自動セキュリティ診断　Amazon Inspector

一部の重大脆弱性を検出せず 利用は容易で開発時にも使える

AWS上のシステムを対象とする自動セキュリティ診断サービス「Amazon Inspector」のプレビュー版が2015年10月に米国リージョンで公開された。検証の結果、重大な脆弱性の一部を検出しなかったが、比較的容易に利用できることが分かった。運用フェーズはもちろん、開発時にも繰り返し使えて、セキュリティの不備による手戻りを減らせる。

中山 秀夫　／　協力・監修　アイレット cloudpack事業部 吉田浩和（セキュリティエンジニア）

「Amazon Inspector」は、AWS上のシステムを対象にした自動セキュリティ診断サービスだ。対象システムの仮想マシンAmazon EC2に専用エージェントソフトを配置し、米Amazon Web Servicesが用意した診断ルールに基づいて、主としてミドルウエア、OS、ネットワークというプラットフォーム層についてのセキュリティ診断を行う。

診断ルールは、非営利団体の米MITREが管理する、オープンソースソフトウエア（OSS）を中心に集めた脆弱性リスト「CVE（Common Vulnerabilities and Exposures）」、認証に関する脆弱性、OSの設定に関する脆弱性、ネットワークの脆弱性、クレジットカード業界のセキュリティ基準「PCI DSS（Payment Card Industry Data Security Standard）」を網羅する。一方、アプリケーションは基本的に対象外だ。

2015年10月に米国リージョンで、Linux環境に対応したプレビュー版がリリースされた。正式サービスインの時期と料金は未定だ。

従来、セキュリティ診断は、新システムのテスト工程、運用に入って半年後といった節目ごとに、セキュリティベンダーへ依頼するケースが多かった。そうした専門家による診断は間隔が空くため、補完できるように、一部の診断を自動的に行うツールの活用が始まっている。Inspectorも専門家による診断を補完するサービスと位置付けられる。

検証内容

意図的に脆弱性を埋め込み 正しく診断できるか試す

利用者としてセキュリティの専門家ではない運用担当者や開発者を想定。検証項目に、Inspectorのセキュリティ診断の対象分野から、間隔の空く専門家による診断を補完する価値が高い「CVE」「サーバー認証」「OSの設定・権限」を選んだ（**図1**）。

一つめのCVEでは、そのリストに載る脆弱性を正しく検出するか調べるため、2014年に見つかった重大な脆弱性「ShellShock」「HeartBleed」「POODLE」「Struts ClassLoader」を埋め込んだUbuntuサーバーをそれぞれ用意した。

二つめは認証である。SSH（Secure Shell）での認証方式、パスワード運用の脆弱性について診断した。

最後に、OSの設定・権限の脆弱性診断を検証した。診断項目は、メインメモリーのアドレス空間配置のランダム化「ASLR（Address Space Layout Randomization）」が無効になっている（アドレス空間を固定にしている）脆弱性、システムディレクトリーのファイルを一般ユーザーが編集できる脆弱性の2点だ。

