Monthly Research

Windows 8 – 脆弱性緩和機能の強化

Fourteenforty Research Institute, Inc.
株式会社 フォティーンフォティ技術研究所
http://www.fourteenforty.jp

Ver2.00.01

背景：脆弱性攻撃の可能性を減らすために

- 最近のオペレーティングシステムは、
  脆弱性攻撃が行われた時にそれが脅威に結びつかないよう
  緩和する機能を搭載している
  - XP SP2 のデータ実行防止 (DEP と SafeSEH)
  - Vista 以降のアドレス空間のランダム化 (ASLR)
- Windows 8 とそれに対応したコンパイラである Visual C++ 2012 では
  これらの脆弱性緩和機能が大幅に強化された
  - 脆弱性攻撃の一部を無効化 / 緩和することができる

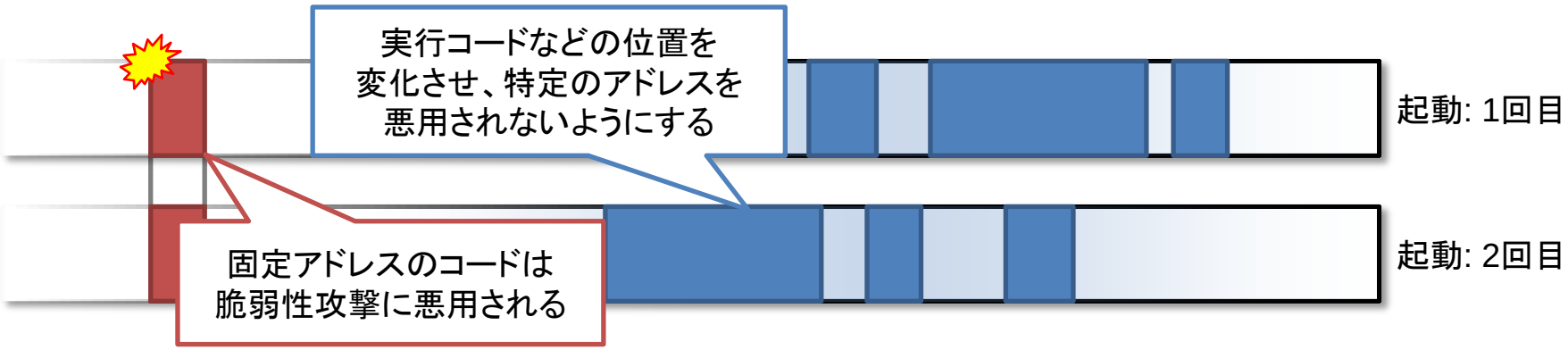
概要

- 強化された Windows 8 の脆弱性緩和機能の紹介
  - Visual C++ コンパイラの改良
  - ASLR の強化
  - ヒープ保護の強化
  - カーネル内部の強化
- 今後考えられる攻撃
  - 緩和機構の回避
  - Type Confusion による攻撃
- まとめ

