高速な VMI 機構を実装したバイナリ解析基盤
― FastVMIX / Fast VMI on Intel VT-X ―

1.　背景

マルウェアの解析や、ソースコードの与えられていないバイナリの解析など、リバースエンジニアリングはあらゆる分野において需要が高い技術である。特に情報セキュリティにおいては、マルウェアの解析をいち早く行い、攻撃手法や動作をいち早く解明する必要がある。マルウェアの解析には静的解析と動的解析の 2 種類の手法が存在する。静的解析はデコンパイラでアセンブリを読み、得られるコードグラフや文字列の列挙などから文字通り、マルウェアを動作させずして解析を行う。一方、動的解析はマルウェアを実際に動作させ、ネットワークアクセスをトレースしたり、システムファイルの改ざんや攻撃のトラフィックなどを解析したりする。静的解析に比べ動的解析はすばやく動作を把握することができるため初期対応としての解析手段としては重宝される傾向にある。しかし、マルウェアの中には動的解析耐性を持つものがあり、単なるデバッガでは解析を妨害されてしまう恐れがある。そうした中でハイパーバイザ（仮想マシンモニタ）を用いた解析システムが提案されている。特にベアメタルハイパーバイザは、ハードウェアの上で直接動作し、ゲストマシンとして実際の OS やアプリケーションを動作させることができ、ハードウェア仮想化支援技術を用いて、特権的にゲストのレジスタやメモリを読み書きすることができるため、ゲストから不可視の解析を行うためのプラットフォームとして近年注目されている。しかし、ハイパーバイザを用いた解析基盤には、ゲストマシンの解析によるオーバーヘッドやタイミング解析への対応の甘さなどの問題が存在する。

2.　目的

本プロジェクトでは、バイナリの動的解析プラットフォーム『FastVMIX』をベアメタルハイパーバイザの BitVisor をベースに開発することを目的とした。また機能として、高速なゲストの解析を行える機構、マルウェアの解析耐性機能の迂回機構を実装することも目的とした。

まず、高速なゲストの解析だが、これはコンテキストスイッチを極力抑えるシステムを構築する。コンテキストスイッチには、エミュレーションに必要なコンテキストスイッチ（図 1）と、解析時にゲストからハイパーバイザに制御が移るコンテキストスイッチの大きく 2 種類が存在する。本プロジェクトでは後者の解析時におけるコンテキストスイッチを極力減らすデザインで開発を進めた（図 2）。

次に、マルウェアの解析耐性機能の迂回機構だが、今回はタイミング解析と呼ばれるマルウェアの機能を迂回することを目標とした。タイミング解析は実環境と解析環境とで発生する実行時間の差異から解析環境であると断定する機能である。これを、CPU 時間の偽装という手法で迂回する機能を提供することとした。

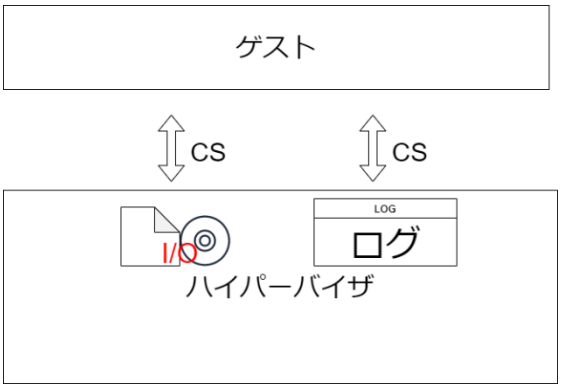

図 1. 通常のコンテキストスイッチ

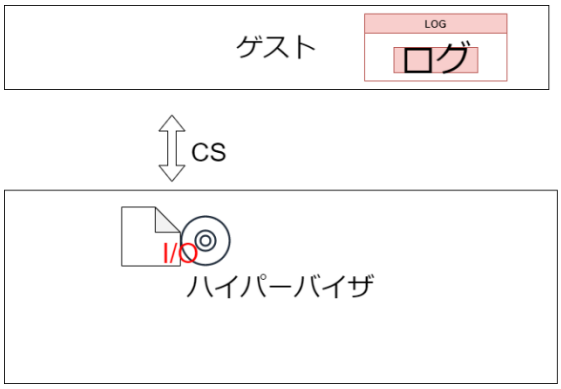

図 2. FastVMIX でのコンテキストスイッチ

3. 開発の内容

FastVMIX の全体像を図 3 に示す。FastVMIX を構成するのは BitVisor ベースのハイパーバイザと、ハイパーバイザを操作したりカーネル空間での準備を行ったりするカーネルモジュール、カーネルモジュールを操作するためのフロントエンドコマンド、特殊なメモリ空間 1GB Huge Page で動作する PIC バイナリの大きく 4 つで構成されている。