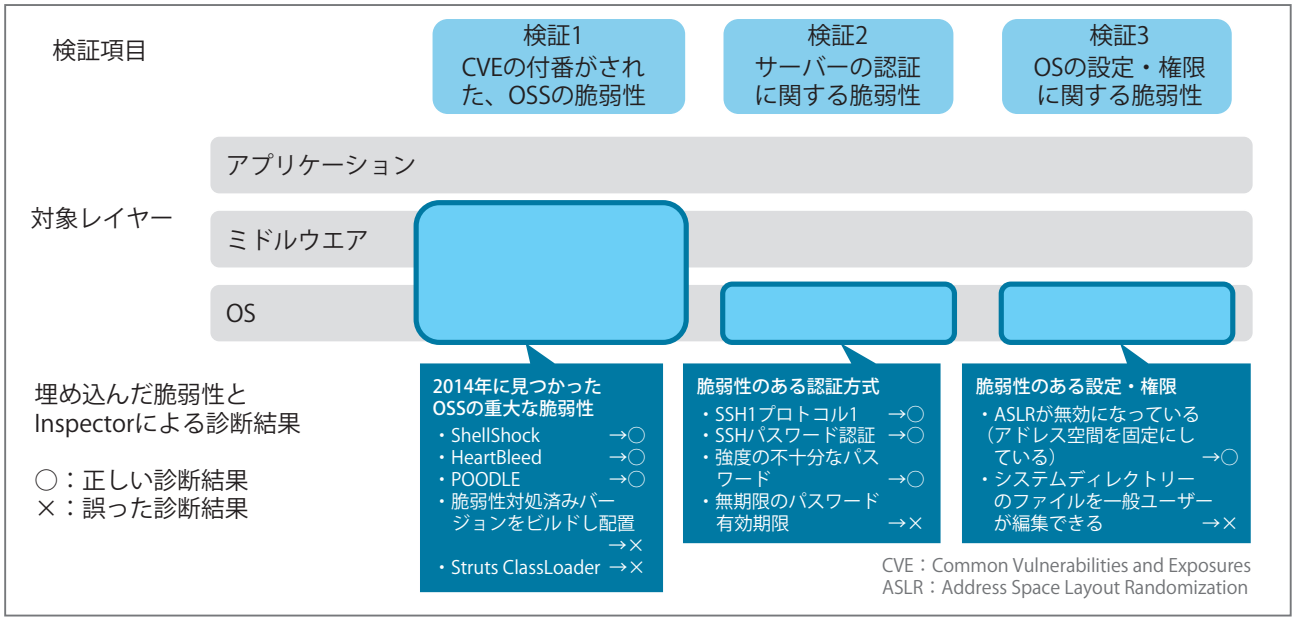
さらに、これらの診断を通して利用の容易さについても検証した。

結果サマリー

診断精度は発展途上の段階 専門知識がなくても使える

一つめの検証テーマであるCVEについては、ShellShock、HeartBleed、POODLEという三つの脆弱性を正し

図1　三つの検証項目
オープンソースソフトウエア（OSS）の重大な脆弱性、サーバーの認証の脆弱性、OS（Ubuntu）の設定・権限の脆弱性について自動診断を行った



く検出した。HeartBleedについては脆弱性パッチを適用したサーバーも用意。これも、正しい診断を下した。

しかし、POODLEの脆弱性に対処したミドルウエアの新しいバージョンを、インストールせずソースコードからビルドして配置したところ、古いバージョンのままだと誤って判断し、脆弱性ありと診断した。

推察できるのは、Inspectorはdpkgやyumといったパッケージソフト管理機構のパッケージデータベースを照会しており、配置されたソフト自体は見ていないことだ。Inspectorの診断精度を高めるには、ソフトウエアをインストールするなどして、パッケージデータベースに履歴が残るようにする必要がある。

また、Struts ClassLoaderの脆弱性は検出できなかった。重大な脆弱性でも一部は検出漏れになる。

二つめのサーバー認証については、脆弱性のあるSSHプロトコル1とSSHパスワード認証、セキュリティ強度の不十分なパスワードを正しく検出した。ただしパスワードの有効期限では、無期限を意味する「99999日」というデフォルト設定のままになっている脆弱性を検出できなかった。原因は、設定値の有無だけを参照しているためだろう。

三つめのOSの認証・設定では、ASLRが無効になっているという脆弱性を正しく検出した。しかしシステムディレクトリーのファイルを一般ユーザーが編集できるという脆弱性は検出できなかった。これも、診断ルールが用意されていないためと考えられる。

総合的に見て検出精度は、重大な脆弱性を一部検出しないなど、機能や検出ルールの網羅性で発展途上だ。補完的な診断ツールとしてもやや物足りない。

一方、利用の容易さについては、設定項目が少なく、操作方法を学べば、セキュリティ専門のエンジニアでなくても利用できる。しかも所要時間は、例えばCVEの診断で1時間程度。1日もあれば、基本的なセキュリティ診断を一通り行える。

このため、診断の頻度が高められる。この利点は、稼働中のシステムの脆弱性を素早く見つけられることだけでない。システム開発中に繰り返し診断することで、セキュリティの不備による手戻りを防げる。

以降で検証結果を詳しく解説し、最後に検証を実施したアーキテクトからの提言を示す。

検証1　CVE 重大な四つのOSS脆弱性 一つは検出せず

CVEは、米国政府が支援するMITREによる脆弱性リストであり、世界的に使われている。脆弱性一つひとつに「CVE-2014-3513」「CVE-2014-3567」といった具合に識別子が振られており、その概要や参考URLがまとめられている。

InspectorはこのCVEのリストを基にした診断ルールを備える。2014年に見つかった、OSSの代表的な脆弱性「ShellShock」「HeartBleed」「POODLE」「Struts ClassLoader」の四つについて、それぞれを意図的に埋め込んだUbuntuサーバーを用意し、Inspectorで自動診断を試みた。

