Vol.2009-SE-164 No.29
Vol.2009-EMB-13 No.29
Vol.2009-CSEC-45 No.29
2009/5/29

Memento Project: 仮想化技術を用いたOS内部観測による解析用データセットの構築と公開

安藤　類央[†1]　　門林雄基[†1]　　篠田陽一[†1]

最近のマルウェア補足技術や仮想化技術の急速な発展により、セキュリティログが大量に収録できるようになった。これに併行して、セキュリティデータの格納管理、検索、分類手法の需要が高まっている。本論文では、情報通信研究機構で行っている、高粒度な観測データ、セキュリティインシデントのデータの生成のためのシステム設計とプローブ技術、そしてデータのフィルタリングと解析技術について述べる。現在、複雑化、高度化するセキュリティインシデントの解析と対応のためには、より高粒度なデータ取得と、多様な解析手法が必要であるが、両者の連携方法を統合的に評価するためのデータセットを提供する。これにより、先鋭化するセキュリティインシデントの解析と対応の方法の開発と評価が可能になる。

Memento Project: Information security dataset generation and filtering using virtual machine introspection

Ruo Ando [†1] Youki Kadobayashi [†1]
and Yoichi Shinoda [†1]

With the rapid advance of malware capturing technology, networked systems is able to obtain a large amount of logs. At the same time, storing, searching and classifying techniques for coping with a large size of logs. In this paper we propose a project called memento undergoing in National Institute of Information and Communication Technology. Memento project aims to deal with fine-grained observed data, probe design for security incident and data filtering and its analysis. For taking some measurement for complicated and sophisticated security incident, more fine-grained logging and a diversity of analysis techniques are necessary. Memento project is going to provide integrated environment and dataset for evaluating both probe and analyzer. We hope that the analysis and response of security incident will become more sophisticated by the contribution of Memento project.

1. はじめに

近年、ハニーポットやハニーネットなどのマルウェア補足技術や、ハイパーバイザーや仮想マシンなどのハードウェア仮想化技術により、Microsoft Windows などの OS 上で挙動するマルウェアのより大量のログを取得することが可能になった。しかしながら、マルウェアの種類は増加の一途を辿っており、ログ取得技術の発達に併行して、これらのデータをどのように管理格納し、実際の検出に利用するかに関しての方法論は確立されていない。また、ハードウェア仮想化技術、またはデバイスドライバ開発技術の発達により、より高粒度なログを取得することが可能になった。そのため、今後のマルウェア検出、挙動解析のためには、より高粒度なログを検索、利用する必要がある。本論文では、 仮想化技術を用いたOS 内部観測による解析用データセットの構築と公開について、ログ生成のためのシステム設計と、データセットの生成と適用アルゴリズムについて述べる。

2. ログ生成のためのシステム設計

2.1 VM Instrospection

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。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。また、完全仮想化モードは、プロテクトモード以来のプロセッサ構造の革新と指摘されており、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の観点から、注目されている。本論文では、仮想化された Windows OS のメモリ挙動モニタと Libvirt によるゼロデイ攻撃の検出システムの構築を提案する。Windows OS の仮想メモリへのアクセスを行う Native API をフックし、挙動をモニタすることで異常検知を行う。検出時に攻撃あるいは感染を受けたメモリのレンジを仮想マシンモニタに通知し、Libvirt を用いることで当該箇所のスナップショットを取得する。提案手法により、ホスト Windows OS のパフォーマンスにインパクトを与えることなく、検出と