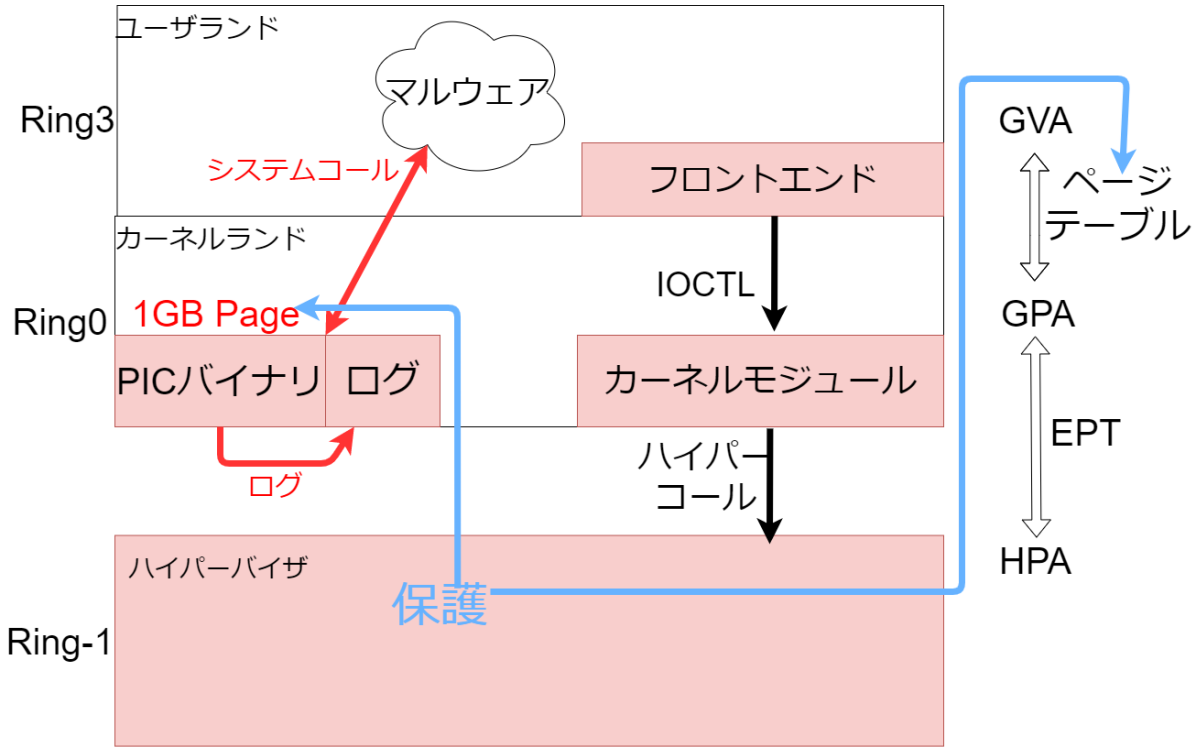

図 3. システムの全体図

FastVMIX は解析時におけるコンテキストスイッチを可能な限り廃したデザインになっている。これを実現するために、ゲストのセンシティブな動作をトラップし、ログを取る解析機構をゲスト内部に挿入した。システムのコンポーネントでは PIC バイナリにあたる。PIC バイナリはゲストの内部（Ring0）で動作する。すると、ゲスト内部（Ring3 or Ring0）に侵入したマルウェアがシステムへ攻撃を行うことができる。これを防ぐためにハイパーバイザがシステム全体を保護する。

保護の手法としてページテーブル保護とメモリマップの動的変更をハイパーバイザに実装した。

攻撃者はゲスト内部のページテーブルを操作することにより、システムの機能を停止させることができる。例えば PIC バイナリやログをアンマップしたりすることが可能である。そこで FastVMIX では PIC バイナリとログのページテーブルエントリを保護する。具体的な手法としてはページテーブルが切り替わるたびに、該当するエントリが書き換わっていないか監視する。例えば MOV-TO-CR3 な操作をハイパーバイザでトラップし、ページテーブルエントリを監視する。しかし、ページテーブルエントリの保護はプロセスが切り替わるたびに監視を行わなければならず、オーバーヘッドの要因となりうる。そこで 1GB Huge Page を採用し、PIC バイナリとログをこの特殊なページにマップすることでオーバーヘッドを解消した。ページテーブルは通常 4 レベルページングが採用されており、エントリが階層ごとに別れている。通常の 4KB もしくは 2MB ページのエントリを保護する場合、下のレベルだけでなく、自身より上のレベルまで保護してやる必要がありコストが増大する。しかし、1GB Huge Page ならば保護するレベルが 2 レベルで済む。したがって、1GB Huge Page の採用により保護するエントリ数の削減が達成できた。