Visual C++ コンパイラの改良：開発者のために

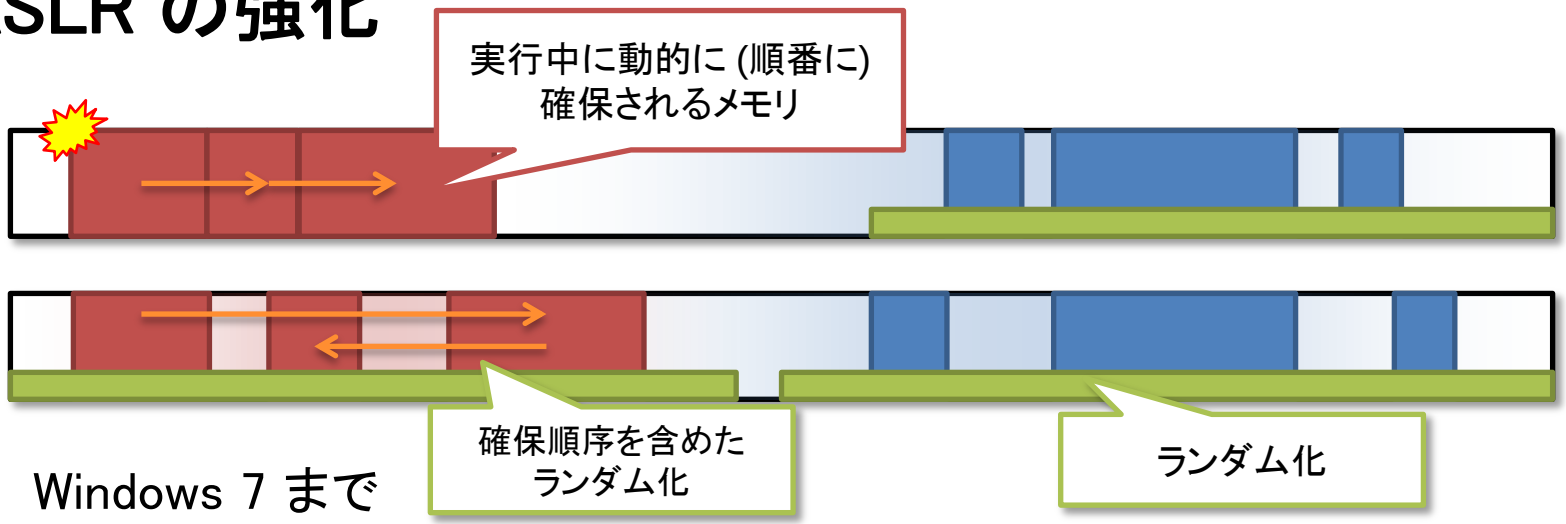
- 最新のセキュリティ機能に対応し、バッファオーバーフローなどの被害を最小限に抑えられるようになっている
  - スタックバッファオーバーフローの防止
    (Enhanced /GS – Visual C++ 2010 以降)
  - C++ オブジェクトへの参照を悪用する攻撃を緩和しつつ高速化
    (Sealed Optimization – Visual C++ 2012 以降)
  - C++ オブジェクトの (オーバーフローなどによる) 改ざん防止
    (Virtual Table Guard – Visual C++ 2012 以降)
- Windows 8 の新しい脆弱性緩和機能を有効に活用できる
  - Visual C++ 2008 以降で推奨されるコンパイラ / リンカーオプション
    /GS, /NXCOMPAT, /DYNAMICBASE, /*SAFESEH* (x86 のみ)
  - これらは (おおむね) デフォルトで有効になっている

ASLR とは



- アドレス空間のランダム化 (Address Space Layout Randomization)
  - 攻撃者は DEP などの古い緩和技術を回避するため、“特定のアドレスにある” プログラムやデータを悪用するよう変化
  - ROP (Return-Oriented Programming) の利用が目立つように
  - ASLR はプログラムやデータのメモリ上での位置を起動ごとに変更することによって、脆弱性攻撃が成功する確率を減らす
- Windows Vista で実装された
  - ASLR のない XP は最近の脆弱性攻撃に対してほとんど無力

