# サーバーにインストールされているソフトウェアバージョン情報をステルスで自動収集する手法の提案

勝野 恭治[1]

**概要**：緊急度が高いソフトウェア脆弱性が発見された場合、クラウド事業者には、脆弱性を持つ可能性があるサーバーを特定し、対象サーバーを使っているクラウド利用者にソフトウェア更新プログラム適用を促したり、長期間ソフトウェア更新プログラムを適用していないサーバーを一時隔離するといった対応が求められる。しかし、クラウド事業者はクラウド利用者に引き渡した後のサーバーに対して原則、クラウド利用者の許可なくアクセスすることができず、各サーバーにインストールされているソフトウェアの現状を把握できないため、対象サーバーを特定するのが困難である。
本論文では、ソフトウェアパッケージをダウンロードする通信に対してモニタリング・ロギングを行い、サーバーにインストールされているソフトウェアとバージョンを収集する手法を提案する。本提案手法を用いることによって、サーバーに直接アクセスを行わず、ログファイルをチェックするのみで、サーバーにインストールされているソフトウェアとバージョンを推定できる。そのため、提供後のサーバーに直接アクセスができないクラウド事業者にとって大変有益である。
本論文では、代表的なパケットモニタリングツールとソフトウェアパッケージインストール・管理システムを検証対象として選定し、実証実験を行うことで、本論文のアプローチが実用的であることを示す。

# A Proposal to Automatically Collect Software Versions installed on Servers without Touching Servers

Yasuharu Katsuno[1]

## 1. はじめに

### 1.1 背景

近年、発見されるソフトウェア脆弱性の数が増大し、脆弱性がサーバーに与える影響が大きくなっている。そのため、ソフトウェア脆弱性に対応するためにソフトウェア更新プログラムやソフトウェアバージョンアップを早期に行う必要性がますます高まっている。ソフトウェア脆弱性は通常、対象となるソフトウェア名とバージョン情報の組で指定される。例えば、脆弱性 CVE-2016-3115[1] では、対象ソフトウェアは OpenSSH[2] で、対象バージョンは 7.2p2 以前である。そのため、脆弱性 CVE-2016-3115 に対処するには、まず、バージョン 7.2p2 以前の OpenSSH がインストールされているサーバーを探さなければならない。

サーバー上にインストールされているソフトウェアとバージョン情報を取得する方法は、大きく分けて、サーバーに直接アクセスをして取得する方法と、サーバー管理システムから取得する方法の 2 通りある。

サーバーに直接アクセスをして取得する方法は、対象サーバー一台一台にログインして、指定のコマンドを実行することで行われる。例えば、Red Hat Enterprise Linux 系の Linux サーバーの場合は、yum コマンドを対象サーバー上で実行することによって、インストールされている主要なソフトウェアとバージョン情報を取得できる。サーバーログインは手作業で行う方法もあるが、Ansible[3] といったサーバー構成管理ツールを使うことによって自動的に行うことも可能である。

サーバー構成管理システムから取得する方法は、サーバー構成管理を行う管理サーバーに問い合わせることに

[1] IBM 東京基礎研究所、東京都中央区日本橋箱崎町 19-21
IBM Research - Tokyo, 19-21 Nihonbashi, Hakozaki-cho, Chuo-ku, Tokyo, Japan

よって行われる。サーバー構成管理システムでは、対象サーバーに管理エージェントと呼ばれるプログラムを事前にインストールする。管理エージェントはサーバーの情報を収集し、管理サーバーに情報をアップロードする。そのため、管理サーバーは対象サーバーにインストールされているソフトウェアとバージョン情報を一元的に管理できる。例えば、サーバー構成管理システムでよく使われている Chef[4] の場合は、管理エージェントは Chef クライアント、管理サーバーは Chef サーバーである。

サーバーに直接アクセスをして取得する方法とサーバー管理システムから取得する方法、どちらがいいかは、サーバーの提供方法、管理するサーバーの台数、許可されているサーバーへのアクセス、といったサーバー提供・管理条件に依存する。例えば、サーバーの台数が少ない場合はサーバーに直接アクセスをして取得する方法を、サーバーの台数が多い場合はサーバー管理システムから取得する方法を採用するのが効率的である。

