートのエミュレーションのみを行う。そして、レジューム後に仮想マシンにエミュレーション結果の反映を行わせる。我々は Xen 上で VM Shadow を用いてエミュレーション環境を構築し、Aufs を用いて更新されたファイルの抽出を行った。

1　はじめに

仮想マシンを用いると一台の計算機上に複数の計算機を仮想的に作成することができる。仮想マシンは必要な時だけ動かすことが容易であるため、それ以外の時にはサスペンドして停止させていることが多い。そのため、サーバにもデスクトップにも長期間使用しない仮想マシンが多く存在している。

長期間停止していた仮想マシンは、使用していない間に脆弱性が発見されていることが多く、そのまま再開すると攻撃を受ける可能性が高い。仮想マシンの再開後にアップデートを行う従来手法は、セキュリティアップデート中に攻撃を受ける可能性が高く危険である。近年、停止している仮想マシンのアップデートを行うオフラインアップデートも提案されている [4] [5] が、シャットダウンされた仮想マシンを対象としており、サスペンドされた仮想マシンに適用すると仮想ディスクの整合性がとれなくなる。

そこで、サスペンドした仮想マシンのオフラインアップデートを可能にする OUassister を提案する。OUassister はオフラインで仮想マシンのアップデートのエミュレーションを行い、レジューム後にエミュレーション結果を仮想マシンに反映させる。OUassister は仮想マシンに対してアップデータの実行をエミュレーションするための環境を提供し、アップデータによって更新されるファイルを抽出する。そして、仮想マシンがレジュームされてオンラインになった時に抽出した更新ファイルを仮想ディスクに対して適用される。これにより、仮想ディスクの整合性を保ったまま、可能な限り仮想マシンをオフラインでアップデートすることができる。

我々は OUassister を Xen 上に実装した。OUassister は VM Shadow [6] を用いてアップデータのエミュレーション環境を構築し、Union ファイルシステムの Aufs [2] を利用して更新されたファイルの抽出を行う。OUassister を用いて Ubuntu の apt コマンドを動作させ、パッケージのオフラインアップデートが行えることを確認した。また、レジューム後の処理時間を従来手法と比べ大幅に短縮できることも示せた。

以下、2 章で仮想マシンのアップデートの問題点に

Yuji Shiota, Kenichi Kourai, 九州工業大学, Kyushu Institute of Technology

ついて述べ、3 章で OUassister について述べる。4 章で OUassister を用いて行った実験について述べる。5 章で関連研究に触れ、6 章で本論文をまとめる。

2 仮想マシンのアップデート

仮想マシンは必要な時だけ動かすという使い方をすることができるため、長期間使わない仮想マシンも多く存在している。例えば、サーバは負荷に応じて仮想マシンの数を調整してスケールアウトできるようにするために、予備の仮想マシンを用意しておく場合がある。サーバ全体にかかる負荷が増えた時、予備の仮想マシンを動かして負荷を分散する。また、デスクトップでは Windows 7 の上で Windows XP を動かすなど、異なる OS を使うために仮想マシンが作成されることが多い。このような仮想マシンは特定の OS を必要とする時だけ動かし、それ以外の時は停止させている。

長期間停止していた仮想マシン内の OS やアプリケーションには、脆弱性が発見されていることが多い。動いている仮想マシンであれば短期間でセキュリティアップデートが適用されるため、攻撃が蔓延する前に対策を行うことができる。しかし、停止している仮想マシンにはセキュリティアップデートが適用されないため、脆弱性が残ったままになる。

そのため、仮想マシンを再開した後でアップデートを行う従来の手法は危険である。仮想マシンを再開した時点で攻撃が蔓延していた場合、仮想マシンをインターネットに接続すると即座に攻撃を受ける可能性が高い。アップデートを行うにはインターネットに接続してアップデータをダウンロードしなければならないため、ネットワーク経由の攻撃を防ぐのは難しい。アップデートには時間がかかることも多く、その間に攻撃を受ける可能性も高くなる。