ASLR の強化



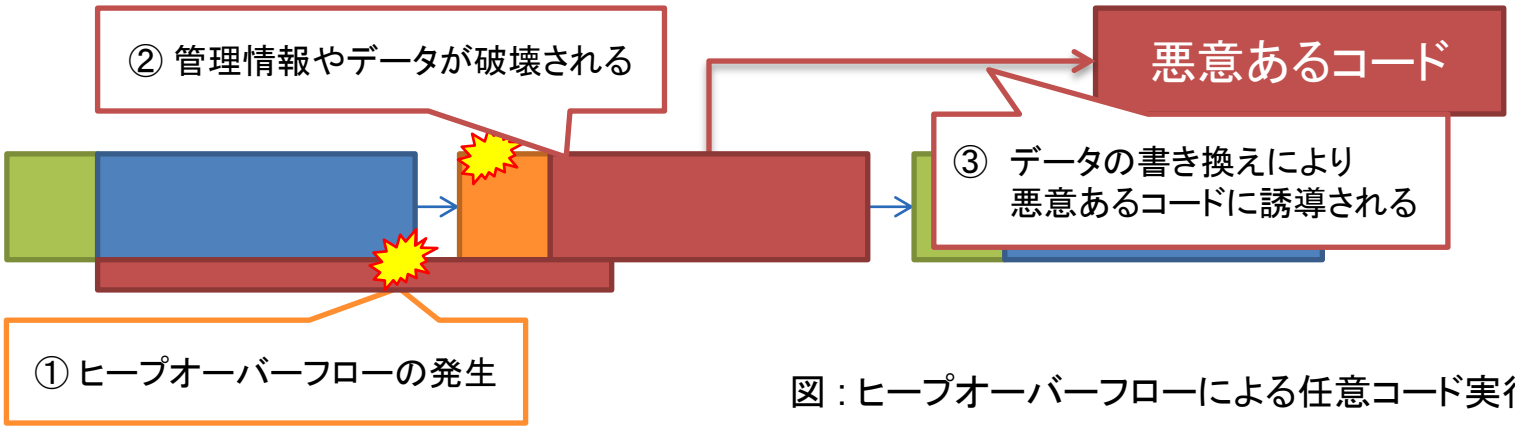
- Windows 7 まで
  - 実行コードやスタックなどの領域はランダム化されていた
  - 実行時に確保されるメモリ領域はランダム化されていない
- Windows 8
  - Bottom-up/Top-down Randomization とヒープへの ASLR により、実行時に確保するメモリ領域がランダム化されるようになった
  - 特に 64-bit 版において、より広い範囲でランダム化が行われる
  - 脆弱性攻撃の信頼性を大幅に下げることが可能

システムの設定：ASLR の強制

- Windows 8 または更新プログラムを適用した Windows 7 (1) では ASLR が確実に適用されるよう強制することができる
  - ASLR はモジュール (EXE/DLL) 単位で有効か否かが決まる
  - 攻撃者は ASLR が無効のモジュールを狙おうとする
  - ASLR を強制することで攻撃者に対する隙を最小限にできる (Microsoft の EMET 2.1 以降も類似の独自機能を持つ)

(1) : http://support.microsoft.com/kb/2639308/en

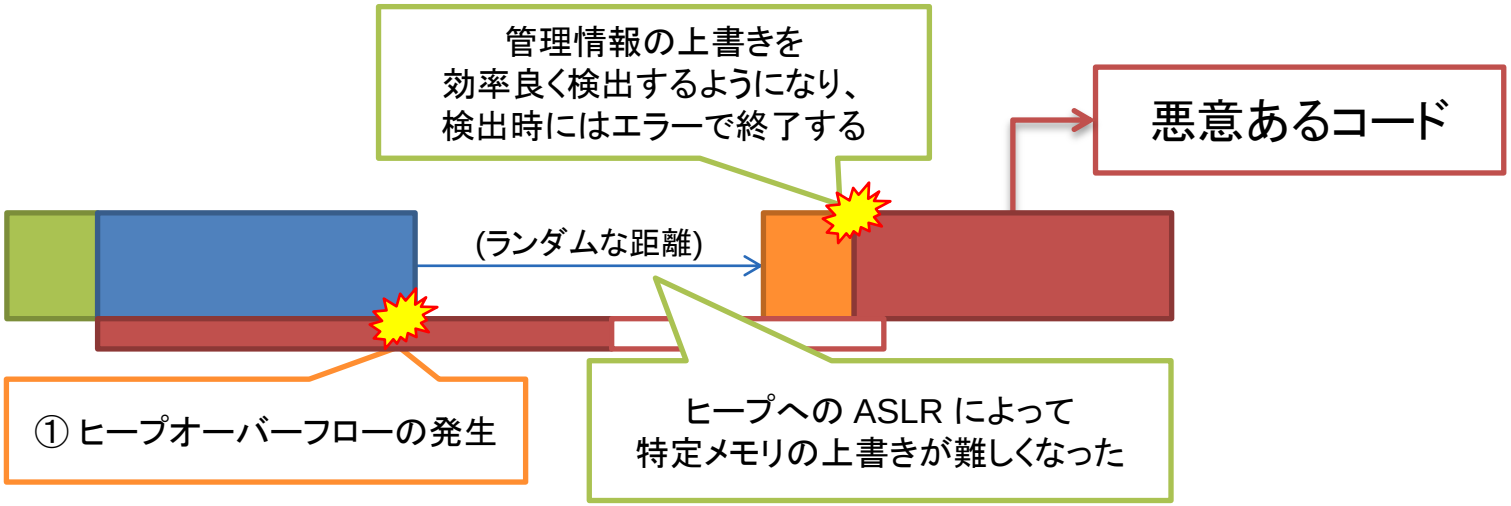
ヒープ保護の強化 (1)



