社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－CSEC－42（19）
2008 / 7 / 24

Code injection 検出のためのVMM スナップショット機能の強化

安藤類央 門林雄基 篠田陽一

独立行政法人　情報通信研究機構 情報通信セキュリティ研究センター
〒 184-8795　東京都小金井市貫井北町 4-2-1

あらまし
VMM(仮想マシンモニタ）は、従来のダンプやデバッグツールと比較して、対象となる OS、仮想マシンの外部観測を容易にすることを可能にした。コードインジェクション攻撃は、ユーザの実行するプログラムに、任意の悪意のあるコードを挿入することで、意図せざる結果や影響をシステムに及ぼす手法である。コードインジェクション攻撃のプログラムは、汎用のウィルススキャナでのシグニチャの解析を用いて行うことが難しいことが多く、プロアクティブと呼ばれるような振る舞い検知が行われる場合が多い。本論文では、API フックなどを行うイベント検出モジュールに、VMM へパラメータの受け渡しを行う通知関数を追加することで、コードインジェクションが行われた際に、即時的に攻撃コードのあるメモリの部分のみのスナップショットを取得するためのシステムを提案する。

An enhancement of VMM snapshot utility for detecting code injection

Ruo Ando, Youki Kadobayashi, Youichi Shinoda

National Institute of Information and Communication Technology, Tracable Network Group
4-2-1 Nukui-Kitamachi, Koganei,
Tokyo 184-8795 Japan
ruo@nict.go.jp

Abstract VMM (Virtual Machine Monitor) provide a more fine grained external observability of virutal machine compared with previous operating system and debug tool inside operating system. In this paper we propose an enhancement of snapshot of VMM for detecting code injection attacks. In code injection attacks, attacker inserts arbitrary byte which causes bad effects and result for target system. For detecting code injection attacks, it is hard to apply singnature matching. Instead, behavior based detection such as proactive is applied. In propsed system, notification routine for VMM is inserted to API hool module in virtualized host OS. By doing this, we can take a snapshot of part of memory attacked (infected) just when code injection is occurred.

1　はじめに

近年のプロセッサ性能の急速な向上は、複数の OS を同時に稼動させる仮想化技術の実用性を高めた。特に、仮想マシン、仮想マシンモニタなどのシステムは、従来のデバッガやダンプツールと比較して、OS やプロセスの外部観測を容易にした。この仮想化技術が提供する外部観測性は、デバッグやソフトウェアのチェック、特に VMM（仮想マシンモニタ）に関してはセキュリティ機能の実装に使われることが多くなった。一方、最近のセキュリティインシデントを引き起こす手法のトレンドの 1 つとして、コードインジェクションを用いたものがある。これは、受動的攻撃と同様に、ユーザの実行によって発生し、実行したプログラムによって意図しない結果や影響

をシステムに及ぼすというものであるが、ユーザが実行するプログラムに、任意のコードを埋め込むことで、より実行率を高めるというものである。コードインジェクション攻撃には、WEB のクロスサイトスクリプティング、SQL（構造化照会言語）インジェクション、そして、常駐型プロセスへの感染などがある。本論文では、仮想マシンモニタのスナップ機能の強化による Windows OS のプロセスへのコードインジェクションの検出と検査システムを提案する。スナップショット機能の強化は、Windows 上でのイベント検出モジュールの実装と、VMM でのスナップショット機能の修正からなる。Windwos へのイベント機能モジュールを作成し、イベント検出時に CPU コンテキストなどを用いた VMM のパラメータ引渡しとスナップショットのトリガを行うルーチンを追加する。これにより、コードインジェクション攻撃を検出し、攻撃によって感染したメモリの部分のスナップショットをとることが可能になった。実装対象としては、VMM を 2 種類に分類し、VMM のタイプ 1（KVM）とタイプ 2（XEN）のについて議論する。