この問題を解決するために、仮想マシンのオフラインアップデートがいくつか提案されている。その一つは、アップデート専用環境を用意して仮想マシンのアップデートを行う手法である [1]。この方法では、アップデート専用環境内のサーバにアップデータをダウンロードしておき、その中で仮想マシンを動かしてアップデートを行う。外部ネットワークに接続しないため安全にアップデートを行うことができるが、アップデート専用環境を構築してメンテナンスする必要がある。

また、仮想マシンを停止させたままでオフラインアップデートを行う手法もある [4] [5]。この手法では停止中の仮想マシンのディスクに対してアップデートを適用する。この手法はシャットダウンした仮想マシンには有効であるが、サスペンドした仮想マシンには適用できない。サスペンドされた仮想マシンのディスクに書き込みを行うと OS 内の状態との整合性がとれなくなり、仮想マシンの仮想ディスクが壊れてしまうためである。仮想マシンのサスペンドは?よく用いられている。そのため、サスペンドされた仮想マシンもアップデートできるようにする必要がある。

3 OUassister

本論文では、サスペンドした仮想マシンのオフラインアップデートを可能にする OUassister を提案する。OUassister は仮想マシンがオフラインの間に仮想マシン内の OS やアプリケーションのアップデートのエミュレーションを行なっておく。そして、仮想マシンをレジュームしてオンラインになった直後にエミュレーション結果の反映を行う。

3.1 概要

OUassister はオフライン時に仮想マシン内の OS やアプリケーションのアップデートのエミュレーションを行う。図 1 に OUassister の全体の流れを示す。OUassister を使ってソフトウェアアップデートを実行すると、まずアップデータのダウンロードが行われる。このアップデータを実行すると、OUassister は仮想マシンに対してアップデートのエミュレーションを行う。この際に仮想マシンのディスクの状態を変更しないため、仮想ディスクの整合性がとれなくなることはない。OUassister はアップデータ実行のエミュレーションによって更新・削除されたファイルの情報などを記録しておく。

仮想マシンがレジュームされてオンラインになると、OUassister はアップデータ実行のエミュレーション結果を仮想マシンに反映する。OUassister は更新・

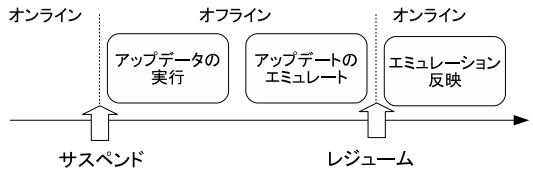

図 1　OUassister の流れ

削除されたファイルの情報を仮想マシンに転送し、仮想マシン内でディスクに対して実際にファイルの更新・削除を行う。仮想マシン自身にディスクのアップデートを行わせることにより、ディスクの整合性を保つことができる。アップデータにオフライン時に実行できない処理が含まれていた場合には、オンラインになった後で実行する。例えば、サーバの再起動などが挙げられる。

3.2　エミュレーション環境

OUassister は VM Shadow [6] を用いてアップデータのエミュレーション実行を行うための環境を構築している。VM Shadow はホスト OS に構築される実行環境であり、仮想マシン内の情報を参照する既存のプログラムを動作させることができる。VM Shadow 内では Shadow ファイルシステムが提供され、仮想マシンのディスク内のファイルへのアクセスが可能である。これにより、図 2 のように VM Shadow 内で実行されるアップデータは仮想マシン内で動作しているのと同じようにファイルを扱うことができる。

VM Shadow はプロセスが発行するシステムコールのエミュレーションも行う。例えば、アップデータが OS の情報を取得するシステムコールを発行した場合には、仮想マシン内の OS の情報をカーネルメモリから取得することでエミュレーションを行う。一方、アップデータをダウンロードする際に使われるネットワーク関連のシステムコールについては、エミュレーションを行わずにホスト OS に実行させる。