図：ヒープオーバーフローによる任意コード実行例

- ヒープから動的に確保されたメモリは管理情報によって管理される
  - 管理情報を改ざんすることができれば、
    攻撃者はメモリの状態を書き換えることができる
  - 詳細は実装依存だが、任意コードの実行に発展する場合もある
- Windows Vista で導入されたヒープは管理情報の上書きを
  検出するが、既にこれを回避する方法が発見されている

ヒープ保護の強化 (2)

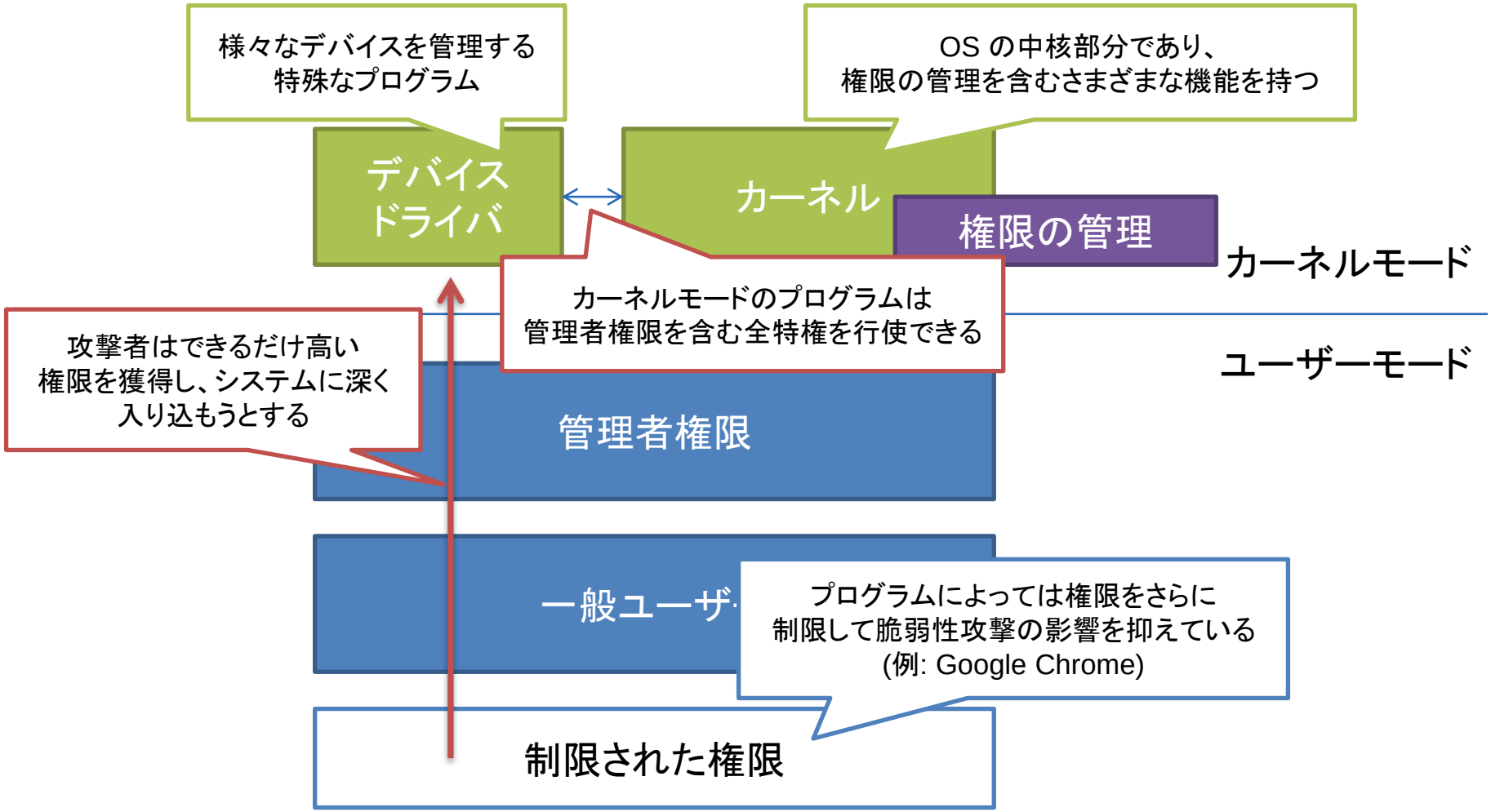


- Windows 8 は主に 3 つの観点からヒープ保護を強化した
  - ASLR によって特定のメモリを狙う攻撃を困難にした
  - 管理情報の上書きを効率良く、より正確に検出するようになった
  - 上書きを検出したときにエラーで終了することを徹底するようになった
- ヒープを悪用する攻撃が不可能になったわけではないが、
  攻撃の難易度はかなり上昇したと推測される

カーネル内部の強化

- 攻撃者がより高い権限で侵入活動を行う場合、
  権限の昇格を行わなければならない
  1. 何らかの手段でコンピュータ内の一般ユーザー権限を奪取する
  2. 高い権限のプログラムにある脆弱性を用いて
     管理者権限やさらに高いカーネルモードの権限を奪取する
- Windows 8 は最も高い権限を持つ中核部分であるカーネルが
  脆弱性攻撃を受けにくいよう、改良が施されている
  - 無効なポインタを悪用する攻撃を防止するために
    NULL ポインタ（通常無効なメモリアドレス）の利用が原則禁止された
  - カーネル内の DEP が導入され、ASLR も強化された
  - 最新の CPU が持っている機能（SMEP/PXN）を用いて
    脆弱性攻撃による異常なコード実行を阻止

カーネル内部の強化：権限の階層



まとめ：Windows の脆弱性緩和機能比較

緩和機能	XP SP0	XP SP2	Vista, 7	Win 8
DEP (ソフトウェア)	×	○	○	○
DEP (ハードウェア)	×	○	○	○
ASLR (スタック)	×	×	○	◎
ASLR (モジュール)	×	×	○	◎
ASLR (ヒープ)	×	×	×	○
ヒープ保護	×	×	△	○
カーネル保護 (ASLR)	×	×	△	○
カーネル保護 (DEP, NULL 保護など)	×	×	×	○

