仮想マシンモニタによるプログラムコードの秘匿化

平井成海[†] 高橋一志[†] 大山恵弘[†]

1. はじめに

世界には多くの商用アプリケーションが存在する. 商用アプリケーションには多くの技術が含まれており, それらの技術の中には利用者から隠したいものが含まれている場合がある. アプリケーションのコードに含まれている技術を隠すための手段として, 難読化[1)] と呼ばれる手法が広く使われている. 難読化はコードを複雑にすることでコードの解析を妨害する手法であるが, 今日においてはそれ以上の対策が求められている. そこで, 秘密にしたい技術が含まれる部分のコードそのものを隠し利用者には入出力のみ見せることで, 解析を困難にすることができると考えた. 本研究では, 仮想マシンモニタ (VMM) を利用することでアプリケーション内の関数を秘匿しながら実行できるシステムを提案する. 提案システムでは, プログラマがコード内の秘匿化したい部分を関数単位で指定して秘匿化を行い, 秘匿した関数を呼び出すための命令が実行されたときに, VMM で検知して秘匿した関数を VMM 内で実行し, ゲスト OS に結果を返すことで秘匿化を実現する.

2. ユースケースと提案システム

以下に提案システムのユースケースを述べる. また, 図 1 にシステムの全体図を示す. 高価なアプリケーションは不正利用されたときの被害が大きくなるため, 悪意のある利用者の解析対象となりやすいシリアル番号認証のプログラムコードは利用者から隠したい場合がある. 本研究では, IaaS クラウド環境に対して適用することを前提条件として考えている. また, アプリケーションベンダとクラウド事業者は味方であり, 信用できないのは利用者のみとする. アプリケーションは VM 内にプリインストールされた状態で提供され, 利用者は VMM 以下にはアクセスできない. アプリケーションベンダのプログラマは秘匿したい関数に印を付ける (図中の`_ATTR(HIDDEN)`) ことで, 秘匿したい関数をアプリケーションから切り出す (図中の `hidden_func.o`). なお, この部分は LLVM を利用できると考えている. その他のコード (図中の `main.o`) は, 秘匿化対象関数の呼び出しが通常の `call` 命令から関数を識別するための関数 ID の `mov` 命令と VM-VMM 間の関数呼び出しを可能にするハイパーコールに置き換えられる. 最終的に生成された 2 つのオブジェクトファイルをクラウド事業者に提供し, クラウド事業者は `main.o` を利用者の VM 内に, 秘匿化対象コードが含まれる `hidden_func.o` を VMM 内に配置する. 利用者がアプリケーションを利用して秘匿化対象関数が実行されると (1) ハイパーコールが呼ばれ VM-exit が発生し VMM に制御が移る (2) VMM は渡された関数 ID から呼び出すべき秘匿化関数の本体を決定し, 実行する (3) 最後に, 関数の戻り値をゲスト OS に返す (4) 仮に悪意のある利用者がデバッガ等を利用してコードを解析しようとしても, 手に入れられる逆アセンブル結果はハイパーコールのコードのみである. 利用者は VMM 以下にアクセスできないので, `hidden_func.o` の入手もできない. 結果, ユーザは関数の入力を変化させたときの出力の変化から内部コードを推測するしかない. これは内部コードの逆アセンブル結果がわかっているときのコード解析に比べ, はるかに困難である. 結果として, 悪意のあるユーザからコードのロジックを守ることが可能になる.

3. 実装と実験

提案システムを実現するために, KVM[2)] と QEMU[3)] を改造して実装を行なった. 動作環境は, Intel VT-x 対応の CPU 上で動作している 64bit Linux を想定した.

3.1 秘匿された関数の呼び出し

ハイパーコールには, Intel VT-x に用意されている `vmcall` 命令を利用した. また, 秘匿された関数を呼び出す際の呼び出し規約は, 64bit Linux で利用されている System V AMD64 ABI を基にした独自の呼び出し規約を採用し, 関数 ID などの情報を渡せるようにした. また, 秘匿化対象関数の本体の呼び出し規約には System V AMD64 ABI を利用し, QEMU に ABI に従った関数呼び出しを行うコードを実装した.

