令和 2 年度 卒業論文概要			
所　　属	機械情報工学科	指導教員	光來 健一
学生番号	15237079	学生氏名	渡部　太貴
論文題目	Linux eBPF を用いた仮想マシン内情報の安全な取得機構		

1　はじめに

近年, AWS などの IaaS 型クラウドを用いたサービスが広く利用されるようになってきている．クラウドはオートスケールやセキュリティなどの付加サービスを利用者に提供するために，仮想マシン（VM）内の情報を取得して性能監視や侵入検知を行う．クラウド側から VM 内の情報を取得する方法として，エージェント方式や VM イントロスペクション方式が用いられる．エージェント方式は，VM の利用者にエージェントと呼ばれるソフトウェアをインストールさせ，エージェントと通信して VM 内の情報を取得する方法である．しかし，エージェントの保守コストが高く，更新を怠ると脆弱性になる危険性がある．もう一つの VM イントロスペクション方式は，クラウド側から VM のメモリに直接アクセスして OS データを取得する．しかし，低レベルなメモリ解析が必要となるため開発が難しく，VM 内の OS のバージョンに強く依存してしまう．

本研究では，クラウド側から eBPF プログラムを VM 内の OS カーネルに送り込んで実行することで，VM 内の情報を安全に取得するシステム WaeP を提案する．

2　VM 内の情報取得

エージェント方式は，クラウド側から VM 内の情報を取得するための方法の一つである．この方式では，VM 内のシステムにエージェントと呼ばれるソフトウェアをインストールさせ，クラウド側はネットワーク経由もしくは VM 専用の経路でエージェントと通信を行う．エージェントは OS のプロセスとして実行されることが多い．例えば，AWS で利用可能な Amazon CloudWatch エージェントは VM の性能監視やログ監視を行うことができる．一方，OS に組み込まれるカーネルモジュールとして実行されることもある．例えば, VirtualBox Guest Addition では VM 内のクリップボードの共有などの機能拡張を行うことができる．

エージェント方式を利用する際の問題点は，VM の利用者自身がエージェントをインストールする必要があることである．インストール後も継続的にエージェントの保守に手間がかかり，エージェントのバージョンアップを怠ると脆弱性となって外部からの攻撃を受ける可能性がある．また，エージェントを OS のプロセスとして実行する場合はできることが限られるという問題もある．プロセスからシステム内のあらゆる情報が取得できるわけではないためである．エージェントをカーネルモジュールとして実行すればこの問題は解決できるが，OS に組み込むためにシステムが不安定になる可能性がある．さらに，カーネルモジュールは OS のバージョンに強く依存するため，OS をバージョンアップした際にはエージェントもバージョンアップすることが必要となる．

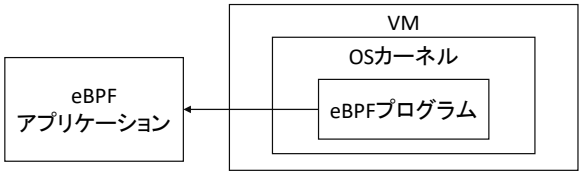

図 1　WaeP のシステム構成

VM イントロスペクション方式は，クラウド側から VM 内の情報を取得するためのもう一つの方法である．この方式では，クラウド側から VM のメモリに直接アクセスして情報を取得する．VM のメモリには様々なシステム情報が格納されているため，メモリを解析して OS データを特定することで性能やセキュリティに関する情報が得られる．VM イントロスペクション方式には VM 内にソフトウェアをインストールする必要がないという利点があるが，様々な問題もある．まず，メモリの低レベルな解析が必要となるため，開発が難しくなる．VM のメモリから OS データを取得するには OS のデータ構造に関する情報が必要となるため，OS のバージョンに強く依存する．また，最近の AMD 製 CPU は VM のメモリを暗号化して VM 外部への情報漏洩を防ぐ SEV 機能を提供しているため，クラウド側からは VM のメモリにアクセスすることができない．

3　WaeP

本研究では，クラウド側から eBPF プログラムを VM 内の OS カーネルに送り込んで実行し，VM 内の情報を安全に取得するシステム WaeP を提案する．WaeP は図 1 のように，クラウド側で既存の eBPF アプリケーションを実行し，VM 内の OS に送り込んだ eBPF プログラムと通信することによって VM 内の情報を取得する．

3.1　eBPF

eBPF は Linux の Berkeley パケットフィルタ（BPF）を拡張したものである．eBPF プログラムは独自の命令セットを用いて作成され，OS カーネルに送り込んで実行することができる．eBPF プログラムは OS カーネル内の多くの処理をフックすることができ，その際に OS 内の様々な情報を取得することができる．取得した情報はカーネル外部の eBPF アプリケーションに渡すことができる．eBPF はプログラムがシステムに危険を及ぼさないかどうかを実行時にチェックするため，OS カーネル内でも安全に実行することができ，システムを不安定にすることはない．また，eBPF は OS データを抽象化しているため，OS のバージョンの違いの影響を受けにくい．

```
SEC("raw_tracepoint/sys_enter")
int bpf_prog(struct bpf_raw_tracepoint_args *ctx)
{
  unsigned long syscall_id = ctx->args[1];

  if (syscall_id == __NR_execve)
    bpf_printk("Hello World!\n");

  return 0;
}
char _license[] SEC("license") = "GPL";
```