†1 情報通信研究機構〒 184-8795 東京都小金井市貫井北町４－２－１

© 2009 Information Processing Society of Japan

Vol.2009-SE-164 No.29
Vol.2009-EMB-13 No.29
Vol.2009-CSEC-45 No.29
2009/5/29

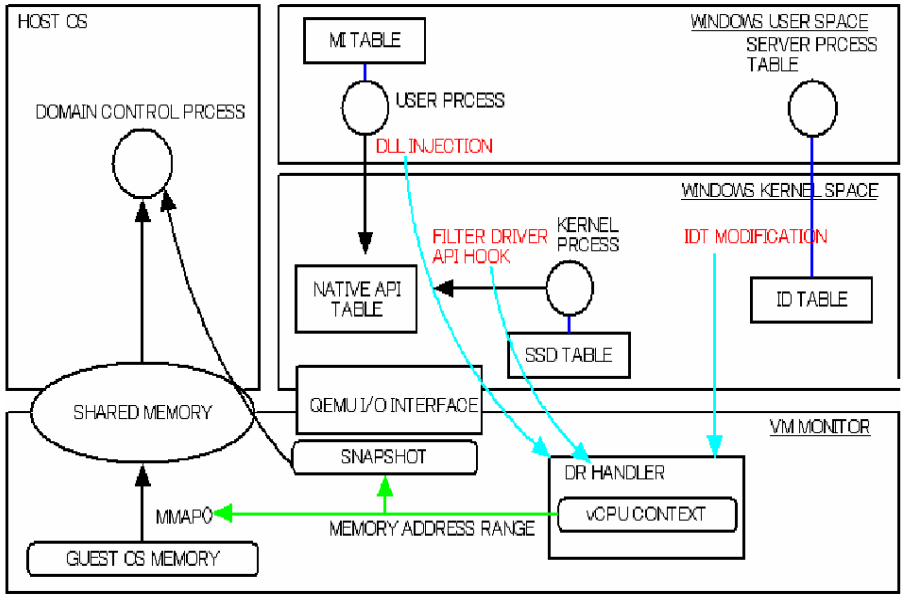


図 1 VM Introspection を用いたデータログの生成. レジスタや共有メモリをインターフェイスにして情報をホスト OS 側に転送する。

解析を行うことが可能になる。

2.2 Windows OS の修正

2.2.1 DLL Injection

DLL Injection とは、任意の関数をライブラリ化して特定の API が発行されたときに、実行させるもので、既存のアプリケーションに新しい機能を加えたいとき、ソフトウェアのデバッグの際に用いられる。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、対象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 1 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。

2.2.2 Native API Hook

Native API Hook は、カーネルモードでのイベント検出に用いられる。ユーザモードでの API 発行は、ntdll.dll 経由でカーネル空間に伝達される。Windows では、カーネルモードで動作する API を、native API といい、これらは、SystemServiceDescriptorTable という構造体によって、制御されている。そのため、Native API Hook を行うためには、SystemServiceDescriptor に修正を加える必要がある。

2.2.3 IDT Modification

IDT (Interruption Descriptor Table) の修正と IP(命令ポインタ) の追跡のために、本論文ではプロセス構造ルーチンを用いてロード時に登録されるコールバック関数を通じて上の操作を行う。ここで、プロセス構造ルーチンとは、実行ファイルがメモリにロードされプロセスとなる際に、コールバック関数やブレークポイントを挿入するために用いるドライバ関数のことを指す。図 2 は、コールバック関数内での操作を示したものである。提案システムでは、プロセス構造ルーチンを用いて、実行ファイルのロード時にコールバック関数を登録する。Win32 のドライバ関数を用いると、イメージが実行時にロードされるときに呼び出

© 2009 Information Processing Society of Japan

Vol.2009-SE-164 No.29
Vol.2009-EMB-13 No.29
Vol.2009-CSEC-45 No.29
2009/5/29

されるコールバック関数を登録することができる。

```
void LoadImageNotifyRoutine{
PUNICODE_STRING FullImageName,
HANDLE ProcessId,
PIMAGE_INFO ImageInfo
}
```

図２は、プロセス構造コールバックルーチンを用いたブレークポイントの挿入と IDT の修正を示したものである。NTFS 上の実行ファイルがロードされるとき、LoadImageNotify-Routine が呼び出される、この関数の中で、sys ファイルのエントリポイントへのデバッグポイントの挿入と、例外ハンドラの修正が行われる。