3.3　更新されたファイルの抽出

VM Shadow 内で実行したアップデータによるファイルの更新をエミュレーションできるようにするために、OUassister は Union ファイルシステム [3] を用

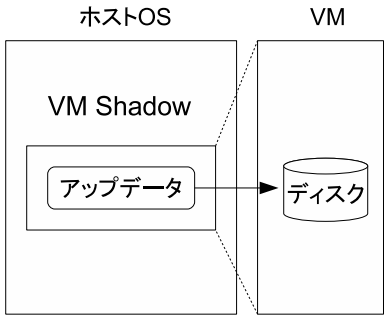

図 2　エミュレーション環境

いる。Union ファイルシステムは複数のディレクトリを透過的に重ねることができるファイルシステムである。現在の実装では、Union ファイルシステムとして Aufs [2] を用いている。Aufs ではそれぞれのディレクトリはブランチと呼ばれ、読み取り専用または読み書き可能の属性を設定できる。Aufs でマウントしたディレクトリには上のブランチから優先されて読み込まれる。OUassister では図 3 のように仮想マシンのディスクをマウントしたディレクトリを読み取り専用に設定し、更新されたファイルの保存用ディレクトリを読み書き可能に設定する。更新ファイルの保存用のディレクトリの優先度を高く設定することで仮想ディスクの上に重ねる。

Aufs でマウントしたディレクトリに対して書き込みを行うと、更新されるファイルは保存用のディレクトリに書き込まれる。既に存在するファイルの場合は新しいバージョンのファイルが保存用ディレクトリに作成され、新しいファイルを作成した場合は新しいファイルが保存用ディレクトリに作られる。いずれにしても仮想ディスクは更新されないため、仮想ディスクの整合性がとれなくなることはない。Aufs ではファイルの読み込みは優先度が高いディレクトリから行われるため、更新されたファイルが存在すれば保存用ディレクトリから新しいバージョンのファイルが読み込まれる。このように、Aufs でマウントしたディレクトリのアップデートを行うと、アップデータから見ると仮想ディスクをアップデートしたように見える。

Aufs でマウントしたディレクトリに対してファイ

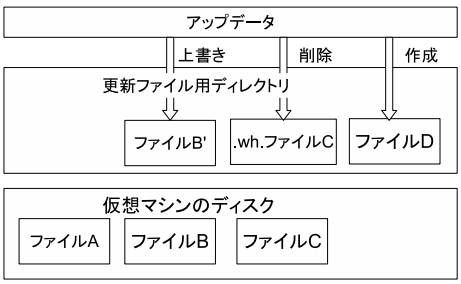

図 3　ファイルの抽出

ルやディレクトリの削除を行うと、更新ファイルの保存用ディレクトリに whiteout という特殊なファイルが作成される。whiteout ファイルはそのブランチより下のブランチのファイルが削除されたことを示し、削除されたファイル名の先頭に.wh. がついたファイル名になる。Aufs でマウントしたディレクトリにアクセスすると保存用のディレクトリが優先されて whiteout ファイルが読み込まれ、その下のブランチにあるファイルやディレクトリは削除されたように見える。削除を行う場合も仮想ディスクに変更を加えないので整合性がとれなくなることはない。

このようにして Aufs を用いて更新や削除を行うと、保存用ディレクトリに更新されたファイルやディレクトリが抽出される。更新されたファイルのディレクトリも自動的に作成されている。削除されたファイルについては、保存用ディレクトリ全体を探索して whiteout ファイルのパスを取得し、削除ファイルのリストを作る。

