2003－CSEC－23　(1)
2003／12／19

シェルコード解析による不正アクセス検出手法

北條　孝佳[1)]　佐久間　英夫[2)]　種茂　文之[2)]
{hojo.takayoshi, sakuma.hideo, tanemo.fumiyuki}@lab.ntt.co.jp

あらまし　インターネットの重要性が高まる中、システムに被害を与える不正アクセスが頻出している。不正アクセスのうちシステムを乗っ取るなど悪質なものは、バッファオーバフローと呼ばれるセキュリティホールを攻撃するものが大半を占めている。このバッファオーバフローを利用した攻撃をネットワーク上で検出する既存技術である侵入検出システムの多くは、シグネチャマッチングと呼ばれる検査手法によりパケットが不正アクセスかどうかの判定を行う。しかし、この検査手法では既知のバッファオーバフローを利用した攻撃に対しては検出することが可能であるが、未知のセキュリティホールに対する攻撃を的確に検出することは不可能である。未知のバッファオーバフローを利用した攻撃を確実に防御するため、我々はネットワークを流れるパケットをマシン語の命令列として解析することにより、不正アクセスを検出するという新しい手法を提案する。また、本提案手法を実装したプログラムを用いて評価試験を行ったのでその結果も報告する。

Shellcode Analysis Intrusion Detection Method

Takayoshi Hojo[1)]　Hideo Sakuma[2)]　Fumiyuki Tanemo[2)]

Abstract　The importance of the Internet increases, and so does the damage caused by illegal access . Most security attacks such as system takeover exploit a security flaw called buffer overflow. To detect such attacks on the network, many IDS (Intrusion Detection Systems) use a matching method, which compares the data segment of each network packet with a list of signatures. Each signature defines a specific packet pattern in a known security attack. However, such matching method cannot detect attacks exploiting unknown buffer overflow problems. In order to defend against unknown buffer overflow attacks , we present a new detection strategy which detects illegal access by analyzing data segment of network packet as a machine language code and validating code by executing it in a virtual machine. We also present the evaluation results of our method.

1　はじめに

昨今、様々な情報提供や生活の利便性向上、電子商取引等のインフラとしてインターネットの重要性が高まっている中、システムに被害を与える不正アクセスが頻出している。実際、OSやアプリケーションのセキュリティホールが多数発見されており、これらを悪用してネットワークから不正アクセスを行う攻撃ツールやワームが作成され使用されることにより、多大な被害をもたらしている。このような攻撃から守るには、不正アクセスを的確に検出することが重要である。特に不正アクセスのうちシステムを乗っ取るなど悪質なものは、バッファオーバフロー（以降「BOF」と称する）と呼ばれるセキュリティホールを攻撃するもの（以降「BOF攻撃」と称する）が大半を占めているため、BOF攻撃を的確に防御できる技術が求められている。

近年、ネットワークを利用した不正アクセスを検出する目的で、多くの企業がネットワーク侵入検出システム

1)　警察庁
National Police Agency
2)　日本電信電話株式会社
情報流通プラットフォーム研究所
NTT Information Sharing Platform Labora-tories

（IDS）[1]を導入している。多くのIDSはシグネチャマッチング方式を採用しており、攻撃の特徴をシグネチャとしてデータベースに登録しておき、ネットワークを流れるパケットをシグネチャと比較することにより攻撃を検出する。この方式では対応するシグネチャが存在する攻撃については高い検出率を示すが、未知のセキュリティホールへの攻撃や、既存のセキュリティホールへの攻撃であってもシグネチャに登録されていないものには対応できないという問題点がある。

ネットワーク上で不正アクセスを検出する別の方法として、異常検出方式と呼ばれる統計や学習により攻撃を検出する手法もあり、この方式により未知の攻撃も検出できる可能性がある。しかし、検出精度を高めるための調整が難しく、誤検知率が高い等の問題がある。

BOF攻撃対策としては、サーバ上でファイルや動作しているプロセスを監視する等を行い、攻撃を検出する手法も存在する[2][3]。しかし、この手法はBOF攻撃を行うパケット（以下「攻撃パケット」と称する）がサーバに到達し、BOFが発生して初めて攻撃を検出することができる手法であり完全な防御ができないこと、またサーバ全てに対策を施さなくてはならず管理上の手間がかかる等の問題がある。

そこで本稿では、BOF攻撃を確実に検出するため、ネットワークを流れるパケットをキャプチャし、そのデータ部分をマシン語の命令列として解析を行うことで、パケットが不正アクセスに関わるかどうかを正確に判定する新しい不正アクセス検出手法を提案する。これにより、未知のBOF攻撃も確実に検出することが可能となる。