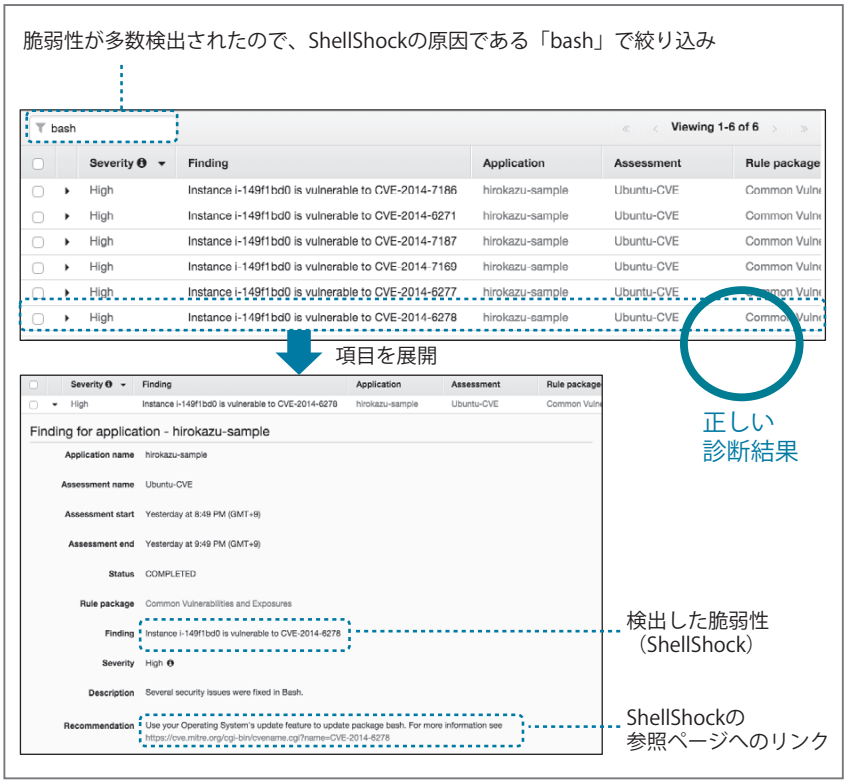
ShellShockは正しく検出

ShellShockの診断からInspectorを試す。ShellShockとは、Webサーバーなどへの不正侵入に悪用される脆弱性である。この脆弱性のあるシェルスクリプトBashは、ヘッダーの値に埋め込まれた任意のコマンドを実行する。

ShellShockの脆弱性がある「Bash 4.3-7」を実装した古いバージョンのUbuntuサーバーを立ち上げ、Inspectorで診断を行った。

診断の結果、古いバージョンのUbuntuであるため、脆弱性が多く見つかった。「bash」のキーワードで絞り込むと、ShellShockの脆弱性に関するCVE-2014-6278の検出結果が表示された（**図2**上）。これは正しい診断結果だ。

図2　ShellShockの脆弱性の診断結果
ShellShockの脆弱性がある「Bash 4.3-7」については正しく検出した。診断結果にはCVE公式サイトの該当ページへリンクが張られる



検出結果の該当項目をクリックすると画面が展開され、Findingの項目に、CVE-2014-6278の脆弱性が見つかった、という意味の指摘が表示された（図2下）。Recommendationの欄には、検出した脆弱性であるShellShockの情報ページへのリンク（MITREのサイト）が張られている。

パッチ適用の有無も勘案

次に、HeartBleedの脆弱性を検出する。HeartBleedは、OpenSSLの拡張機能「heartbeat」の脆弱性。悪用されると、OpenSSLライブラリが利用しているメインメモリーの内容の一部を盗み見られる。メインメモリーの内容には、ユーザーID、パスワード、画面に表示された個人情報などが含まれる可能性がある。HeartBleed（心臓出血）という名称が決して大げさではない重大な脆弱性だ。

HeartBleedに該当する古いバージョンのOpenSSLを実装したUbuntuサーバーを用意し診断したところ、正しく脆弱性を検出した。

続いて、HeartBleedのセキュリティパッチを適用したサーバーを用意して診断する。Inspectorが OpenSSLのバージョンだけを見て

図3　HeartBleedのパッチを適用したサーバーの診断結果
HeartBleedの脆弱性に対処するパッチを適用し、Inspectorで診断したところ、該当なしという正しい診断結果になった