クラウド基盤サービスで提供しているサーバーのケースを考えてみる。クラウド基盤サービスでは、AWS[5] や SoftLayer[6] といったクラウド事業者がサーバーが使える状態までセットアップを行い、クラウド利用者がセットアップが完了したサーバーを使用する。クラウド基盤サービスにおけるサーバー管理の範囲は、物理サーバーやストレージ、ネットワークといった OS よりも下位レイヤーをクラウド事業者が担当し、OS とそれよりも上位レイヤーをクラウド利用者が担当する。そのため、サーバーにどのようなソフトウェアをインストールするのかを決定して導入を行うのはクラウド利用者となる。

他のサーバーに対して影響を及ぼしかねない緊急度が高いソフトウェア脆弱性が発見され、対応するソフトウェア更新プログラム、新しいバージョンのソフトウェア、がリリースされたとする。クラウド事業者は、自身が提供するサーバーへの影響を考慮して、ソフトウェア更新プログラムを対象サーバーに適用したい。しかし、サーバー管理の担当範囲から、ソフトウェア更新プログラムを自身で対象サーバーに適用することができない。そのため、対象サーバーを使っているクラウド利用者にソフトウェア更新プログラム適用を促したり、長期間ソフトウェア更新プログラムを適用していないサーバーを一時隔離するといった対応が求められる。その際に、クラウド事業者は、対象サーバーを特定しなければならないが、クラウド利用者に引き渡した後のサーバーに対して、クラウド利用者の許可なくアクセスしたり管理エージェントをインストールすることが原則としてできない。つまり、各サーバーにインストールされているソフトウェアの現状を把握できず、どのサーバーに脆弱対象ソフトウェアがインストールされているのかが分からないので、効果的な対応が困難となる。

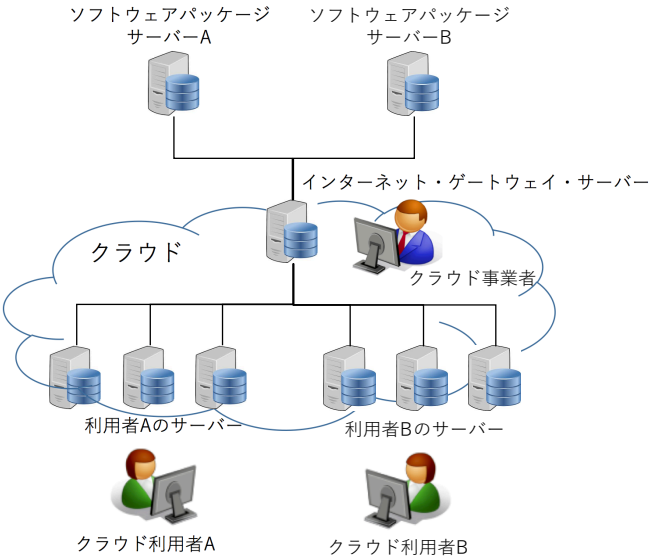


**図 1** 想定クラウド・サーバー環境

### 1.2 本論文の貢献

本論文では、ソフトウェアパッケージをダウンロードする通信をモニタリング・ロギングして、サーバーにインストールされているソフトウェアとバージョンを収集する手法を提案する。本提案手法を用いることによって、サーバーに直接アクセスを行わず、かつ、サーバーにエージェントをインストールすることなく、ログファイルをチェックするのみで、サーバーにインストールされているソフトウェアとバージョンを推定できる。そのため、提供後のサーバーへの直接アクセスや管理エージェントのインストールができないクラウド事業者にとって大変有益である。

本論文では、幅広く用いられているパケットモニタリングツール：tshark[7] と代表的なソフトウェアパッケージインストール・管理システム：Yum (Yellowdog Updater Modified)[8] と APT (Advanced Packaging Tool)[9] を検証対象として選定し、実証実験を行うことで、本論文のアプローチが実用的であることを示す。