2 コードインジェクション

2.1 コードインジェクション攻撃

コードインジェション攻撃とは、攻撃の対象に任意のコードを挿入し、対象がサービスであれば意図しない出力やアプリケーションであれば有害な振る舞いを引き起こすものである。コードインジェクションの対象となるものは、WEB アプリケーション、SQL(構造化照会言語)、OS のプロセスなど、非常に広いが、狭義には、コードインジェクションは埋め込まれたコードの瞬間的な実行を利用したものである。コードの挿入には、脆弱性を利用するもの、受動的攻撃を利用するものがある。また、二次的なコードインジェクション攻撃の存在も指摘されており、これは、埋め込まれたコードがユーザよって実行され、検出、除去が行われるまで記憶装置の中にコードが常駐することによって、攻撃の対象となったアプリケーションだけでなく、それと相互作用するプログラムも標的となるというものである。

2.2 Windows でのコードインジェクション

本論文では、仮想マシンモニタのセキュリティ機能拡張の適用として、Windows OS 上のプロセスへのコードインジェクションの検出とダンプ、解析を扱う。Windows OS 上のコードインジェクションとは、他のプロセスメモリ空間を書き換えることで、そのプロセスに任意のコードを実行させるものである。この手法は、常駐型のプロセスに適用されることが多く、悪意のあるプログラムがその存在を隠蔽するために利用される。Windows OS 上のコードインジェクションには、DLL を実行させるものと、コードそのものを実行させるものの 2 種類がある。両手法とも、他のプロセスメモリ空間に WRITE 処理が行われた時をフックし、VMM にプロセッサのコントロールを渡すことで、検出とスナップショットの入手を行うことができる。

2.3 コードインジェクションの検出

コードインジェクションを行うプログラムの検出は、汎用のウィルススキャナを用いて行うことは難しく、つまりシグニチャのマッチングを用いて行うことは難しい。そのためプロアクティブディフェンスと呼ばれるようなビヘイビアベースの解析を用いて検出を行うことが多い。提案システムでは、プロセスのハンドラを取得→メモリアロケーション→プロセスメモリ書き込み→リモートスレッド実行など、API のシーケンスの解析を用いてコードインジェクションの検出を行った。

3 提案システム

本節では、コードインジェクション検出時のスナップショット取得のための、提案システムの大枠について述べる。対象となる仮想マシンモニタについては、1 つには XEN 型（ブートローダー型）と KVM 型（デバイスドライバ型）の 2 種類に分類することができる。

3.1 仮想マシンモニタ　タイプ 1

仮想マシンモニタタイプ 1 は、ホスト OS をハイパーバイザーとして用いるもので、このタイプには

KVM がある。この方式では、ホスト OS のカーネル空間内にあるゲスト OS の仮想メモリを、ユーザ空間あるいはカーネル空間に移動する。この方式では、QEMU のインターフェイスを使うため、スナップショットの強化には、QEMU のインターフェイスを使い、ユーザ空間から、特定のファイルにダンプする方法と、VMM モジュール内のゲスト OS の仮想メモリの一部を、ホスト OS のカーネル空間などにマップする方法の 2 通りがある。

3.2 仮想マシンモニタ タイプ 2

仮想マシンモニタタイプ 2 は、独自にブートローダとハイパーバイザーを用意するもので、このタイプには、XEN がある。この場合、準仮想化と完全仮想化に分かれるが、本論文では、Windows の完全仮想化を扱っているんで、スナップショットの取得には、XEN のインターフェイスではなく、QEMU のインターフェイスを用いることになる。そのため、スナップショット機能の強化には、QEMU のモニターインターフェイスの修正か、VMM (XEN) 内のゲスト OS の仮想メモリの一部を、ホスト OS にマップする方法の 2 通りがある。この際には、メモリ転送のインターフェイスである Grant Table で使われている共有メモリを扱う関数を適用されると想定される。