3.4　エミュレーション結果の反映

仮想マシンがレジュームされた直後にオフライン時に行われたファイルの更新を仮想ディスクに反映する。図 4 にエミュレーション結果の反映を示す。まず、更新されたファイルが格納されている保存用ディレクトリのアーカイブを tar コマンドで作成し、仮想ネットワーク経由で仮想マシンに送る。そして、ssh コマンドを使って仮想マシン上で tar コマンドを実行することで、保存用ディレクトリを仮想ディスクに展開する。削除されたファイルに関しては、削除ファイルのリストも仮想マシンに転送し、そのリストに基

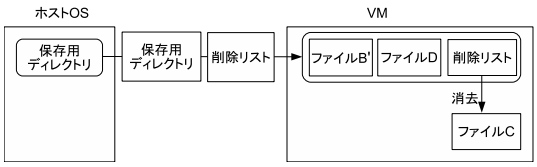

図 4　エミュレーション結果の反映

づいて仮想ディスク上のファイルを削除する。同時に whiteout ファイルも削除する。このように、仮想マシンの中でファイルを更新するため、仮想ディスクの不整合は起こらない。

アップデータが前処理や後処理を行う場合は、保存用ディレクトリを展開する前後で実行する。例えば、Ubuntu のパッケージには前処理と後処理を行うスクリプトが含まれている場合がある。この場合、前処理では設定ファイルの保存やサーバの停止などを行い、後処理では新しい設定ファイルの作成やサーバの再起動などを行う。これらの処理の大部分はオフラインでもエミュレーション可能であるが、前処理と後処理を行うアップデータへの対応については今後の課題である。

エミュレーション結果を反映している間に外部からの攻撃を受けないようにするために、反映が完了するまでは仮想マシンを外部ネットワークには接続しない。エミュレーション結果の反映にかかる時間はアップデータを一から実行する場合と比べて短いため、レジューム後に外部と通信できない時間は許容範囲内だと考えられる。ただし、保存用ディレクトリのアーカイブを仮想マシンに送るために仮想マシンをネットワークに接続する必要があるため、物理マシン内部での通信は許可する。

4　実験

OUassister の動作確認および、レジューム後にアップデートを行う場合と OUassister を使用したときのオンライン時の処理時間の比較を行った。実験に使用したマシンは、CPU Intel Core2 Quad 2.83GHz、メモリ 4GB であった。Xen 3.4.0 を用い、ドメイン 0 で Linux 2.6.32.25、ドメイン U で Linux 2.6.27.24 を動作させた。どちらにも Ubuntu10.04 を用いた。

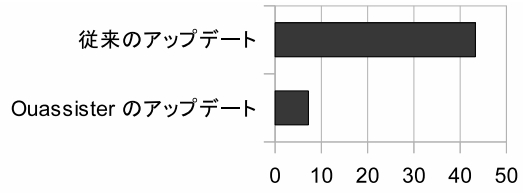

図 5　処理時間の比較

4.1　動作確認

apt コマンドを使用し、OUassister を用いてパッケージをインストールできるかどうかの確認を行ったインストールするパッケージには C コンパイラの bcc を用いた。その結果、仮想マシンをサスペンドした状態で bcc 関連のパッケージのインストールを行うことができ、パッケージデータベースが正しく更新されていることを確認することができた。

4.2　オンライン処理時間の比較

従来のアップデートと OUassister を用いたオフラインアップデートについて、仮想マシンをレジュームした後のオンライン処理時間の比較を行った。従来のアップデートはオンラインでアップデータのダウンロードとアップデート処理を行い、OUassister はエミュレーション結果の反映のみをオンラインで行う。この実験でも bcc パッケージのインストールを行った。測定の結果、図 5 のように OUassister によるオフラインアップデートが 7.2 秒、従来のアップデートが 43.3 秒となった。OUassister を用いた場合のオンライン処理時間の方が大幅に短いことが分かる。OUassister ではダウンロードやパッケージの処理などをオフライン時に行うことによって、オンライン時の処理時間を削減することができている。