本論文の構成は次の通りである。第 2 章で設計、第 3 章で実証実験について述べる。最後に、第 4 章で結論と今後の課題について述べる。

## 2. 設計

本論文で想定するクラウド・サーバー環境を図 1 に示す。

クラウド環境内に、利用者サーバーとインターネット・ゲートウェイ・サーバーが動作している。利用者サーバーは、クラウド事業者から各クラウド利用者に提供されたサーバーである。一旦、提供されたサーバーは基本的には利用者の許可なくアクセスできない（アクセスしてはいけない）。インターネット・ゲートウェイ・サーバーは、利用者サーバーがクラウドの外にあるインターネット上のサーバーと通信するために設置され、クラウド事業者によって

管理・運用されている。利用者サーバーがインターネット上のソフトウェアパッケージサーバーにアクセスしてソフトウェアのインストールやアップグレードを行う際に、ソフトウェアダウンロードに関する通信パケットはインターネット・ゲートウェイ・サーバーを必ず通過する。

本論文のアプローチを次に示す。

(1) クラウド事業者は、クラウド利用者に提供するサーバーにインストール済みのソフトウェアに関する名前とバージョンの組のリストを事前に作成する。
(2) クラウド事業者は、インターネット・ゲートウェイ・サーバーでパケットモニタリングを常時動かし、ソフトウェア通信ダウンロードの通信パターン定義に基づいて該当パケットをファイルにダンプする。
(3) 緊急度が高いソフトウェア脆弱性が報告された場合など、特定のソフトウェアの特定のバージョンがインストールされたクラウド利用者サーバーを調べたい時は、事前作成リストとインターネット・ゲートウェイ・サーバーにあるパケットモニタリングをダンプしたファイルを照らし合わせる。

## 3. 実証実験

本実証実験環境として、Yum 対象サーバーに CentOS 6.8 マシン、APT 対象サーバーに Ubuntu 14.04 マシン、インターネット・ゲートウェイを想定したマシンとして CentOS 6.8 マシンを用意した。

インターネット・ゲートウェイを想定したマシン上でパケットモニタリングツール：tshark[7] を常時実行し、ソフトウェアダウンロードに該当するパケットをモニタリングし、モニタリングした結果をログファイルに格納する。ログファイルからソフトウェアバージョンアップや新規インストールが行われた可能性があるかどうかが検出できるかどうかを検証した。

### 3.1 Yum

図 2 と図 3 に、OpenSSL パッケージの例を示す。

図 2 に、Yum 対象サーバーで実行した yum コマンドの実行結果を示す。コマンド実行結果より、OpenSSL がインストールされており、そのバージョンが 1.0.1e-48 であることが分かる。

図 3 に、インターネット・ゲートウェイを想定したマシン上で取得したパケットモニタリング結果の抜粋を示す。パケットフィルタリングルールは次のように決定した。

- インターネット・ゲートウェイを想定したマシンのネットワークインターフェース: eth0
- Yum でデフォルト設定で使われる HTTP プロトコルの IP アドレスポート番号: 80
- Yum パッケージファイルの拡張子名: rpm
- 送信元サーバーの識別: Internet Protocol

図 3 から、バージョン 1.0.1e-57 がダウンロードされており、ソフトウェアバージョンアップが行われた可能性があることが検出できることが分かる。

### 3.2 APT

図 4 と図 5 に、Bind9 パッケージの例を示す。

図 4 に、Apt 対象サーバーで実行した dpkg コマンドの実行結果を示す。コマンド実行結果より、Bind9 がインストールされていないことが分かる。

図 5 に、インターネット・ゲートウェイを想定したマシン上で取得したパケットモニタリング結果の抜粋を示す。パケットフィルタリングルールは次のように決定した。

- インターネット・ゲートウェイを想定したマシンのネットワークインターフェース: eth0
- Apt でデフォルト設定で使われる HTTP プロトコルの IP アドレスポート番号: 80
- Apt パッケージファイルの拡張子名: deb
- 送信元サーバーの識別: Internet Protocol