図 2　eBPF プログラムの例

図 2 は eBPF プログラムの例である．このプログラムは，OS カーネル内のシステムコールの実行を開始する箇所に設置されたトレースポイントで処理をフックする．そして，システムコール番号が execve に一致した場合にはメッセージを出力する．

3.2　bpf システムコールの転送

eBPF の基本機能は bpf システムコールを用いて実現されている．例えば，eBPF プログラムをカーネルにロードしたり，処理をフックするトレースポイントを設定したりするために使われる．そこで，WaeP は eBPF アプリケーションによる bpf システムコールの実行を横取りし，VM 内で動作させたプロキシに転送する．そして，プロキシが代わりに bpf システムコールの実行を行う．eBPF アプリケーションと VM 内のプロキシは TCP/IP を用いて通信を行う．eBPF アプリケーションは実行を開始するとプロキシとの間でネットワーク接続を確立する．bpf システムコールを実行する際にはコマンド番号および，それぞれのコマンド実行に必要なデータを格納した bpf_attr 共用体とそのサイズを送信し，システムコールの実行結果を受信する．

WaeP は bpf_attr 共用体のデータをプロキシに送信する際に，図 3 のように共用体の中でポインタを使って指されている外部データを別途，送信する．例えば，eBPF プログラムをロードするコマンドの場合には，eBPF プログラムをコンパイルして生成されたバイトコードを送信する．プロキシは新しく確保したメモリにバイトコードを格納し，bpf_attr 共用体の中のポインタを更新する．また，ライセンスを表す文字列（図 2 では”GPL”）についてもその長さとともに送信する．トレースポイントを設定するコマンドの場合には，トレースポイント名を表す文字列（図 2 では”sys_enter”）を同様にして送信する．

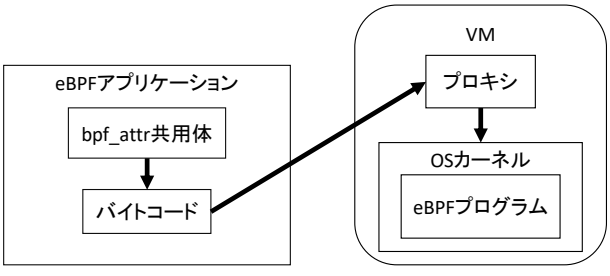

図 3　bpf_attr 共用体とその外部データの送信

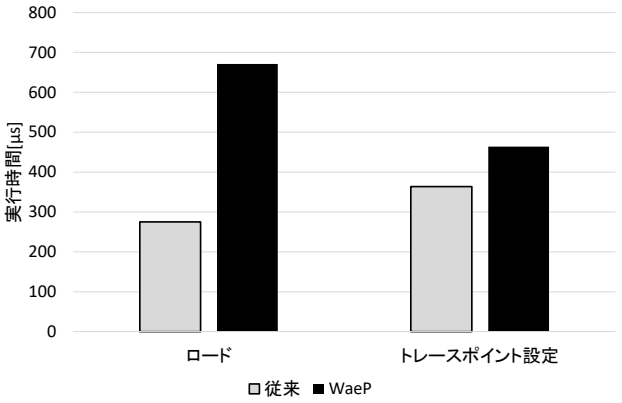

図 4　bpf システムコールの各コマンドの実行時間

3.3　eBPF ライブラリでの透過的な転送

WaeP は既存の eBPF アプリケーションに対して透過的に bpf システムコールの転送を行うことができる．C 言語で eBPF アプリケーションを作成する際には，libbpf と呼ばれる Linux カーネル付属の標準的な eBPF ライブラリが用いられることが多い．libbpf は eBPF プログラムが格納されたファイルを指定して簡単にロードが行える関数を提供している．その関数の中で指定された ELF オブジェクトファイルを解析して eBPF プログラムを読み込み，ロードを行う．そこで，libbpf の中で bpf システムコールを呼び出す関数を書き換えることで bpf システムコールを横取りする．

4　実験

WaeP を用いて eBPF プログラムを VM 内の OS カーネルに送り込んで実行できることを確認する実験を行い，実行時間を測定した．比較として，VM 内で従来の eBPF アプリケーションを実行した時の実行時間についても調べた．この実験では，図 2 の eBPF プログラムを用いた．実験には，Intel Core i7-9700 の CPU，16GB のメモリを搭載したマシンを使用した．仮想化ソフトウェアとして KVM を用い，VM には仮想 CPU を 1 個，1.5GB のメモリを割り当てた．OS には Linux 5.4 を用いた．

実験の結果，eBPF プログラムを VM 内に送り込んだ後，新しいプログラムを実行するたびにログにメッセージが出力されることが確認できた．また，bpf システムコールの 2 つのコマンドの実行時間は図 4 のようになった．実験結果より，WaeP では eBPF プログラムのロード時に 396μs，トレースポイントの設定時に 100μs のオーバヘッドがあることが分かった．

5　まとめ

本研究では，クラウド側から eBPF プログラムを VM 内の OS カーネルに送り込んで実行し，VM 内の情報を安全に取得するシステム WaeP を提案した．WaeP では eBPF ライブラリにおいて bpf システムコールの実行を横取りして VM 内のプロキシに転送し，プロキシが代わりにシステムコールを実行する．今後の課題は，様々な eBPF アプリケーションに対応することである．