5　関連研究

NetChk Protect [4] は、オフラインの仮想マシンにディスクにアップデータを書き込んでおき、オンラインになった時にそのアップデータを実行しアップデートを行う。仮想マシンの起動後にアップデータの実行を行わなければならないが、アップデータのダウンロードはオフライン時に行うことができるためアップデート時間を短縮できる。また、オンライン時にアップデータをダウンロードするためのネットワーク接続を必要としないので、外部との通信を遮断しておくことでアップデート中の攻撃を防ぐことができる。しかし、オフライン時に仮想ディスクに書き込みを行うため、サスペンドした仮想マシンには適用できない。

Nüwa [5] はアップデータの前処理と後処理をオフラインで実行できるように書き換えることで、アップデータをオフラインの仮想マシンに対して実行することができる。Nüwa はアップデータに含まれるスクリプトを分析し、スクリプトをオフライン時に実行可能なものと不可能なものに分ける。実行できないスクリプトもオフライン時に実行できるように書き換えてできる限りに実行する。書き換えが不可能だった時はオンライン時に実行する。しかし、オフライン時に仮想マシンのディスクの更新を行うため、サスペンド状態の仮想マシンには対応していない。

Offline Virtual Machine Servicing Tool [1] はオフラインの仮想マシンをアップデート専用環境で動かしてアップデートを行う。アップデート専用環境では更新プログラム管理サーバを動かし、新しいアップデータが発行されるとダウンロードしておく。オフラインの仮想マシンをアップデートする時にはアップデート専用環境にいどうして再開し、更新プログラム管理サーバから安全にアップデータをダウンロードしてアップデートを行う。アップデートが完了したら仮想マシンを再び停止させて、元のマシンに戻す。この手法はサスペンド状態の仮想マシンにも適用できるが、アップデート専用環境が必要となる上、アップデート中に仮想マシンから外部へのネットワーク接続が切断されてしまうなどの問題が発生する場合がある。

6　まとめ

本論文では、サスペンドした仮想マシンのオフラインアップデートを可能にする OUassister を提案した。OUassister は、オフライン時に VM shadow によってアップデータ実行のエミュレーションを行い、Aufs を使って更新ファイルを抽出しておく。仮想マ

シンのレジューム後に仮想ディスクに対してエミュレーション結果の反映を行うことで、オンライン時における仮想マシンのアップデートを短時間で完了させる。オフライン時には仮想マシンのディスクへの書き込みを行わないため仮想ディスクの整合性を保ちつつ、オフラインアップデートを行うことができる。OUassister を用いて Ubuntu のパッケージのオフラインアップデートを行えることを確認した。

今後の課題として、アップデータに含まれる前処理と後処理のスクリプトのエミュレーションが挙げられる。スクリプトもできるだけオフラインで実行し、実行できない部分だけを記録しておいて、レジューム後に反映させるようにする。

謝辞

本研究の一部は、科学技術振興機構 戦略的創造研究推進事業 (CREST) 研究領域「実用化を目指した組み込みシステム用ディペンダブル・オペレーティングシステム」による。

参考文献

[1] Microsoft. Offline Virtual Machine Servicing Tool.
[2] J. R. Okajima. Aufs3 – advanced multi layered unification filesystem version 3.x. http://aufs.sf.net.
[3] J. Pendry and M. McKusick. Union Mounts in 4.4BSD-Lite. In *In Proceedings of USENIX Technical Conference*, pp. 25–33, 2011.
[4] VMware, Inc. NetChk Protect.
[5] W. Zhou, P. Ning, X. Zhang, G. Ammons, R. Wang, and V. Bala. Always Up-to-Date:Scalable Offline Patching of VM Images in a Compute Cloud. In *In Proceedings of the 26th Annual Computer Security Applications Conference*, 2010.
[6] 飯田貴大、光来健一. VM Shadow: 既存 IDS をオフロードするための実行環境. 情報処理学会第 119 回 OS 研究会, 2011.