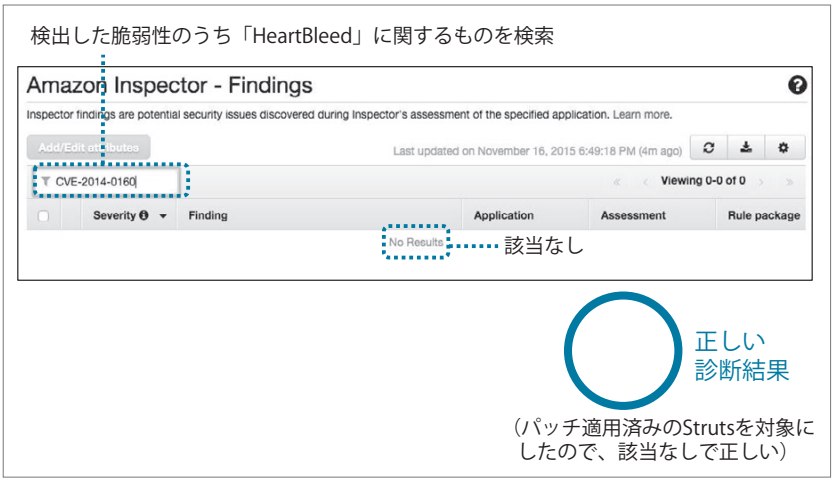


図4　POODLEの脆弱性の診断結果
POODLEの脆弱性も正しく検出した

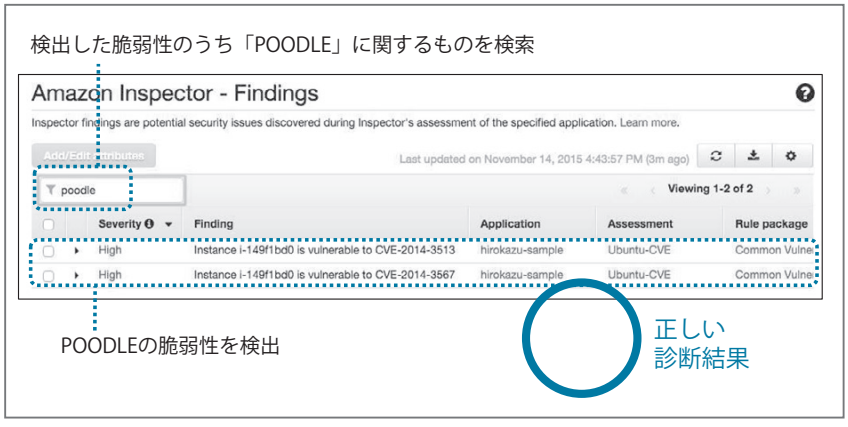
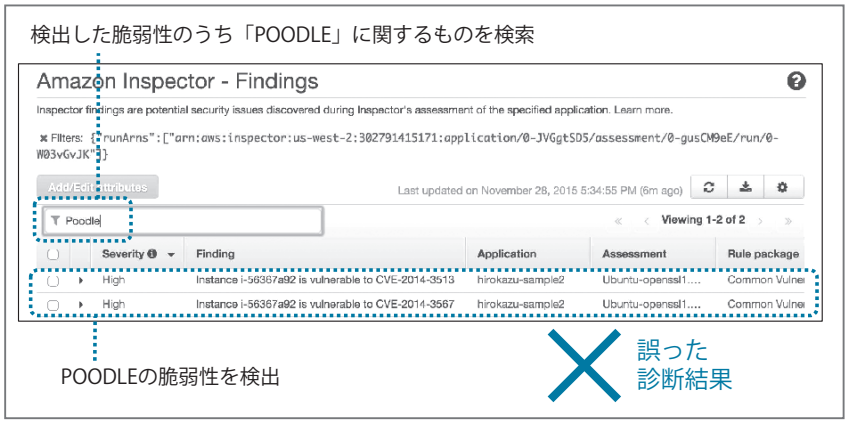


図5　POODLE脆弱性に対応したソースコードからビルドして配置した場合の診断結果
脆弱性に対応済みのOpenSSLをソースコードからビルドして配置し、改めてInspectorで診断したところ、誤った診断結果になった



いるなら、脆弱性ありという診断結果になる。パッチを当てているかどうかまで調べるなら、該当なしの結果になるはずだ。

この検証をしたところ、診断結果は「No Results（該当なし）」（**図3**）。正しい診断結果だった。Inspectorは、パッチの適用についても診断対象に含めているとみられる。

照会先はパッケージデータベース

次に、POODLEの脆弱性診断を試す。POODLEは、SSLバージョン3の脆弱性。SSLバージョン3で暗号化した通信内容が解読される恐れがある。パッチを適用する、SSLのバージョンを上げる、SSLバージョン3を無効化する――のいずれかの対策を講じる必要がある。

POODLEの脆弱性があるOpenSSLを実装したUbuntuサーバーを対象に、Inspectorで診断したところ、正しく脆弱性を検出した（**図4**）。

Inspectorは何を照会してソフトウエアのバージョンを把握しているのか。UbuntuなどのLinuxでは、OS標準のソフト管理機構が備えるパッケージデータベースがその一つと考えて間違いない。ただしパッケージデータベースでは、インストールせずにソースコードからビルドするなどした配置型ソフトを管理しない。

そこで、Inspectorが、配置型ソフトについて正しく診断できるかどうかを検証する。

POODLEの脆弱性に対処したOpenSSL 1.0.1kを、アップグレー

ド(インストール)するのではなく、makeというコマンドを使ってビルド。この処置を施したUbuntuサーバーを、Inspectorで診断した。

結果は、POODLEの脆弱性あり、という誤った診断になった(前ページの**図5**)。これからいえるのは、Inspectorはサーバー上に存在するソフトそのものを調査していないこと。配置型ソフトは対象外になる。