図 5 から、バージョン 9.9.5 がダウンロードされていることが分かり、新規インストールが行われた可能性があることが検出できることが分かる。

## 4. おわりに

クラウド事業者がクラウド利用者に引き渡したサーバーにインストールされているソフトウェアの現状把握を、サーバーに直接アクセスすることなく実現するために、本論文では、ソフトウェアパッケージをダウンロードする通信をモニタリング・ロギングして、サーバーにインストールされているソフトウェアとバージョンを収集する手法を提案した。さらに、本論文では、パケットモニタリングツール：tshark とソフトウェアパッケージインストール・管理システム：Yum (Yellowdog Updater Modified) と APT (Advanced Packaging Tool) を用いて実証実験を行い、本論文のアプローチが実用的であることを示した。

本論文では、対象サーバーとソフトウェアパッケージインストール・管理システムサーバーとの通信パケットペイロードがモニター可能であることを想定している。そのため、SSL (Secure Sockets Layer) などを用いて通信パケットペイロードを暗号化している、ソフトウェアパッケージインストール・管理システムに対応できないので、パケット・インペクション技術 [10][11] などを併用する必要があると考えらえれる。

### 参考文献


[1] : CVE-2016-3115, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-3115.
[2] : OpenSSH, https://www.openssh.com.
[3] : Ansible, https://www.ansible.com.

```
centos6> yum list installed | grep openssl
openssl.x86_64                      1.0.1e-48.el6_8.1                  installed
```

**図 2** OpenSSL パッケージ現行バージョンの Yum による確認

```
centos6> tshark -i eth0 -f 'dst port 80' -V | grep -e rpm -e "Internet Protocol"
<略>
Internet Protocol Version 4, Src: X.X.X.X (X.X.X.X), Dst: Y.Y.Y.Y (Y.Y.Y.Y)
    GET /Linux/packages/CentOS/6.9/os/x86_64/Packages/openssl-1.0.1e-57.el6.x86_64.rpm HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /Linux/packages/CentOS/6.9/os/x86_64/Packages/ \\
        openssl-1.0.1e-57.el6.x86_64.rpm HTTP/1.1\r\n]
            [Message: GET /Linux/packages/CentOS/6.9/os/x86_64/Packages/\\
            openssl-1.0.1e-57.el6.x86_64.rpm HTTP/1.1\r\n]
        Request URI: /Linux/packages/CentOS/6.9/os/x86_64/Packages/\\
        openssl-1.0.1e-57.el6.x86_64.rpm
    [Full request URI: http://www.ftp.ne.jp/Linux/packages/CentOS/6.9/os/x86_64/Packages/\\
    openssl-1.0.1e-57.el6.x86_64.rpm]
Internet Protocol Version 4, Src: X.X.X.X (X.X.X.X), Dst: Y.Y.Y.Y (Y.Y.Y.Y)
Internet Protocol Version 4, Src: X.X.X.X (X.X.X.X), Dst: Z.Z.Z.Z (Z.Z.Z.Z)
Internet Protocol Version 4, Src: X.X.X.X (X.X.X.X), Dst: Z.Z.Z.Z (Z.Z.Z.Z)
Internet Protocol Version 4, Src: X.X.X.X (X.X.X.X), Dst: Z.Z.Z.Z (Z.Z.Z.Z)
    GET /Linux/packages/CentOS/6.9/os/x86_64/Packages/openssl-1.0.1e-57.el6.x86_64.rpm HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /Linux/packages/CentOS/6.9/os/x86_64/Packages/\\
        openssl-1.0.1e-57.el6.x86_64.rpm HTTP/1.1\r\n]
            [Message: GET /Linux/packages/CentOS/6.9/os/x86_64/Packages/\\
            openssl-1.0.1e-57.el6.x86_64.rpm HTTP/1.1\r\n]
        Request URI: /Linux/packages/CentOS/6.9/os/x86_64/Packages/\\
        openssl-1.0.1e-57.el6.x86_64.rpm
    [Full request URI: http://ftp-srv2.kddilabs.jp/Linux/packages/CentOS/6.9/os/x86_64/Packages/\\
    openssl-1.0.1e-57.el6.x86_64.rpm]
<略>
```