3. データセットの生成と適用アルゴリズム

前節で述べた PROBE システムから生成されたデータに対して解析処理を行うには、次元数やデータ数の削減などのデータの前処理を行う必要がある。また、検出に使われるクラスタリングやクラシフィケーションなどのアルゴリズムを適用するために、データに適切なラベリングやプリプロセスを行う必要がある。

3.1 疎な行列の扱い

Memento システムによって生成されるデータは、ひとつには API のインターセプトによって追記される。そのため、１つの API の呼び出しによって行列が縦方向に伸びる形式のデータ取得を行った場合、生成されるデータの行列が疎になる場合がある。また、Windows OS の場合、アクセスログを構成するデータ長が長い場合があり、このような sparse かつ大きなサイズのデータを扱うために、データ長の削減や次元数の削減を行う必要がある。

3.1.1 データ長の削減

データ長の削減には、ハッシュ関数を用いるが、通常のハッシュ関数では対象データに１バイト程度の違いが生じても結果として出力されるハッシュ値が大きく変わる。提案システムでは、Locality Sensitive Hashing を持ちいて同系統のデータを、類似性を保ちつつハッシュをとる方式を採用している。

3.1.2 データ数、次元数の削減

前節で述べたように、提案システムから取得されるデータは疎なものが多いため、データ数や次元数の削減をする必要がある。現在、PCH（主成分分析）や SVM（サポートベクトルマシン）を用いてデータ数と次元数を削減する方法を検討中である。

3.2 データの表現と可視化

解析手法としては、主に機械学習系のデータを用いる。現在は、行列演算を行う自己組織化マップとサポートベクトルマシンを検討中であり、クラスタリングと可視化を行うには自己組織化マップを、クラシフィケーションを行うにはサポートベクトルマシンを検討中である。

3.2.1 データフォーマット

提案システムでは、生成されたデータに対して、クラスタリングとクラシフィケーションの２つの処理を行う。クラスタリングは自己組織化マップ、クラシフィケーションは、サポートベクトルマシンの２つのアルゴリズムを準備中である。図２は、公開予定のデータフォーマットを示したものである。どちらも API が発行されるたびに、呼び出し時の引数を抽出して行列を生成する。ラベリングあるいは訓練データの生成のために、一定時間ごとに通常、異常、もしくは悪意のある振る舞いを Windows OS 上で実行し、サンプリングあるいはラベルノイズ削減処理を行う。

3.2.2 自己組織化マップ

自己組織化マップは、kohonen(9) が提案した手法で、クラスタリングの手法の１つである。k-means などの他のクラスタリングの手法と違い、クラスタ数の決定を与件としないところに特徴がある。

3.2.3 サポートベクトルマシン

サポートベクトルマシン (8) は、機械学習の手法の１つで、教師付きのデータ分類アルゴリズムである。マージン最大化を用いたノンパラメトリックの手法を用いることで、他のニューラルネットワークの手法より汎化能力に優れるとされる。

4. まとめと今後の予定

最近のマルウェア補足技術や仮想化技術の急速な発展により、セキュリティログが大量に収録できるようになった。これに併行して、セキュリティデータの格納管理、検索、分類手法の需要が高まっている。本論文では、情報通信研究機構で行っている、高粒度な観測データ、セキュリティインシデントのデータの生成のためのシステム設計とプローブ技術、そしてデータのフィルタリングと解析技術について述べる。現在、複雑化、高度化するセキュリティインシデントの解析と対応のためには、より高粒度なデータ取得と、多様な解析手法が必要であるが、両者の連携方法を統合的に評価するためのデータセットを提供する。これにより、先鋭化するセキュリティインシデントの解析と対応の方法の開発と評価が可能になる。