冒頭でも述べたように、Inspectorの診断精度を高めるには、ソフトをインストールするなどして、パッケージデータベースに履歴を残す必要がある。これは、運用保守のセオリーであり、Inspectorのために作業が増えるわけではない。

Strutsの脆弱性は検出せず

CVEについては最後に、Struts ClassLoaderの脆弱性診断を試した。これは、Javaクラスをメモリー上に展開する機構ClassLoaderに関する脆弱性である。悪用されると、ファイルを操作されたり使用不能にされたりする恐れがある。

この脆弱性がある「Struts 2.3.8」を実装したUbuntuサーバーを用意し、Inspectorで診断した。

診断結果は「No Results(該当なし)」(**図6**)。誤った診断になった。

原因は明らかである。Inspectorに適用されるCVEリストを確認すると、「CVE-2014-0094」「CVE-2014-0112」というStruts ClassLoaderの該当番号が含まれていない。

実は、InspectorはCVEリストの全てを診断対象にしているわけではない。AWSのInspectorのページで、診断対象のCVE番号が公開されており、番号は所々抜けている。

今後、補充されていくことが予想されるが、現時点ではCVEのリストにある重要な脆弱性を網羅しているとはいえない。

検証2　認証
SSHの脆弱性を正しく検出
パスワード有効期限は誤診断

認証の脆弱性診断についての検証に話を進める。Ubuntuサーバーへのリモートログインにおける SSH(Secure Shell)での認証で脆弱性のある方式を有効にしていないか、パスワードの運用に脆弱性はないか、という診断をInspectorで行う。

SSHについては、まず脆弱性のある「SSHプロトコル1」での通信を許可する設定にして、これを検出できるか試した(**図7**上)。すると、SSHプロトコル1での通信が許可されているという脆弱性を正しく検出した。

さらに、「SSHパスワード認証」を許可する設定にして、これを検出できるかどうかを試す。パスワード認証は総当たり攻撃で破られる可能性があり、リモートログインでは電子証明書を用いた認証が推奨される。そのため、SSHパスワード認証は脆弱性と見なされる。

Inspectorで診断したところ、SSHパスワード認証が許可されているとして、脆弱性を指摘した(図7下)。これは正しい診断結果である。

有効期限の値の有無だけを見る

パスワードの検証については、Ubuntuの初期設定のままで、リモートログインパスワードの最少文字数と複雑性(大文字・小文字・数字の混在など)のどちらも設定しない状態で、診断を行う。最少文字数と複雑性はどちらも、設定しないと脆弱性がある状態だ。