**図 3** Yum による OpenSSL パッケージダウンロードに該当するパケットのモニタリング結果の抜粋

```
ubuntu14 >  dpkg -l bind9
dpkg-query: no packages found matching bind9
```

**図 4** Bind9 パッケージ現行バージョンの Apt による確認

```
ubuntu14 >  tshark -i eth0 -f 'dst port 80' -V | grep -e deb -e "Internet Protocol"
<略>
Internet Protocol Version 4, Src: XX.XX.XX.XX (XX.XX.XX.XX), Dst: YY.YY.YY.YY (YY.YY.YY.YY)
    GET /pub/Linux/ubuntu/pool/main/b/bind9/bind9utils_9.9.5.dfsg-3_amd64.deb HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /pub/Linux/ubuntu/pool/main/b/bind9/\\
        bind9utils_9.9.5.dfsg-3_amd64.deb HTTP/1.1\r\n]
            [Message: GET /pub/Linux/ubuntu/pool/main/b/bind9/bind9utils_9.9.5.dfsg-3_amd64.deb HTTP/1.1\r\n]
        Request URI: /pub/Linux/ubuntu/pool/main/b/bind9/bind9utils_9.9.5.dfsg-3_amd64.deb
    [Full request URI: http://ftp.jaist.ac.jp/pub/Linux/ubuntu/pool/main/b/bind9/\\
    bind9utils_9.9.5.dfsg-3_amd64.deb]
Internet Protocol Version 4, Src: XX.XX.XX.XX (XX.XX.XX.XX), Dst: YY.YY.YY.YY (YY.YY.YY.YY)
<略>
Internet Protocol Version 4, Src: XX.XX.XX.XX (XX.XX.XX.XX), Dst: YY.YY.YY.YY (YY.YY.YY.YY)
    GET /pub/Linux/ubuntu/pool/main/b/bind9/bind9_9.9.5.dfsg-3_amd64.deb HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /pub/Linux/ubuntu/pool/main/b/bind9/\\
        bind9_9.9.5.dfsg-3_amd64.deb HTTP/1.1\r\n]
            [Message: GET /pub/Linux/ubuntu/pool/main/b/bind9/bind9_9.9.5.dfsg-3_amd64.deb HTTP/1.1\r\n]
        Request URI: /pub/Linux/ubuntu/pool/main/b/bind9/bind9_9.9.5.dfsg-3_amd64.deb
    [Full request URI: http://ftp.jaist.ac.jp/pub/Linux/ubuntu/pool/main/b/bind9/\\
    bind9_9.9.5.dfsg-3_amd64.deb]
<略>
```

**図 5** Apt による Bind9 パッケージダウンロードに該当するパケットのモニタリング結果の抜粋

[4] Marschall, M.: Chef Infrastructure Automation Cookbook, *Packt Publishing, August, 2013.*
[5] : Amazon Web Service, http://aws.amazon.com/.
[6] : SoftLayer, http://www.softlayer.com/.
[7] : tshark - Dump and analyze network traffic, https://www.wireshark.org/docs/man-pages/tshark.html.
[8] : Yum (Yellowdog Updater Modified), http://yum.baseurl.org.
[9] : APT (Advanced Package Tool), https://wiki.debian.org/Apt.
[10] Sherry, J., Lan, C., Popa, R. A. and Ratnasamy, S.: BlindBox: Deep Packet Inspection over Encrypted Traffic, *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM 2015), 2015.*
[11] Cascarano, N., Ciminiera, L. and Risso, F.: Optimizing Deep Packet Inspection for High-Speed Traffic Analysis, *Journal of Network and Systems Management, Volume 19, Issue 1, March 2011.*