次に、メモリマップの動的変更による保護について述べる。図 4 で示したように通常のメモリマップと解析用のメモリマップの 2 種類を用意し、通常は通常用のメモリマップ（解析コード PIC バイナリがアンマップ、ログは読み出しのみ）を使用し、解析時のみに解析用のメモリマップ（PIC バイナリがマップされ、ログが読み書き可能、カーネルのコード領域は読み込みのみ）を使用する。メモリマップにはゲストの物理メモリアドレスとホストの物理メモリアドレスの変換テーブルである EPT（Extended Page Table）を用いた。また、メモリマップの動的変更には EPTP Switching を用いて、コンテキストスイッチなしでメモリマップの動的変更が行えるようにした。

最後に、CPU 時間の偽装についてだが、RDTSC 命令をトラップし、解析にかかった時間分もとに戻す実装をハイパーバイザに行った。ベンチマークプログラムではこれが有効に働いていることが確認できた。

4. 従来の技術（または機能）との相違

これまで 1GB Huge Page で解析機能を動作させるシステムは存在しなかった。またこれによりページテーブル保護のエントリ数の削減を図ったシステムも存在しなかった。

加えて、独自のベンチマークプログラム（getpid システムコールを複数回呼び出すプログラム）でシステムの性能評価を行ったところ、先行システムである DRAKVUF よりも 306 倍の高速化が達成されたことが分かった。

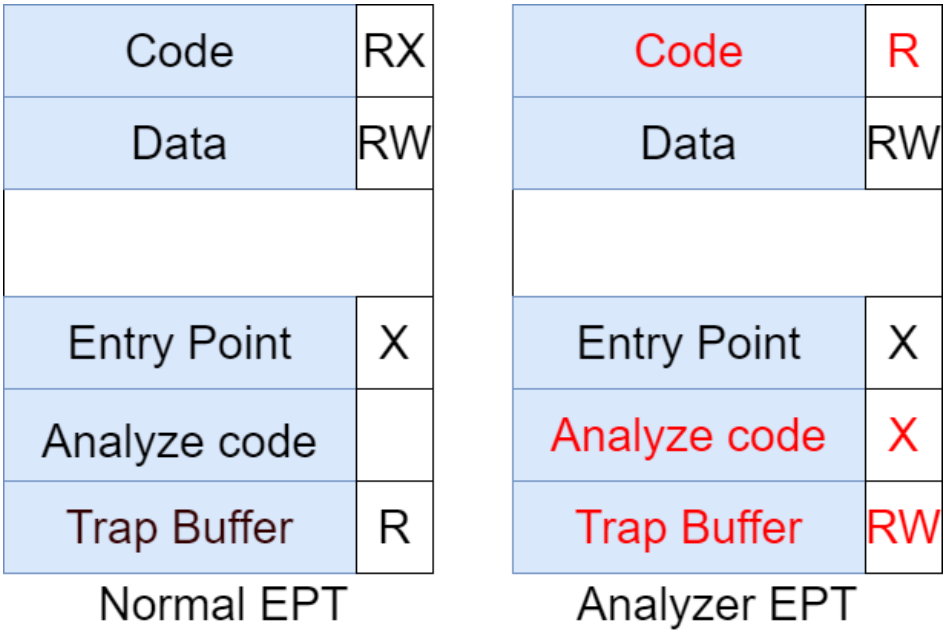

図 4. メモリマップの動的変更

5. 期待される効果

本システムによりバイナリの動的解析が高速になる。例えばマルウェアの解析の場合、解析できるサンプル数の増加や、解析の高速化が図れる。それに伴い、インシデントの発生時に速やかに次の対応が行えるようになる助けになる。

6. 普及（または活用）の見通し

本システムは近日中にオープンソースソフトウェアとする予定であり、オープンソースソフトウェアとなった暁にはマルウェア解析者がシステムへ手を加えることや、独自の解析用 binary の開発を行えるようになる。オープンソースソフトウェアとなった際は様々な解析用 PIC バイナリの拡充が見込まれる。

7. クリエータ名（所属）

森 瑞穂（電気通信大学）