仮想計算機と仮想スイッチを用いたバイパス不可能なアプリケーションレベル・ファイアウォールの提案

庄司 明洋[1]　新城 靖[1]

1. 序論

近年では多くの人々が複雑なアプリケーションを日々利用している．そのようなアプリケーションはしばしばインターネットと通信する．アプリケーションが行う通信の宛先と送信元を検証し，セキュリティを高めたいという要求がある．例えば，あるアプリケーションが開発元と通信することは許すが，他のホストとは通信させたくない場合が考えられる．そのような場合，Little Snitch[1] のようなアプリケーションレベルでのファイアウォールを使うことが有効である．アプリケーションレベル・ファイアウォールとは，ルータ等のネットワーク上の機器で動作するものとは異なり，一般にアプリケーションと同じ OS で動作し，個々のアプリケーションを認識して通信先に対するアクセス制御を行うファイアウォールである．また，ユーザと対話的にポリシーを設定する機能を持つものがある．

従来のアプリケーションレベル・ファイアウォールは，実装は比較的容易ではあるが，バイパス可能であるという問題がある．たとえば，macOS Big Sur では，Little Snitch がバイパスされることがあるという報告がある [2]．また，GNU/Linux で利用可能な Open Snitch[*1]，Douane[*2]，Portmaster[*3]なども Little Snitch 同様にバイパスされうる．

本研究の目的は，バイパス不可能なアプリケーションレベル・ファイアウォールを実装することである．その目的の達成のために，本研究では仮想マシンモニタ (VMM: Virtual Machine Monitor) を用いて作成した仮想マシン (VM: Virtual Machine) インスタンス上のゲスト OS でアプリケーションを動作させ，そのときに VM が行う通信に対してアクセス制御を行う．この際，1 つの VM あたり 1 つのアプリケーションを実行し，VM が送受信するパケットを全て仮想スイッチで監視することで，バイパス不可能なアプリケーションレベル・ファイアウォールを実装する．

2. 提案手法

本研究で実装するアプリケーション・レベルファイアウォールの構成を**図 1** に示す．主要な構成要素は次の 3 つである．

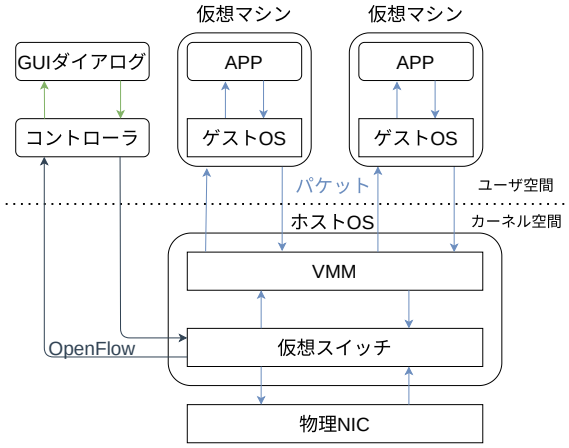

図 1　提案するアプリケーションレベル・ファイアウォールの構成

- アプリケーションの実行環境としての仮想マシン．VMM として，Firecracker[4] を用いて実装する．
- ホスト OS のカーネル空間で動作する，制御可能な仮想スイッチ．Open vSwitch[*4]を用いて実装する．
- ホスト OS のユーザ空間で動作する，仮想スイッチを制御するためのコントローラ．GUI によりユーザと対話する機能を持つ．Rust 言語により rust_ofp[*5]を用いて実装する．

仮想スイッチに VM の NIC と ホストマシンの物理 NIC を接続し，LAN やインターネットへの通信を仮想スイッチを経由して行えるようにする．続いて，コントローラを仮想スイッチに接続する．コントローラと仮想スイッチは OpenFlow[*6]によって接続される．図 1 に示したように 1 つの VM 上では 1 つだけアプリケーションを動作させる．複数のアプリケーションを実行するときには，アプリケーションの数だけ VM を実行する．これにより，特定の VM が行う通信はすべて単一のアプリケーションのものとみなすことができ，アプリケーションごとにポリシーを記述することが容易になる．

[1] 筑波大学
[*1] https://github.com/evilsocket/opensnitch
[*2] https://douaneapp.com
[*3] https://safing.io/portmaster/
[*4] https://www.openvswitch.org
[*5] https://github.com/baxtersa/rust_ofp
[*6] https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf

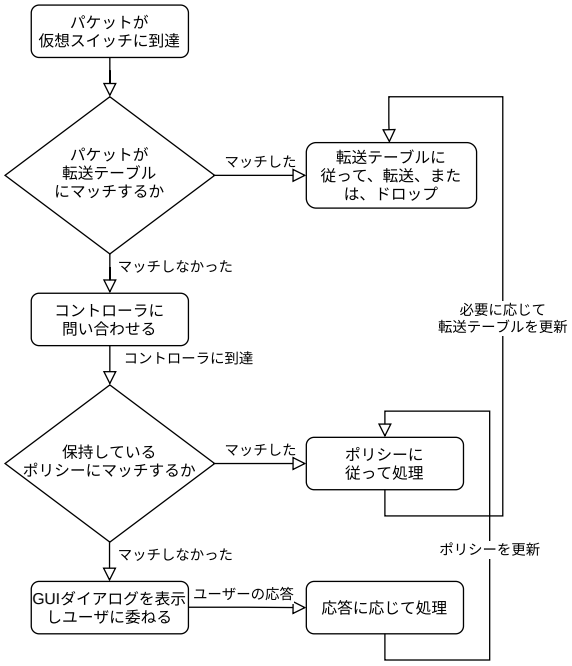

図 2 仮想スイッチとコントローラの動作図

本研究では，VM エスケープは不可能なものとする．ゲスト VM 上の OS は信用しない．すなわち，脆弱性によってゲスト OS 上のセキュリティ機構がバイパスされても問題ないようにする．本研究ではゲスト OS に改変を行わない．

また，本アプリケーションレベル・ファイアウォールをホスト OS 上のアプリケーションとして実装する．これにより，ホスト OS の機能を利用して GUI によるユーザインターフェースを実装することも可能になる．

本研究で実装するアプリケーションレベル・ファイアウォールは，アプリケーションの行う通信の宛先と送信元の IP アドレスとポート番号をチェックする．具体的には，パケットのヘッダを見て，ユーザが設定したポリシーを満たすと判定した場合にパケットを通過する．ポリシーを満たさないと判定した場合はドロップする．ポリシーを満たすかどうか判定できない場合は，GUI のダイアログを表示し，ユーザが動的にポリシーを追加する．

仮想スイッチとコントローラの動作を**図 2** に示す．仮想スイッチは内部に保持する転送テーブルに基づいて仮想スイッチに到達したパケットに対する振る舞いを決定する．仮想スイッチの転送テーブルにマッチしないパケットが仮想スイッチに到達した場合は，そのパケットに対する振る舞いを決定できない．このとき，仮想スイッチはコントローラにパケットを転送しコントローラにそのパケットに対する振る舞いの決定を依頼する．

コントローラは，仮想スイッチを経由するパケットの転送の可否（ポリシー）を管理する．ポリシーは，アプリケーションごとに次のようなもので記述できる．

- 通信を許すホストのホスト名，または，IP アドレスとポート番号
- 通信を許す期間，または，回数

ポリシーを記述するとき，IP アドレスの代わりにドメイン名による記述も可能にする．そのために仮想スイッチは DNS 名前解決のパケットをキャプチャし，コントローラに送る．コントローラは，ドメイン名と IP アドレスの対応表を維持する．もし，保持しているポリシーにマッチしないパケットを受け取った場合はユーザとの対話により動的にポリシーを追加する．

3. 関連研究

Hyperwall[3] は，ハイパーバイザを用いて悪意のある通信を防止する．この研究では，ハイパーバイザはシステム・コール経由で生成されたパケットのみを通過させ，カーネル内部で生成されたパケットは悪意のあるものとみなし、落とす．本研究とは，ハイパーバイザを用いてアクセス制御を行う点が共通している．本研究の特徴は、アプリケーション単位で通信可能なホストを設定できることである。

4. まとめ

本研究の目的は，バイパスが不可能なアプリケーションレベル・ファイアウォールを実装することである．本研究では，アプリケーションを VM 内で動かし，仮想スイッチで VM の通信を制御することでバイパスを不可能にする．また，仮想スイッチの挙動はユーザが対話的に制御可能にする．

現在，アプリケーションレベル・ファイアウォールの基本的な機能は動作している．今後の課題は，ユーザとの対話によってポリシーを編集する機能を完成させ，使い勝手と性能を評価することである．

参考文献

[1] LittleSnitch: ⟨https://www.obdev.at/products/littlesnitch/index.html⟩ (2021-11-15)
[2] Jeffrey Paul: *Your Computer Isn't Yours.*, ⟨https://sneak.berlin/20201112/your-computer-isnt-yours/⟩ (2021-11-15)
[3] Michael Kiperberg, Raz Yehuda, and Nezer Zaidenberg. Hyperwall: A hypervisor for detection and prevention of malicious communication. In *International Conference on Network and System Security*, pp. 79–93, 12 2020.
[4] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. Firecracker: Lightweight virtualization for serverless applica- tions. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pp. 419–434, February 2020.