3.3 検出とスナップショットの取得

本論文で扱うコードインジェクション攻撃は、一時的 (瞬間的なコード実行) であるため、ホスト OS でインシデントが検出された場合、即時にパラーメータを VMM に渡してスナップショットを取る必要がある。検出については前節で述べたが、コードインジェクションが行われたメモリのレンジを指定してスナップショットを取る必要があるため、これらの数値は、レジスタに保存し、VMM が持っている CPU のコンテキストを渡すことで、パラメータの受け渡しを行う。VMM タイプ 1 の場合、ホスト OS への CPU のコンテキストからのパラメータの受け渡しは、ホスト OS がハイパーバイザーになっているため比較的容易であるが、VMM のタイプ 2 の場合、VMM からホスト OS への情報伝達については、別途チャンネルを設置する必要がある。

4 実装方法

4.1 Windows API フック

4.1.1 DLL のマッピング

以上のような開発の要請から、DLL (独自 API) を他のプロセスに任意のタイミングで実行させる DLL Injection という技術が用いられることがある。DLL Injection の方法には、Mi-crosoft Windows が提供している機能を使うもの、デバッグ機構の機能を使うもの、リモートスレッドを使うもの、そしてモジュールのインポートセクションの変更によるものなどがある。本論文では、モジュールのインポートセクションの変更による方法により、P2P ソフトウェアのデータ送受信関数をフックし、ここに適切な操作を入れることにより、ソフトウェアの挙動を追跡し、問題となるトラフィックが発生または到着する前に防止を行うことを可能にした。図 2 は、インポートテーブルの変更よる DLL 注入を示したものである。インポートテーブル (インポートセクション) とは、セクションテーブルの中にあるデータ構造体 (ヘッダ) で、プログラムが実行される前に必要な DLL のアドレスと、利用する DLL 内のシンボルのアドレスのリストが格納されている。ここで、フックしたい DLL のアドレスを、任意の DLL へのアドレスに書き換えることで、ソフトウェアの動作の修正を行うことができる。この方法は、特定の CPU の仕様に依存しなく、またスレッドの同期の問題もないため、非常に柔軟な方法として用いられることが多い。処理としては、1) 注入したい DLL を用意

する。2) 対象となるソフトウェアのインポートテーブルアドレスを取得する。3) フックしたい関数のアドレスを探す。4) 発見したアドレスを、注入したい DLL (関数) へのアドレスに書き換える。関数形は

```
void ReplaceIATTableInP2Psoftware
("kernel32.dll",
funcORG,
funcINSERT",
moduleHandler);
```

