Virtual Machine Introspection のための VMCrypt の拡張

田所秀和[†] 光来健一[†,††]

1. VMCrypt

Amazon EC2[1] に代表される Infrastructure as a Service (IaaS) と呼ばれるクラウド環境が広く利用されている。IaaS では、サービスとしてデータセンタに置かれた仮想マシン (VM) をユーザに提供している。ユーザは従来自分達で管理していたハードウェアやソフトウェア、データを IaaS に置き、必要なとき必要なだけ VM を使うことができる。IaaS では、ユーザはインターネットを通して遠くから VM 利用する一方、IaaS の管理者は、管理 VM と呼ばれる特別な VM を使い VM を管理している。IaaS 管理者は、ハードウェアのメンテナンスやロードバランスなどの管理のために管理 VM を使いユーザ VM を別のホストに移送する。

このような管理 VM の特権は、管理 VM を経由した情報漏洩の原因になりうる。ユーザからは、IaaS 管理者は必ずしも信用できるとは限らない。怠惰な管理者は管理 VM への侵入を許すかもしれないし、管理者自身が信頼できず攻撃者となる可能性もある。このような攻撃者は、ユーザ VM から簡単に情報を盗むことができる。管理 VM は VM マイグレーションなどの管理のためにユーザ VM のメモリにアクセスできるため、攻撃者は管理 VM を利用することでユーザ VM 内の情報を盗むことができる。

このような問題を解決するために、我々は *VMCrypt*[2] と呼ばれる管理 VM 経由の情報漏洩を防ぐシステムを開発してきた。**図 1** のように、VMCrypt はノーマルビューと暗号ビュー 2 つのメモリビューを提供する。ユーザ VM はノーマルビューで通常通りに動き、管理 VM がユーザ VM のメモリにアクセスする場合には暗号ビューにより情報を盗むことができない。これら 2 つのビューは同時に存在することができるため、管理 VM とユーザ VM は同じメモリに同時にアクセスすることができる。また、管理 VM がユーザ VM の管理を可能にするために、VMCrypt はユーザ VM のいくつかのメモリをノーマルビューとして提供する。VMCrypt は、管理 VM やユーザ VM、仮想マシンモニタ (VMM) 間のやりとりを観察することで、管理のために必要ないくつかのメモリを非暗号化メモリとして自動的に識別している。

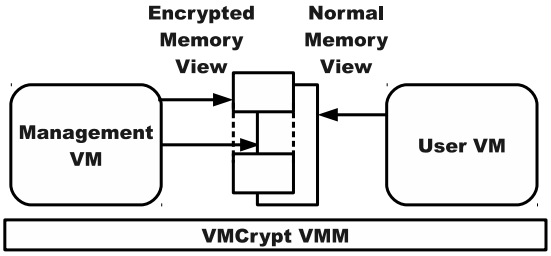

図 1 A dual memory view provided by VMCrypt.

しかし、VMCrypt を単純に利用すると、Virtual Machine Introspection (VMI) と呼ばれる技術が利用できなくなってしまう。VMI とは、ゲスト OS カーネルの中のデータ構造を VMM やホスト OS から識別する技術であり、セキュリティソフトウェアを安全に実装するためなどに使われている。例えば Livewire[3] では VMI を利用することで VM の外に侵入検知システムを置くことを可能にしている。VMCrypt では管理 VM からのメモリアクセスを暗号化する。管理 VM に情報が漏れないという反面、暗号化のため Introspection に必要な情報も取得できなくなってしまう。

2. Introspection のための拡張

そこで、情報漏洩を防ぎつつ管理 VM による Introspection を可能にするよう VMCrypt を拡張した。**図 2** のように、Introspection のために必要なメモリを限定的に非暗号化メモリとすることで、Introspection と情報漏洩の防止を両立した。現在の実装では、ゲスト OS カーネル内のプロセス構造体を Introspection 用の情報として非暗号化メモリに指定できる。これにより管理 VM はユーザ VM のプロセス構造体を見ることができるので、プロセスリストをたどりプロセス