本稿の構成は以下の通りである。第2章では、BOF攻撃に対する既存の検出手法の問題点について分析し、BOF攻撃をネットワーク上で検出する我々の提案した不正アクセス検出手法について説明する。第3章では、本提案手法の実装について説明し、第4章では、実装したプログラムに対して模擬的なBOF攻撃を送出して検出させた結果を示し、最後にまとめを行う。

2 提案手法の概要

2.1 BOF を利用した攻撃の概要

不正アクセスに使われる攻撃ツールやワームの多くはBOFと呼ばれるセキュリティホールを利用して、任意の動作を行わせることが可能になるコード（以下「シェルコード」と称する）を攻撃対象となるサーバに送り込む。図1にIntel製32ビットCPUのサーバに対するBOF攻撃の例を示す。

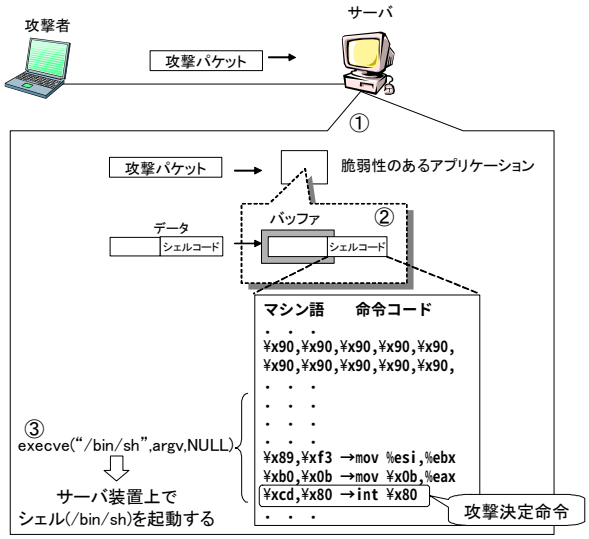

図 1　バッファオーバフロー攻撃（BOF攻撃）の例

このBOF攻撃は次のようなステップを実行することでサーバ上のアプリケーションの乗っ取りを行う。

① アプリケーションの入力値としてシェルコードを含むデータが渡される。
② BOFが発生してシェルコードに制御が移動する。
③ シェルコードを実行することで、サーバ上でシェルの起動等の任意のコマンドが実行される。

ここで、シェルコードはマシン語命令列から構成されており、サーバ上でマシン語命令列が逐次実行され、その結果、シェルの起動やファイルの改竄・削除等、攻撃者が意図する行為が行われるようにCPUのレジスタやメモリが書き換えられる。そして実行過程の最後に処理される命令（以降「攻撃決定命令」と称する）を実行することにより攻撃は完了する。

2.2 既存のシグネチャマッチング方式での問題点

前述したIDSのシグネチャマッチング方式によりBOF攻撃を検出するには、攻撃パケットに含まれる固有のパターンを記述したシグネチャを作成して登録することが必要である。Snort[4]のBOF攻撃に対するシグ

ネチャは主に、

① セキュリティホールに対する攻撃パケットの固有な部分

② シェルコードの特徴的な部分

に関わるものが登録されているが、①は特定のセキュリティホールに関する部分のため、未知のセキュリティホールは検出できない。そのため、②のシェルコードの特徴的な部分を比較することにより未知の攻撃等を検出することも可能であるが、シェルコードのパターンは無限に作成可能であるため、原理的にシグネチャマッチング方式では検出不可能になる。また、特徴的な部分を検出するためにシグネチャの長さを短くすると、正常なパケットの中にシグネチャと同じパターンが含まれていれば、誤検出になってしまうという問題も存在する。

2.3 本手法の着眼点

本手法はシェルコードに含まれている攻撃決定命令に着目して不正アクセスを検出する。しかし、攻撃決定命令のみを発見するだけでは、誤検出の可能性があることや攻撃決定命令を変形することが可能であるため、ネットワークを流れるパケットをマシン語命令列として解析を行い、攻撃決定命令を発見した際にCPUのレジスタの状態を取得してシェルコードかどうかを判定することにより、様々な種類のシェルコードを検出することができる。

なお、本手法はBOF攻撃特有のシェルコードに着目した検出手法であり、BOF攻撃以外の不正アクセスには適用できない。また、BOF攻撃であってもDoS（サービス停止）攻撃のようにシェルコードを含まない場合には検出することができない。さらに、サーバのOSやアプリケーションのバージョンに対して一意的に決まるメモリやレジスタの状態を利用するような特殊なBOF攻撃については検出できない。我々の提案する検出手法は、システムを乗っ取ることを目的とした汎用性の高いBOF攻撃を対象としており、他の攻撃を検知・防御するためには、既存のIDS等と併用する必要がある。