対象とするプロセスの構造などにより、実装方法はいくつか存在するが、大枠は以上で示した通りである。

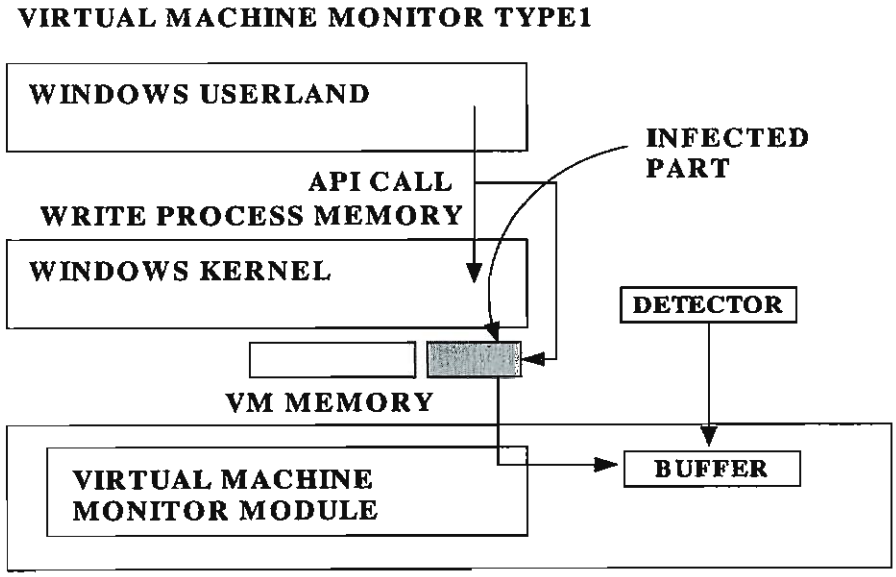

図 1: 通常の TDE と提案システムの TDE の比較。左側は、ケーパビリティを設定し、それを変更する。PERMIT を明示的に設定する。提案では、TDE が実行されたとき、アクセス権 DENY を明示的に設定する。

4.1.2 適用 API とアドレスの検索

提案システムでは、GetModuleHandle などを使って、他プロセスのハンドラを取得する。他プロセスのアドレス空間内に（共有）メモリをアロケートする方法は、VirtualAllocEX などがある。実際にリモートプロセスメモリには、WriteProcess Memory を使ってデータを書き込む。アドレスの検索は、GetModuleHandle を使って、プロセスのハンドラ（アドレス）を取得し、Microsoft Windows の実行ファイル形式を参照しながら、インポートセクション内のフックしたい関数のアドレスを探す。以下にこの概略を示す。

```
pointer=GetModuleHandlecurrentPfn
=GetProcAddress
/*pointer を移動して、インポートセクションを探す*/
/*インポートセクション内*/
if(Npointer==CurrentPfn)?
/*発見*/
```

ただし、対象となるプロセスが特定の Load-Library や createProcess の使い方をしている場合、上のコードに追加操作を加える必要がある。関数をフックする場合、プロセスメモリのスナップショットを取得し、フィルタの動作に移行することができる。

4.2 スナップショットモジュールの修正

提案システムでは、コードインジェクションが行われて、変更が起こったメモリの部分だけのスナップショットを取得する必要があるが、この場合、QEMU のインターフェイス部分を修正する方法と、VMM が持っているゲスト OS の仮想メモリを、ホスト OS（コントロールドメイン）にマップする方法の 2 通りがある。KVM の場合、メモリマップ関数を用いる。XEN の場合、Grant Table のような VM 間でのメモリ転送のインターフェイス内部の関数を用いることができる。両方の手法においても、コピーオンライトの手法が適用することが有効である。

5 まとめと今後の課題

VMM(仮想マシンモニタ）は、ハードウェアと OS の間に位置するため、提供するスナップショットやチェックポインタなどによって、従来の OS の内部

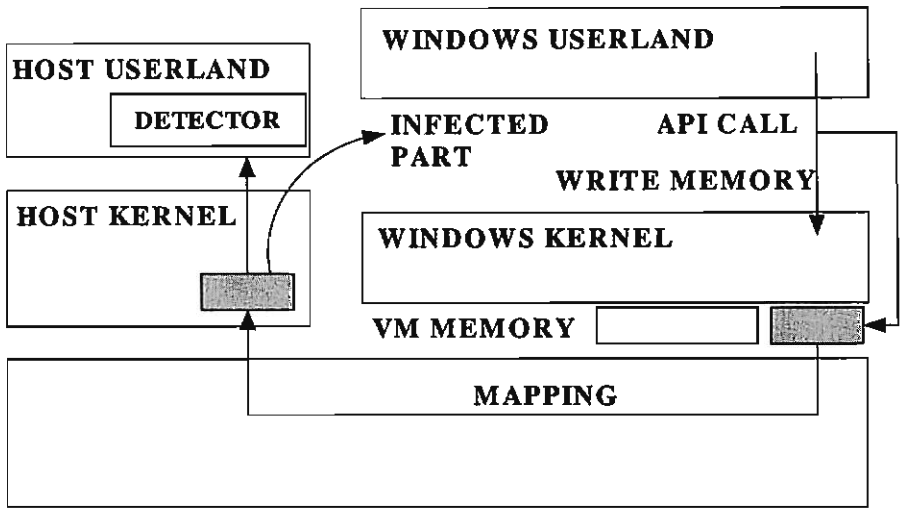

図 2: 通常の TDE と提案システムの TDE の比較。左側は、ケーパビリティを設定し、それを変更する。PERMIT を明示的に設定する。提案では、TDE が実行されたとき、アクセス権 DENY を明示的に設定する。