© 2009 Information Processing Society of Japan

Vol.2009-SE-164 No.29
Vol.2009-EMB-13 No.29
Vol.2009-CSEC-45 No.29
2009/5/29

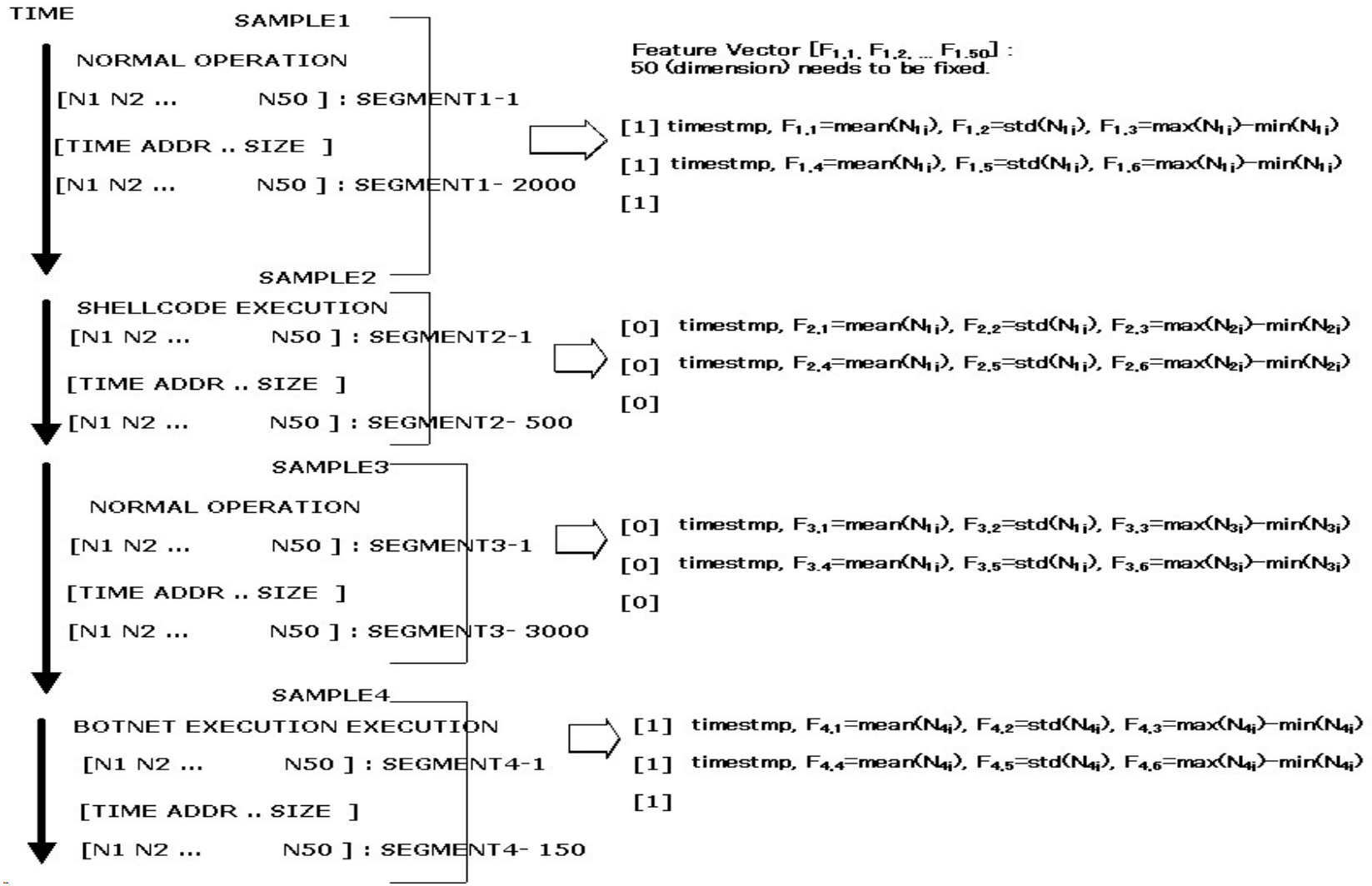


図 2 自己組織化マップ（分類、可視化）、サポートベクトルマシン（学習、検出）のためのデータセット。定性情報がなく、タイムスタンプがエンコーディングされている場合の形式は上図のようになる。