3 本手法の構成と実装

3.1 本手法の構成と検出手順

本手法の構成を図２に示す。

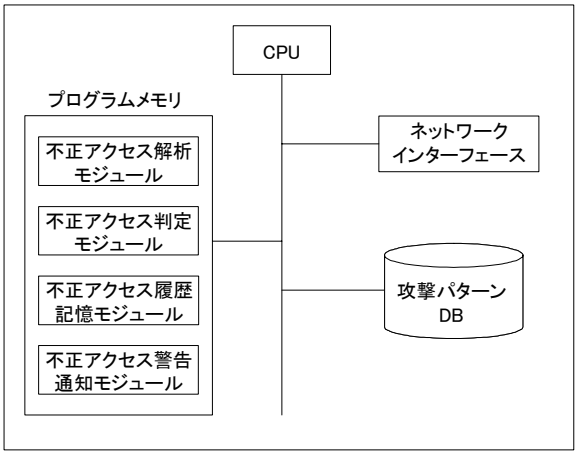

図2 提案手法の構成

本手法の特徴であるパケットを解析して不正アクセスを判定する機能として、不正アクセス解析モジュール、不正アクセス判定モジュール、攻撃パターンデータベースがある。また、不正アクセスと判定された場合にログを保管し、管理者に警告する IDS 相当の機能として、不正アクセス履歴記憶モジュール、不正アクセス警告通知モジュールも併せ持つ。以降、これらのモジュールについて説明する。

不正アクセス解析モジュール

不正アクセス解析モジュールは、ネットワークインターフェースでキャプチャしたパケットのデータ部だけを取り出して、マシン語の命令列として擬似的に実行していくモジュールである。

この不正アクセス解析モジュールには、擬似的に実行されるマシン語命令を監視する命令監視プログラムが含まれる。この命令監視プログラムが攻撃決定命令を発見すると攻撃決定命令とCPUのレジスタの状態を不正アクセス判定モジュールに解析結果として出力する。

不正アクセス判定モジュール

不正アクセス判定モジュールは、不正アクセス解析モジュールから受け取った解析結果が不正アクセスかどうかを判定するモジュールである。この解析結果の攻撃決定命令をキーとして攻撃パターンデータベースから攻撃情報を検索し、解析結果である CPU のレジスタの状態と一致する攻撃情報を取得する。解析結果と取得した攻撃情報が一致しなければ、シェルコードではないと

判定し、不正アクセス解析モジュールにより次の命令を解析する。解析結果が攻撃情報と一致した場合は、シェルコードが含まれる攻撃パケットであると判定して、パケット情報と攻撃情報を不正アクセス履歴記憶モジュールと不正アクセス警告通知モジュールに出力する。

攻撃パターンデータベース

攻撃パターンデータベースは攻撃情報が格納されているデータベースである。攻撃情報には、表1に示すように攻撃決定命令、CPU のレジスタの状態、実行される命令、攻撃の名称が記述されている。

表 1　攻撃パターンデータベース例

攻撃決定命令	レジスタ eax	レジスタ ebx が指すアドレス値	実行される命令	攻撃の名称
¥xcd¥x80	¥xb	/bin/sh	sh	シェルが起動
¥xcd¥x80	¥x17	ebx=¥x0	setuid	root 権限に移行

不正アクセス履歴記憶と警告通知モジュール

不正アクセス履歴記憶モジュールは、不正アクセス判定モジュールから受け取ったパケット情報と警告情報をファイルに保存する。また、不正アクセス警告通知モジュールは不正アクセス履歴記憶モジュールで作成されたファイルの情報をコンソール画面に警告情報として表示したり、管理者にメールを送信したりする。

3.2　提案手法の実装方法

本手法の主要部分である不正アクセス解析モジュールを実装する方法として 2 種類考案した。

第一の方法は、擬似的にプログラム上で CPU とメモリを模した CPU シミュレーションプログラムと擬似メモリを構成し、その上で命令コードを実行させて状態の監視を行う CPU シミュレーション方式である。CPU シミュレーション方式は CPU レジスタを模した擬似レジスタと CPU の命令実行動作を模した擬似命令実行プログラムから構成される。また、攻撃決定命令が擬似命令実行プログラムで実行される前に発見するための命令監視プログラムを実装しており、次に実行する命令が攻撃決定命令かどうかを監視する。この方法は、CPU アーキテクチャに依存することなく動作させることができ、実際のシェルコードを処理することなく解析が行えるため、危険を伴わない。しかしながら、全ての CPU 命令を実装する必要がある等の手間がかかる。