3.2 実験内容と結果

提案システムを利用して, 商用レンダラーである RenderMan のオープンソース版である Pixie[4)] に提案システムを利用したライセンス認証機能を付加する

† 電気通信大学
The University of Electro-Communications

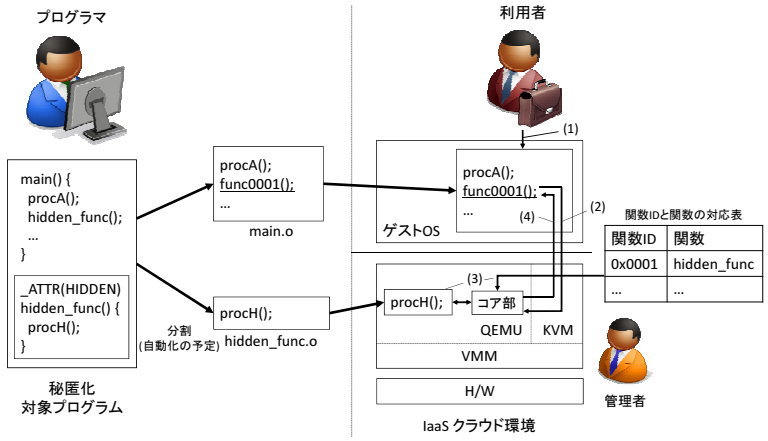

図 1 システムの全体図

```
(gdb) disas check
...
   0x0000000000403a33 <+19>:    mov    %r9,%rdi
   0x0000000000403a36 <+22>:    mov    %r10,%rsi
   0x0000000000403a39 <+25>:    mov    %r11,%rdx
   0x0000000000403a3c <+28>:    mov    %rbx,%rcx
   0x0000000000403a3f <+31>:    callq  *%r8
...

(gdb) disas check
...                                              追加
   0x0000000000403a1c <+12>:    movabs $0xca110001,%rax
   0x0000000000403a26 <+22>:    mov    %r9,%rdi
   0x0000000000403a29 <+25>:    mov    %r10,%rsi
   0x0000000000403a2c <+28>:    mov    %r8,%rdx
   0x0000000000403a2f <+31>:    mov    %r11,%rcx
   0x0000000000403a32 <+34>:    vmcall
...                                              変更
```

図 2 ライセンス認証部分の逆アセンブルの比較

実験を行なった．Pixie のレンダリングプログラムである `rndr` の起動時にライセンスキーを入力を求める処理を追加し，提案システムによってライセンスキーの正当性をチェックする関数の秘匿化を行なった．LLVM を用いた秘匿化対象関数の自動切り出しは未完成であるため，切り出しは手動で行なった．提案システムを適用した結果，正しいライセンスキーを入力するとライセンス認証が行われてアプリケーションの利用ができることが確認できた．図 2 に提案システムを使用しないときと使用したときの秘匿化対象関数の呼び出し部分の逆アセンブルリストを示す．このように，`rndr` のライセンス認証部分を逆アセンブルしても，ライセンス認証のアルゴリズムを知ることはできなかった．

4. 関 連 研 究

佐久間らの研究[5] は，OS カーネルの命令を秘匿するために VMM を用いている．技術的には，OS カーネルを対象としているため特権命令である `hlt` 命令をハイパーコールとして利用した上で，VMM 上で秘匿化対象命令のエミュレーションを行なうことで秘匿化を実現している．本研究は VMM を用いている点においては同一であるが，ユーザ空間から呼び出せるようなハイパーコールとしている点と，秘匿化の細かさが命令単位ではなく関数単位であるため命令のエミュレーションが不要という点において異なる．

5. お わ り に

VMM を利用することで，関数単位で秘匿化を行う技術を提案した．この技術はリバースエンジニアリングを困難にし，プログラムコードを利用者から隠蔽することができる．現時点では引数は即値のみの対応であるが，引数にポインタを利用できるようにすることで，文字列操作を行うような関数などに対しても秘匿化を適用することができると考えられる．このとき，秘匿された関数内でのセグメンテーション違反などのエラー処理などの問題が発生するが，KVM を通してゲスト OS に通知することで適切な処理を行わせればよいと考えている．また，現時点では秘匿化対象関数の分割は手動であるので，LLVM を用いた自動切り出しの機構の実装や導入によるオーバーヘッドの測定などの評価を行うことを予定している．

参 考 文 献

1) Christian Collberg and Jasvir Nagra: *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*, Addison-Wesley Professional (2009).
2) KVM: `http://www.linux-kvm.org/`.
3) QEMU: `http://www.qemu.org/`.
4) Okan Arikan: Pixie - Open Source RenderMan, `http://www.renderpixie.com/`.
5) 佐久間充, 大山恵弘: HyperCensor: 仮想マシンモニタを用いた OS バイナリコードの秘匿化, ディペンダブルシステムワークショップ&シンポジウム (DSW & DSS 2011), 京都工芸繊維大学 (2011).