Inspectorによる診断結果は、どちらも脆弱性あり(**図8**上)。正しい診断結果を得られた。

パスワードではもう一つ、有効期限の設定も試した。有効期限が設定されていなければ、脆弱性があるといえる。有効期限はデフォルト設定

図6　Struts ClassLoaderの脆弱性の診断結果
Struts ClassLoaderの脆弱性は、正しく検出できなかった

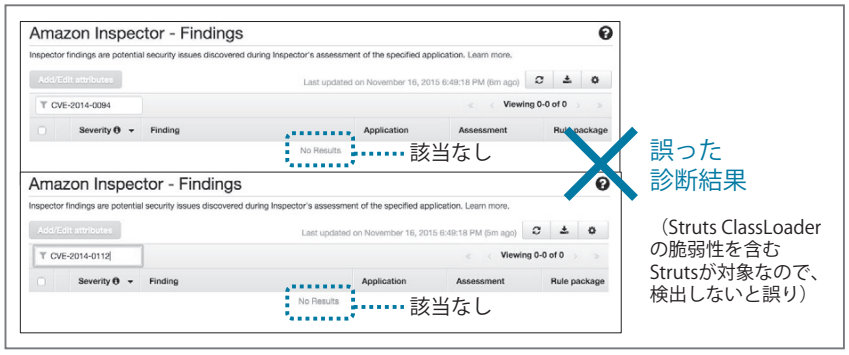


図7　SSH（SecureShell）での認証に関する脆弱性の診断結果
SSHでの認証に関する二つの脆弱性について、どちらも正しく検出した

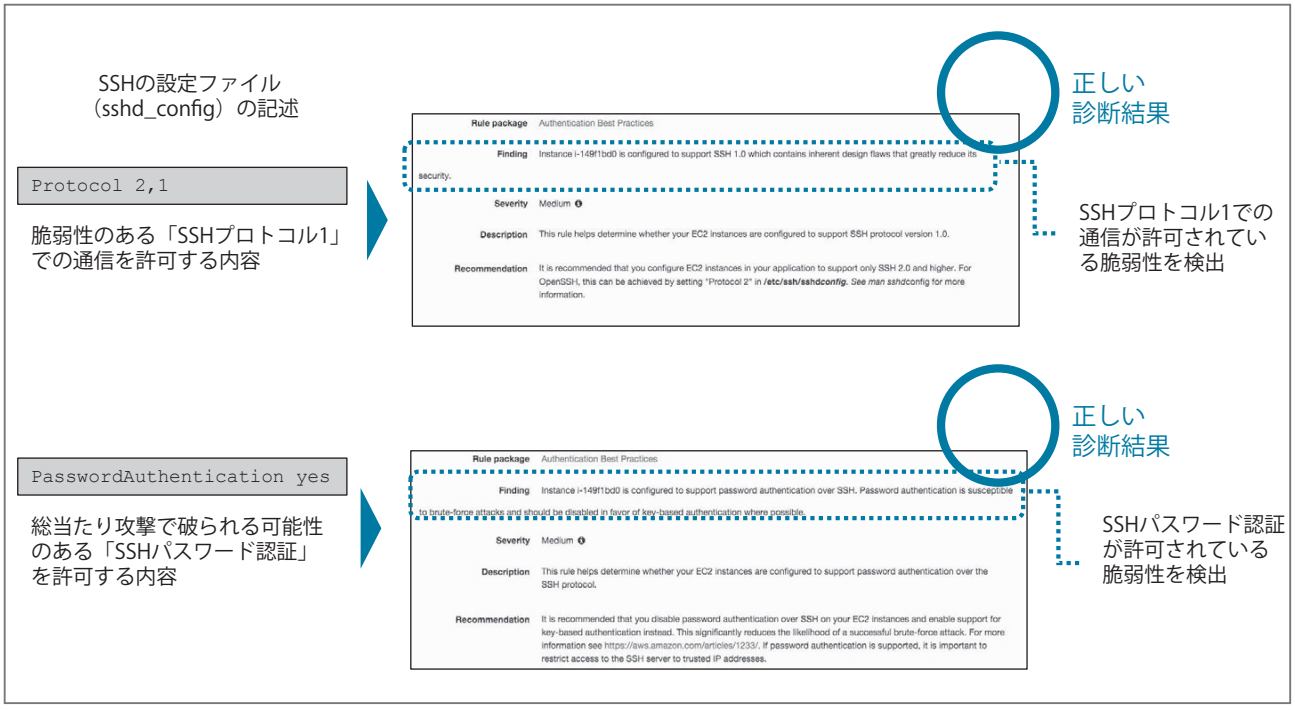
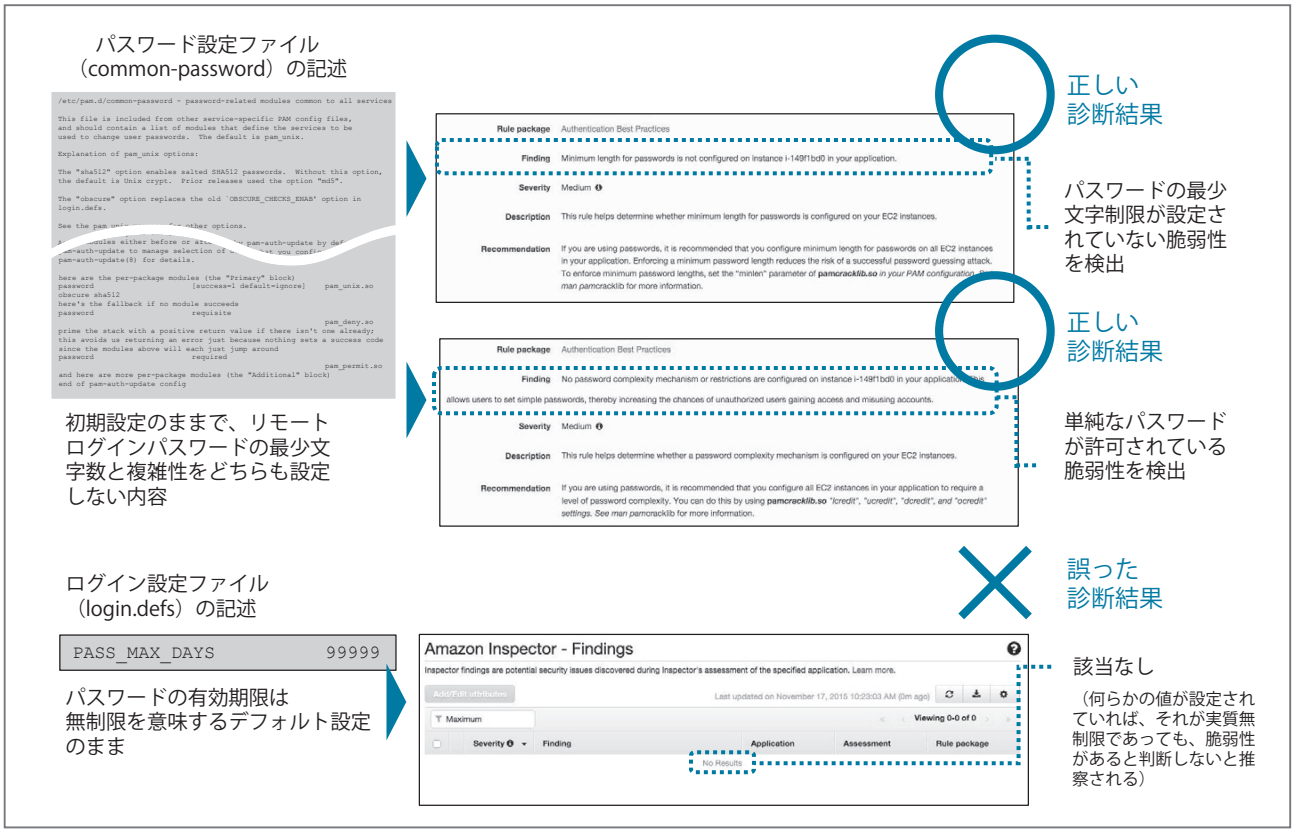