第二の方法は、本手法が構成されるホスト上の実 CPU 及び実メモリを用いることにより実現する。また、命令監視プログラムは実 CPU 上で実行される命令を常に監視し、実行前に攻撃決定命令を発見することが可能である。

第二の実 CPU を用いる方法は、CPU シミュレーションプログラムを作成しなくても不正アクセス解析モジュールが正しくシェルコードを解析できるため実現が容易であるが、監視すべきサーバの CPU アーキテクチャと一致していなければならない、実際に命令を実行するため予期しない異常を引き起こす危険性がある、一命令毎に監視する動作処理に大変な負荷がかかる、という問題がある。特に動作処理負荷は非常に大きいため、今回の実装では、第一の CPU シミュレーション方式を用いることとした。

3.3　実装上の課題

本手法を実装するにあたり、いくつかの考慮すべき課題があったため、説明する。

3.3.1 解析開始位置

シェルコードの開始位置は攻撃によって異なるため、データ部の先頭から開始されているとは限らない。本手法で攻撃パケットを解析するためには開始位置を探索することが必要であり、これを実現するためには以下の 2 通りの方法が考えられる。

1. 解析開始位置を 1 バイトずつ移動する方法（総当り方法）
   データ部の先頭から解析を実行し、攻撃決定命令がなければ先頭から 1 バイト移動した位置を解析開始位置とし、解析を実行する。その様にして開始位置を 1 バイトずつ移動し、パケットの最後が開始位置になるまで解析を行う。
2. NOP 及び擬似 NOP を探索して実行する方法
   汎用性が高いシェルコードは NOP あるいは擬似 NOP をシェルコードの開始位置の手前に配置する。これは、攻撃者が想定したシェルコードの配置と異なる場合においても、攻撃が成功する可能性を高めるためである。ここで NOP（¥x90）とは何も実行しない命令であり、擬似 NOP はレジ

スタやメモリの値の変更等を行うが、シェルコードに直接関わらず意味がない命令のことを言う（例えば inc %eax など）。この NOP 及び擬似 NOP の連続している箇所を発見し、その命令が存在する全ての位置を開始位置として解析を行う。

1の方法は、パケット毎にデータ長回数だけ解析を行う必要があり、解析処理負荷が高くなる。また 2 の方法は、シェルコードの開始位置の手前に我々が設定していない擬似 NOP が配置された場合には、解析を行わない恐れが存在するため、1 の方法を採用している。

3.3.2 無限ループ

シェルコードの作りによっては有限回ループを行うものがあるので、ループ命令を処理する必要がある。しかし、例えば FTP 通信などのバイナリデータが入ったパケットを命令列として解析してしまうと、組み合わせによっては無限ループになる可能性があるが、無限ループが発生する攻撃パケットは存在しないため、処理する必要がない。

このため、ループ命令が実行される際、ループが終了する条件を満たす処理が含まれているかどうかを監視するための無限ループ監視プログラムを追加して、無限ループであると判断される場合には処理を中断し、パケット解析を先に進めるようにした。

また、上記方法を実装しても無限ループを 100%回避することは不可能であるため、ループ回数の上限値を設定し、その設定値以上のループが発生した場合、警告ファイルを作成し、別途解析を行うようにしている。

4 評価

4.1 試験方法

上記の実装によりプログラムを試作して、模擬的環境内においてBOF攻撃を行い、正しく検出できるかどうかの評価試験を行った。また、攻撃パケットはフラグメント化せず、TCPの1パケットでシェルコードを実行させてシェルを起動させることができるものである。

4.1.1 攻撃対象とするサーバ

Intel製32ビットCPUを用いたマシンにOSがRedHatLinux9をインストールしたサーバ上で、BOFのセキュリティホールを持ったサーバプログラムを動作させた。このサーバプログラムは、クライアントから受信した文字列に対してhelloを付けて返答するという単純なプログラムであるが、クライアントから受信する際に受け取る文字列長を確認していないため、長い文字列を受信するとBOFが発生し、シェルコードが実行されるという脆弱性が存在する。

4.1.2 攻撃方法

上記の攻撃対象とするサーバに表2のような攻撃パケットの送出を行い、試作プログラムを動作させた状態で検出可能かどうか測定を行った。なお、表2で①～⑥の攻撃はシェルコードを含むBOF攻撃であり、⑦、⑧はシェルコードに見せかけた攻撃であり、実際にシェルを起動させるものではない。

