カーネルメモリを利用した命令ポインタの退避/復元*

3L−1

岩村 誠† 村岡 洋一‡
早稲田大学大学院理工学研究科§

1 はじめに

計算機への攻撃方法としてbuffer overflowをついた手法がある。これはスタック上の配列をoverflowさせることによって、同じくスタック上に存在する命令ポインタを書き換える。こうして攻撃者はあらかじめメモリ上に配置しておいたプログラムに命令ポインタを設定しプロセスの制御を奪う。このbuffer overflowをついた攻撃を防ぐツールはいくつか存在し、大別すると2通りに分けられる。1つは命令ポインタと配列の間に「壁」を置きbuffer overflowを検出するもの、もう1つはデータセグメント上に命令ポインタを退避するものである。これらはbuffer overflowをついた攻撃に対しては、効果的であったが2000年6月に報告されたwu-ftpdのセキュリティホールをついた攻撃には全く無力であった。この攻撃はformat string attackと呼ばれ、任意アドレスのユーザメモリに対してピンポイントで値を書き込むことが可能である。よって、攻撃者は「壁」をすり抜け命令ポインタだけを書き換えたり、命令ポインタの退避先の値を書き換えられるようになった。

そこで、本稿ではformat string attackやスタック上命令ポインタを狙った未知の攻撃方法に対する防御手段として、スタック上命令ポインタのカーネルメモリへの退避/復元を行う手法を提案する。またこの環境下で実際にいくつかのプログラムを動かした実験結果について述べる。

2 命令ポインタ保護の仕組

本手法では「ユーザモードではカーネルメモリにアクセス出来ない」というマルチタスクOSの特徴を利用している。具体的にはタスク(プロセス、スレッド)を保護するために、そのタスク自身が関数開始直後にスタック上命令ポインタをカーネルメモリに退避し、関数終了直前にカーネルメモリからスタック上命令ポインタを復元する。

正当なタスクはシステムコールによりカーネルモードへ移行し、ユーザタスクのスタック上命令ポインタを退避/復元する。これに対して攻撃者はタスクの制御を奪うことが目的であるが、その目的を達成するにはユーザスタック上命令ポインタを書き換える必要がある。それにはシステムコールを呼び出しカーネルメモリ上にある退避された命令ポインタも書き換えなければならないが、タスクの制御を奪わない限りシステムコールを呼ぶことは出来ないため、結果攻撃は不可能であることが分かる。

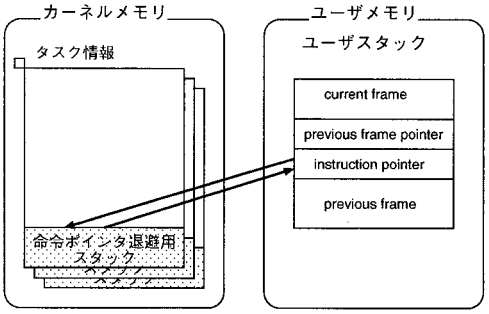

図1: 命令ポインタのカーネルメモリへの退避

2.1 カーネル

拡張を施したOS	Linux 2.2.18 for i386
新システムコール	(スタックへの退避) sys_kmem_push (スタックからの復元) sys_kmem_pop

カーネルメモリ上に命令ポインタ退避用のスタックをタスクの数だけ用意した。またこのスタックに対して退避/復元するためのシステムコールを実装した。復元の際は、ユーザスタック上命令ポインタと退避されている値を比較し、異なるときにはその旨

*Protecting the instruction pointer on the stack using kernel memory
†Makoto Iwamura
‡Yoichi Muraoka
§Graduate School of Science and Engineering, Waseda University

が syslog に出力される。

2.2 Cコンパイラ

拡張を施したコンパイラ	gcc 2.95.2

命令ポインタの退避の契機は関数開始直後、復元の契機は関数終了直前である。これらの退避/復元を行うためのシステムコール呼び出しを、アセンブラコードとして出力するためにコンパイラを拡張した。これにより、既存Cソースをこの拡張を施したコンパイラでコンパイルし直すことで、本手法の保護機能を備えたプログラムを自動的に生成出来るようになった。

こうして出来たプログラムは新システムコールを持たない Linux for i386 環境でも通常通り動作する。

3 実験

以下の環境で各実験を行った。

CPU	PentiumIII 550MHz
RAM	1024MB
OS	Linux 2.2.18 + 拡張 (2.1 参照)
コンパイラ	gcc 2.95.2 + 拡張 (2.2 参照)

3.1 攻撃耐性

セキュリティホールが存在するいくつかのプログラムに対して実際に攻撃し、本手法による保護機能の有効性を検証した。攻撃結果は以下のとおりとなった。

プログラム	保護機能無	保護機能有
named	got root shell	detected
in.ftpd	got root shell	detected

- named
  ICS Bind8 Transaction Signature Buffer Overflow Vulnerability
  2001/1/29 に報告された buffer overflow の脆弱性を持つプログラム

- in.ftpd
  Wu-Ftpd Remote Format String Stack Overwite Vulnerability
  2000/6/22 に報告された format bug の脆弱性を持つプログラム

実験により buffer overflow や format bug をついた攻撃が検出されることを確認出来た。

3.2 オーバヘッド

本手法による保護機能が速度面でどれ程のオーバヘッドを生むかを試した。実験結果は以下のとおりとなった。

プログラム	保護機能無	保護機能有	OverHead
gzip	136.6[sec]	166.5[sec]	21.9%
bzip2	144.3[sec]	186.1[sec]	28.9%

- gzip
  最高圧縮 (-9 オプション) にて 80MB のファイルを圧縮

- bzip2
  最高圧縮 (-9 オプション) にて 80MB のファイルを圧縮

関数呼び出しの度にシステムコールを 2 回 (退避と復元) 呼び出すため、その際に必要となるパラメータの設定やユーザモードからカーネルモードへの移行などで、20〜30%程度のオーバヘッド増加が確認された。

4 まとめ

malloc/free の取り扱いミスなど極めてそのプログラムに特化し、一般的な保護ツールが存在しない (個別にセキュリティパッチを当てる必要のある) ようなセキュリティホールをついたスタック上命令ポインタを書き換える攻撃からも、プロセスを保護出来る方法を提案した。また、実際にカーネルメモリにスタック上命令ポインタを退避することで、buffer overflow をついた攻撃や format string attack からスタック上命令ポインタを保護出来ることを実証した。

ただ、ピンポイントでのフレームポインタや関数ポインタを狙った攻撃からタスクを守れないこと、またシステムコールを呼び出すことによる大きなオーバヘッドは今後の課題として残っている。

参考文献

[1] "http://immunix.org/stackguard.html", StackGuard.

[2] "http://www.angelfire.com/sk/stackshield/", Stack Shield.