† 九州工業大学
Kyushu Institute of Technology
†† 独立行政法人科学技術振興機構, CREST
JST, CREST

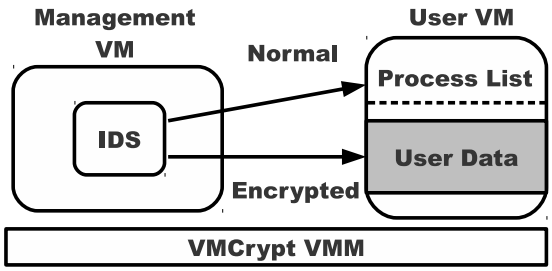

図 2 Introspection のために一部のメモリは暗号化しない

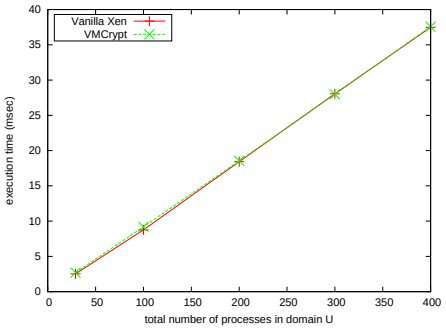

図 3 管理 VM からユーザ VM のプロセスリストをたどる時間

一覧を取得するセキュリティソフトを動かすことが可能になる。一方で、ユーザプロセスが確保したメモリの内容などは従来通り暗号化されるため、情報漏洩は起らない。

今回の Introspection のための拡張では、管理 VM がページをマップするごとに、プロセスとして使われるページを識別し非暗号化ページとしてビットマップに記録する。ビットマップとは、VMCrypt がページを暗号化するかどうかを管理するためのデータ構造である。管理 VM がユーザ VM のメモリページをマップするときに、VMCrypt はこのビットマップを参照し必要なら暗号化を行なう。もし、非暗号化ページならそのままマップを許可し、暗号化ページならマップ先を暗号化した複製に差し替える。このように 2 つのメモリビューを実現している。ビットマップにプロセス構造体として使われるページを記録することで、管理 VM は暗号化していないプロセス構造体を見ることができプロセス一覧を取得できる。VMCrypt は、ゲスト OS のプロセスリストを実際にたどることで、どのページがプロセス構造体のために利用されているかを取得している。

プロセス構造体のために使われるページは、時間とともに変化する可能がある。OS カーネル内のプロセス構造体は、プロセスが作られると新たに確保され、そのプロセスが終了すると解放される。VMCrypt はページマップが起る毎にリストをたどり、プロセス構造体に使われているページかを確認している。このため、ユーザ VM が一時停止している間はプロセス構造体の情報をキャッシュしておくことで、VMCrypt がプロセスリストをたどる回数を減らし、オーバーヘッドを削減している。

3. 実　　験

VMCrypt を使う場合と使わない場合で、管理 VM から Introspection でユーザ VM のプロセスリストをたどる時間を測定した。今回の拡張により、VMCrypt を使う場合でも Introspection が可能になった。プロセス数を変化させ、それぞれプロセスリストをたどるのにかかる時間を測定した。**図 3** が結果である。VMCrypt によるオーバーヘッドは十分小さいことがわかった。これはマップ操作が比較的遅いため、VMCrypt によるオーバーヘッドが見えなくなってしまったためと考えられる。

4. 今後の課題

今後の課題として、より多くのメモリに対して Introspection を許可するように実装することが考えられる。現在の実装ではプロセスの一覧のみ可能であるが、OS カーネルのコード領域への Introspection を許可しコードが改竄されていないか検知するシステムを動かせるようにすることも考えられる。また、色々な種類のメモリに対して Introspection 可能になった場合、情報の保護と Introspection の許可を柔軟に切り替えられる機構も考えられる。どのメモリに対して Introspection を許可するのかを、ユーザがポリシとして記述し、そのポリシに従い VMCrypt が暗号化を切り替えるようにする。

参　考　文　献

1) Amazon Web Services: Amazon Elastic Compute Cloud, `http://aws.amazon.com/ec2/`.
2) Hidekazu Tadokoro, Kenichi Kourai and Shigeru Chiba: Preventing Information Leakage from Virtual Machines' Memory in IaaS Clouds, *IPSJ Transactions on Advanced Computing Systems (ACS)*, Vol.5, No.4, pp.101–111 (2012).
3) Garfinkel, T. and Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, *Proc. Network and Distributed Systems Security Symp.*, pp. 191–206 (2003).