図8　認証のパスワードに関する脆弱性の診断結果
リモートログインパスワードの脆弱性については、有効期限の脆弱性を正しく検出できなかった



の99999日のままにした。99999という数値が設定されているが、無期限として扱われる。

Inspectorで診断したところ、この脆弱性を検出しなかった（前ページの図8下）。なおパスワード認証を有効にしたまま、有効期限を設定しない状態に変更し、改めてInspectorによって診断したところ、脆弱性ありという正しい結果が表示された。このことから、Inspectorは単純に有効期限の値が設定されているかどうかだけを見ていると推察される。

検証3　OSの認証・設定
システムファイルの編集権
一般ユーザーが持つも未検出

三つめとして、OSの認証・設定の脆弱性診断を検証する。診断対象の脆弱性は、メインメモリーのアドレス空間配置のランダム化「ASLR（Address Space Layout Randomization）」が無効になっていること、システムディレクトリーのファイルを一般ユーザーが編集できることの2点だ。

ASLRについては、無効になっていることをInspectorによって検出できた。正しい診断結果である（**図9**）。

一方、システムディレクトリーのファイルに対する編集権限の脆弱性は、Inspectorが正しく診断できなかった（**図10**）。

具体的には、一ひねりを加え、一般ユーザーをrootグループに加入させたうえで、binディレクトリー下にrootグループが編集権を持つファイルを配置。一般ユーザーがbinディレクトリー下のファイルを編集できる状況を作った。Inspectorはこの脆弱性を検出できなかった。複雑なルールには対応していないようだ。

図9　OSのアドレス空間の設定に関する診断結果
メインメモリーのアドレス空間のランダム化（ASLR）に関する脆弱性を、正しく検出した

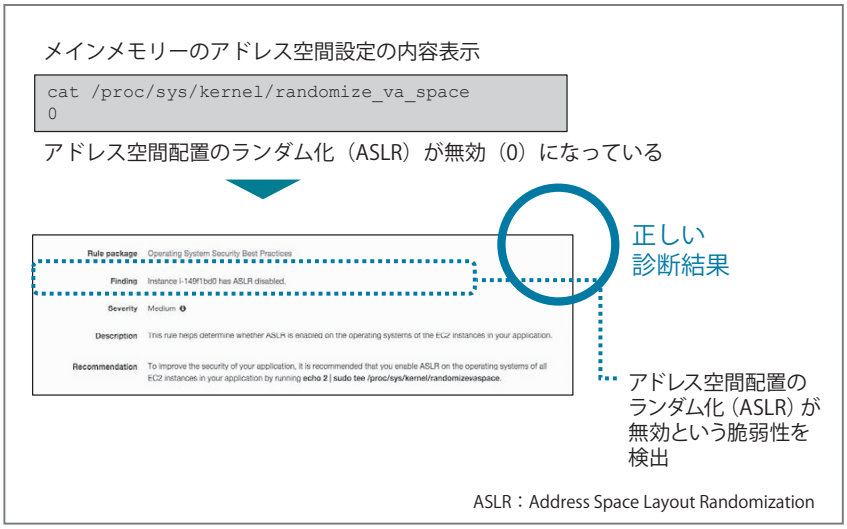
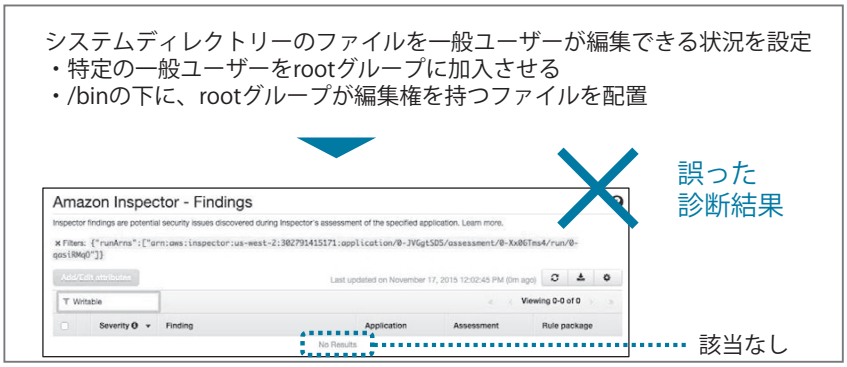