攻撃時には、比較のためにSnortやNGSec[5]のIDSも併せて動作させて検出の確認を行った。Snortはシグネチャマッチング方式を使っているが、シェルコードに含まれる可能性が高い部分とのマッチングを行うことにより、一部の未知の攻撃を検出できるようにしている。シグネチャファイルとして2003年11月10日現在のものを使用した。NGSecはNOP及び擬似NOPが60個連続して存在するパケットを検出してシェルコードだと判断している。NGSecのシグネチャは2002年1月21日のものを使用した。

表 2　攻撃及び偽攻撃パケットの種類
( )内の数字は連続する数

No.	シェルコードの構成
①	NOP(30)+shellcode+NOP(208)
②	NOP(30)+**変形**shellcode+NOP(208)
③	NOP(30)+**変形**(shellcode+NOP)
④	**擬似**NOP(32)+shellcode+**擬似**NOP(154)
⑤	**擬似**NOP(32)+**変形**shellcode+**擬似**NOP(154)
⑥	**擬似**NOP(32)+**変形**(shellcode+**擬似**NOP)
偽攻撃の構成	
⑦	NOP**のみ**80**バイト**
⑧	**擬似**NOP**のみ**80**バイト**

4.2 検出結果

検出結果を表３に示す。○は検出、×は検出しない、△は誤検出を示している。

表 3 検出結果

No.	**本手法**	Snort	NGSec
BOF**攻撃(含シェルコード)**			
①	○	○	○
②	○	○	○
③	○	○	×
④	○	×	○
⑤	○	×	○
⑥	○	×	×
偽攻撃			
⑦	×	△	△
⑧	×	×	△

Snort は NOP（¥x90）が 14 バイト連続で配置されている場合に検出するシグネチャが存在するため、NOP が連続して配置されているシェルコードは検出しているが、擬似 NOP は検出できない。NGSec も NOP 及び擬似 NOP が連続 60 個以上配置されているものは検出できるが、変形されたものは検出できない。また⑦、⑧は、誤検出になっている。

試作プログラムにおける攻撃①の検出時のログの状況を図4に示す。攻撃が成功した際に保存されるログにもきちんとパケット情報と攻撃情報が保存されていることが分かる。

5 まとめ

本手法を実装した試作プログラムの評価により、既存のシグネチャマッチング方式のIDSでは検出できなかった未知のBOF攻撃を確実に検出することが示された。

この手法を既存のIDSに組み込むことで、より広範囲の不正アクセスを検出できるようになる。さらにファイアウォールとの連携により、攻撃パケットを検出後パケットの破棄を行うことも可能になる。

今後、無限ループ解析機能等のプログラムへの作りこみを進め確実にBOF攻撃が検知できるようにするとともに、多様なトラフィックパターンでの試験を実施して、本方式の有効性を確認していく。

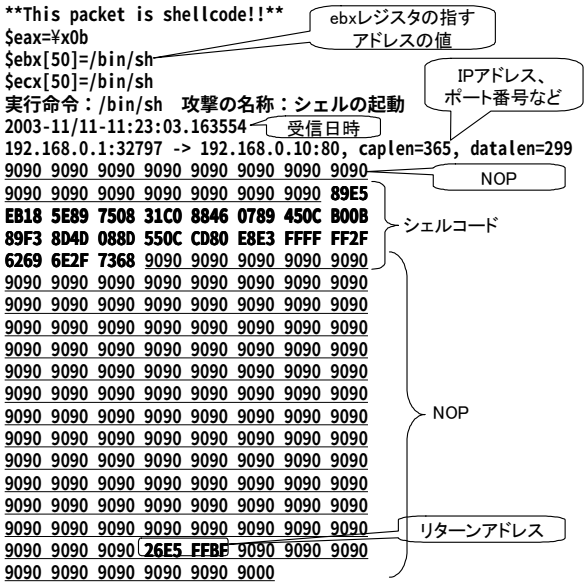

図 3 攻撃検出時の出力結果

参考文献

[1] 武田圭史，磯崎宏：ネットワーク侵入検出，ソフトバンク パブリッシング（株） pp33－pp108

[2] Tripwire, Inc., http://www.tripwire.co.jp/

[3] 江藤博明，依田邦和，“propolice：スタックスマッシング攻撃検出手法の改良”，情処研報，2001－CSEC－14，pp.181－pp.188，July 2001.

[4] Snort, http://www.snort.org/

[5] NGSecTools, http://www.ngsec.com/ngresearch/ngtools/

[6] 三輪信雄，新井悠：ネットワーク攻撃詳解，ソフト リサーチ センター（株）

[7] 塚越一雄:Linux システムコール, 技術評論者(株)

[8] UNYUN：ハッカー・プログラミング大全，データハウス（株）