© 2009 Information Processing Society of Japan

Vol.2009-SE-164 No.29
Vol.2009-EMB-13 No.29
Vol.2009-CSEC-45 No.29
2009/5/29

name	API	attributes	hack method
index	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
index	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
ProcessHandle	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
BaseAddress	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
ZeroBits	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
CommitSize	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
SectionHandle	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
Win32Protect	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
AllocationType	zwMapviewofSection	memory access (kernel)	SDT interlockexchange
index	VirtualAlloc	memory access (user)	DLL injection setWindowsEx
index	VirtualAlloc	memory access (user)	DLL injection setWindowsEx
ProcessHandle	VirtualAlloc	memory access (user)	DLL injection setWindowsEx
dwSize	VirtualAlloc	memory access (user)	DLL injection setWindowsEx
flAllocationType	VirtualAlloc	memory access (user)	DLL injection setWindowsEx
flProtect	VirtualAlloc	memory access (user)	DLL injection setWindowsEx
index	pre−read	file access (NTFS)	filter manager
index	pre−read	file access (NTFS)	filter manager
IsDirectory	pre−read	file access (NTFS)	filter manager
AccessRecord	pre−read	file access (NTFS)	filter manager
AccessAttributes	pre−read	file access (NTFS)	filter manager
ProcessID	pre−read	file access (NTFS)	filter manager
ProcessAttributes	pre−read	file access (NTFS)	filter manager
index	pre−write	file access (NTFS)	filter manager
index	pre−write	file access (NTFS)	filter manager
IsDirectory	pre−write	file access (NTFS)	filter manager
AccessRecord	pre−write	file access (NTFS)	filter manager
AccessAttributes	pre−write	file access (NTFS)	filter manager
ProcessID	pre−write	file access (NTFS)	filter manager
ProcessAttributes	pre−write	file access (NTFS)	filter manager

図 3 データセットを作るための API インターセプトの情報。.

© 2009 Information Processing Society of Japan

Vol.2009-SE-164 No.29
Vol.2009-EMB-13 No.29
Vol.2009-CSEC-45 No.29
2009/5/29

参 考 文 献

1) Kernel Based Virtual Machine, http://kvm.qumranet.com/kvmwiki
2) XEN virtual machine monitor, http://www.cl.cam.ac.uk/Research/
3) Tal Garfinkel and Mendel Rosenblum, ”A Virtual Machine Introspection Based Architecture for Intrusion Detection ”, In the Internet Society ’s 2003 Symposium on Network and Distributed System Security (NDSS), pages 191.206, February 2003.
4) BD Payne, M Carbone, M Sharif, and W Lee. Lares, ”An Architecture for Secure Active Monitoring Using Virtualization ”, In Proceedings of the IEEE Symposium on Security and Privacy (Oakland 2008), May 2008.
5) BAN Tao, ANDO Ruo, KADOBAYASHI Youki, ”Monitoring and Analysis of Network Traffic in P2P Environment”, Journal of the National Institute of Information and Communications Technology, Vol.55 Numbers 2/3 2008, ISSN 1349-3205, 2008.11.
6) 安藤類央、門林雄基、篠田陽一、「Code injection 検出のための VMM スナップショット機能の強化」、情報処理学会コンピュータセキュリティ研究会第４２回研究報告２００８年7月２５日
7) Ruo Ando, Youki Kadobayashi, Youichi Shinoda, ”Incident-driven memory snapshot for full-virtualized OS using interruptive debugging techniques”, International Journal of Security and Its Applications, vol.2, No,3,pp41-48, 2008 July, ISSN 1738-9976
8) Deepak Ravichandran, Patrick Pantel, Eduard H. Hovy: Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. ACL 2005
9) T.Kohonen, Self-Organizing Maps, Springer, 1996
10) V.Vapnik, ”The Nature of Statistical Learning Theory”, Springer, 1995

© 2009 Information Processing Society of Japan