図10　OSのシステムディレクトリーに関する診断結果
システムディレクトリーのファイルを一般ユーザーが編集できるという脆弱性は、検出しなかった



診断実行までは専門知識が不要

最後に、Inspectorの利用の容易さについて報告する。

Inspectorを利用するには、事前の環境構築が必要である。具体的には、AWS IAM（Identity and Access Management）ロールの作成、診断対象リソースへのタグ付け、Inspector Agentのインストールなどだ。あとは、診断対象、診断ルール、所要時間を指定し実行すれば、結果が表示される。これらの作業は、AWSの基本知識があれば行える。

診断結果を読み解いて対処する場面でも、パッチ適用程度なら専門家でなくても実施できるだろう。

アーキテクトの眼

専門家診断の補完に有用 手間が小さく頻度高まる

吉田 浩和（よしだ・ひろかず）

アイレット cloudpack事業部
セキュリティエンジニア
エンタープライズ系のシステムインテグレータを経て2013年にアイレット入社。インフラ構築チームのリーダーを務めたあと、2014年4月からトレンドマイクロの統合セキュリティソフトDeep Securityを使ったセキュリティサービス「securitypack」の運用責任者。

AWSの「共有責任モデル」とは、仮想化レイヤー（物理層、ネットワーク、仮想マシン）の責任を米Amazon Web Servicesが持ち、それ以上のレイヤー（OS、ミドルウエア、アプリケーション、データ、通信の暗号化）の責任は利用者がそれぞれ持つというもの。これに照らすと、OSやミドルウエアに実装されるセキュリティはユーザーが責任を持つ部分だが、AWSはInspectorでこの部分に切り込んだ。

AWSは、金融情報システムセンターが策定した金融業界向けの「FISC 安全対策基準」など高いレベルのセキュリティ監査基準をクリアーしている。しかし、AWS上で稼働するシステムに脆弱性があれば、サイバー攻撃を防げない。

AWSではユーザーによって、実現しているセキュリティレベルは大きく異なるのが実情だ。セキュリティレベルを底上げするためにInspectorを投入したと考えられる。

Inspectorには、検証結果の通り、まだ改善の余地がある。しかし、セキュリティに銀の弾丸は存在しない。セキュリティの製品・サービスには、できること/できないこと、向き/不向きがある。それらを理解し、Inspectorを、外部の専門家による診断など、他の製品・サービスと組み合わせればよい。

開発期間中から診断する

Inspectorは手間やコストの面で、外部の専門家によるセキュリティ診断サービスより有利といえる。

一般に専門家によるセキュリティ診断サービスは、システム開発プロジェクトの終盤で1回、運用期間中は四半期から1年に1回だけ行うケースが多い。Inspectorを使えば、セキュリティ診断の頻度が高まる。例えば開発期間中のインフラ構築完了時とシステムテストの前後に診断を実施することによって、セキュリティについての手戻りを減らせる。

もちろん、Inspectorだけで済むわけではない。専門家によるプラットフォーム診断やWebアプリケーション診断は、Inspectorとは対象や観点が違う。別途行う必要がある。

開発時での活用について述べたが、運用に入ったあと継続的に行うセキュリティ診断でも、Inspectorは有用だ。Inspectorが診断に使うルールは、随時更新されるためである。

最後に、使いこなしのコツを紹介する。

Inspectorの診断結果は画面のFinding欄に表示され、検出した脆弱性ごとに、属性情報を付与できる。ここに担当者や対応状況を記載すると管理しやすい。検索ウインドウに「アサイン済み」などのキーワードを入れると、検出した脆弱性のうち該当するものが一覧表示される。

ただし、設定した属性情報はAWSコンソールからは確認できない。登録するキーワードはあらかじめチームで決めておいたほうがよい。

対象サーバーの台数が多くなった場合、検出した脆弱性への対処の管理が煩雑になる。AWS コマンドラインインターフェイス（CLI）を駆使すればこの問題にある程度対応できるが、サードパーティーも含めて管理ツールの登場を待ち活用したい。