第2部

第2章 スタック・オーバフロー/メルトダウン/スペクタ…攻撃の手法から防ぐ手段まで

CPUへの攻撃とセキュリティの仕組み

永原 柊

最近，情報漏えいやPCが勝手に暗号化されるなど，サイバー・セキュリティの被害を聞くことが増えました．これらの問題の中には，CPUが不正なプログラムを実行することで起こるものがあります．そこで，CPUにセキュリティ対策を導入して被害を防ごうという取り組みが行われています．ここでは，そのセキュリティ対策に関するQ&Aを取り上げます．

2-1　セキュリティの仕組み

Q1　スタック・オーバフロー攻撃

Q サイバー攻撃にはどのようなものがある？
A 古典的な代表例としてスタック・オーバフロー攻撃がある

サイバー攻撃の種類はさまざまであり，また新しい攻撃が生み出され続けています．古典的なサイバー攻撃の1つに，スタック・オーバフロー攻撃があります．プログラムの作り方によっては，攻撃者が外部から任意の攻撃プログラムを送り込むことができる，大変危険な攻撃です．

● スタックの仕組み

攻撃対象となるスタックがどのようなもので，どのように使われるのか説明します．

CPUからアクセスできるアドレス空間内には，**図1**のようにプログラムを格納した領域，外部変数など静的に確保される（プログラム実行中にサイズが変わらない）領域，プログラム実行中にオブジェクト作成など動的に確保される（プログラム実行中にサイズが増減する）ヒープ領域，動的に確保されるスタック領域があります．

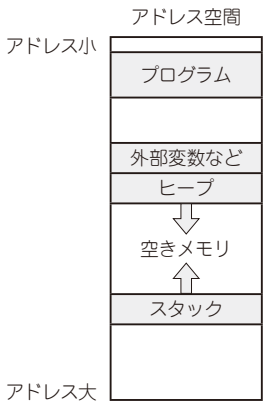

図1
メモリ内で確保される領域
スタックはアドレスの大きい方から小さい方に向かって動的に確保される

動的な領域はサイズが増加する場合，ヒープ領域はアドレスが小さい方から大きい方に伸び，スタック領域はアドレスが大きい方から小さい方に伸びます．

スタックはもともと，関数呼び出しを行ったときの戻り先を記録するものです．便利に使えるので，さまざまな用途に用いられています．ここでは戻り先の記録とローカル変数の保存に絞って説明します．

● 戻り先を記録する

スタックに戻り先を記録した様子を**図2**に示します．関数A内から関数Bを呼び出したとき，関数Bの実行が終了したときの戻り先をスタックに記録します．

さらに関数B内から関数Cを呼び出したとき，関数Cの実行が終了したときの戻り先をスタックに記録します．

記録を進めると，スタックはアドレスの大きい方から小さい方に伸びていきます．

▶ローカル変数を保存する

関数Aから呼び出した関数BがC言語で**リスト1**のようになっていた場合，通常は**図3**のようにローカル変数bufの領域もスタック上に確保されます．つまり，関数A内から関数Bを呼び出した時点で関数Aの戻り先を記録し，次に関数Bを実行するとローカル変