でうごくダンプやデバッグツールと比較して、扱っている仮想マシンの外部観測を容易にすることを可能にした。本論文では、コードインジェクション検出のための仮想マシンモニタのスナップショット機能の強化方法を提案した。コードインジェクション攻撃は、ユーザの実行するプログラムに、任意の悪意のあるコードを挿入することで、意図せざる結果や影響をシステムに及ぼす手法である。コードインジェクション攻撃のプログラムは、汎用のウィルススキャナでのシグニチャの解析を用いて行うことが難しいことが多く、プロアクティブと呼ばれるような振る舞い検知が行われる場合が多い。本論文では、API フックなどを行うイベント検出モジュールに、VMM へパラメータの受け渡しを行う通知関数を追加することで、コードインジェクションが行われた際に、即時的に攻撃コードのあるメモリの部分のみのスナップショットを取得するためのシステムを提案した。今後の課題としては、ゲストＯＳからＶＭＭへ情報通知法の洗練などが挙げられると考えられる。

参考文献

[1] Greg Goth, "Virtualization: Old Technology Offers Huge New Potential," IEEE Distributed Systems Online, vol. 8, no. 2, 2007

[2] Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, Andrew H. Mason, Clifford E. Kahn, "A Retrospective on the VAX VMM Security Kernel", IEEE Trans. Software Eng. 17(11): 1147-1165, 1991

[3] Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, Andrew H. Mason, Clifford E. Kahn, "A Retrospective on the VAX VMM Security Kernel", IEEE Trans. Software Eng. 17(11): 1147-1165, 1991

[4] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In Proceedings of the 19th Symposium on Operating System Principles(SOSP 2003), Bolton Landing, NY, October 2003.

[5] Ruo Ando, Youki Kadobayashi, "Improving VMM based IPS for real-time snapshot and nullification of buffer overflow exploitation" The 1st Joint Workshop on Information Security September, 20-21, Sookmyung Women's University, Korea

[6] A Virtual Machine Introspection Based Architecture for Intrusion Detection Tal Garfinkel and Mendel Rosenblum In the Internet Society's 2003 Symposium on Network and Distributed System Security (NDSS), pages 191–206, February 2003.

[7] Nguyen Anh Quynh, Ruo Ando, and Yoshiyasu Takefuji : "Centralized Security Policy Support for Virtual Machine", USENIX, 20th Large Installation System Administration Conference, December 2006.

[8] Reiner Sailer and Trent Jaeger and Enriquillo Valdez and Ramon Caceres and Ronald Perez and Stefan Berger and John L. Griffin and Leendert van Doorn, "Building a MAC-Based Security Architecture for the Xen Opensource Hypervisor", in Proceedings of the 2005 Annual Computer Security Applications Conference (ACSAC), December 2005.

[9] ReTrace: Collecting Execution Trace with Virtual Machine Deterministic Replay, Min Xu, Vyacheslav Malyugin, Jeffrey Sheldon, Ganesh Venkitachalam and Boris Weissman, MoBS2007, June 2007.

[10] Sanjay Bhansali, Wen-Ke Chen, Stuart De Jong, Andrew Edwards, and Milenko Drinic, "Framework for Instruction-level Tracing and Analysis of Programs", Second International Conference on Virtual Execution Environments (VEE06), June 2006.

[11] George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza Basrai, and Peter M. Chen. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. In Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002.

[12] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, Jacob R. Lorch, "SubVirt: Implementing malware with virtual machines", in Proc. IEEE Symp. on Security and Privacy (the Oakland Conference), May 2006.