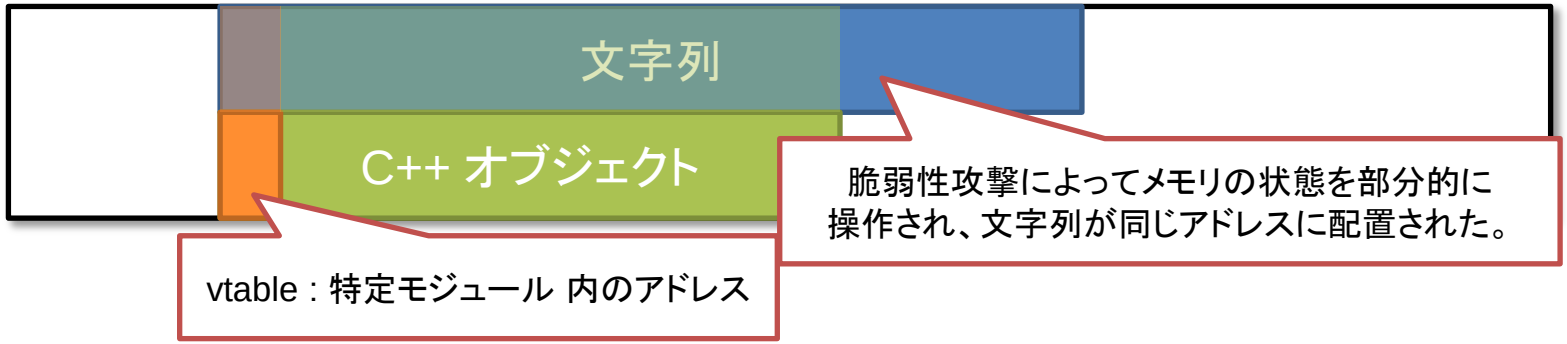
まとめ：Windows 8 における緩和機能の強化

- Windows 8 の緩和機能によって、脆弱性攻撃は困難となった
  - Windows 7 と比べても、複数の点で改良が施されている
  - システムの設定を見直すことによって、
    攻撃者に対する隙を最小限にすることが可能
    (ASLR の強制、16-bit アプリケーションサポートの無効化)

Windows 8 脆弱性緩和機能の回避 (1)

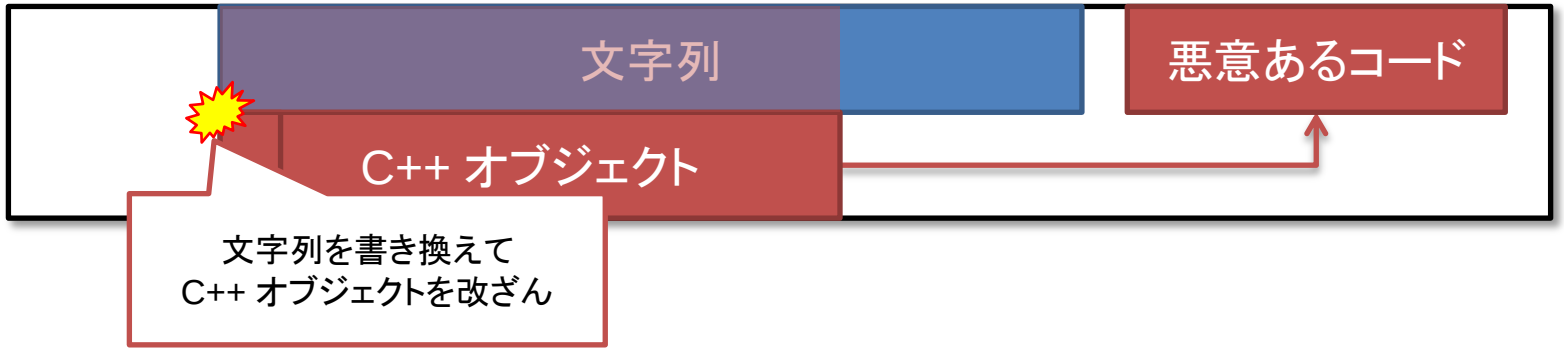
- Windows 8 の脆弱性緩和機能によって、
  脆弱性攻撃が不可能になったわけではない
  1. 脆弱性緩和機能によって十分守られていない古いプログラムを狙う
  2. 脆弱性を持つプログラムではなく、“人” を狙う
  3. 攻撃手法を洗練させることによって脆弱性緩和機能を回避する
- 脆弱性緩和機能を安定して回避するための新しい手法が
  複数提案され、一部は攻撃に用いられている

Windows 8 脆弱性緩和機能の回避 (2)



- Type Confusion (異なる型の取り違え) による例
  - 文字列と C++ のオブジェクトが脆弱性によって同じアドレスに配置された場合、文字列を介して C++ オブジェクトを読み書きできる
  - 読み書きできるデータには、特定モジュール内のアドレスを示すポインタである vtable が含まれる
    - 示されるアドレスは ASLR によってランダム化された後のもの
      → このポインタを悪用することで ASLR を回避することができる

Windows 8 脆弱性緩和機能の回避 (3)

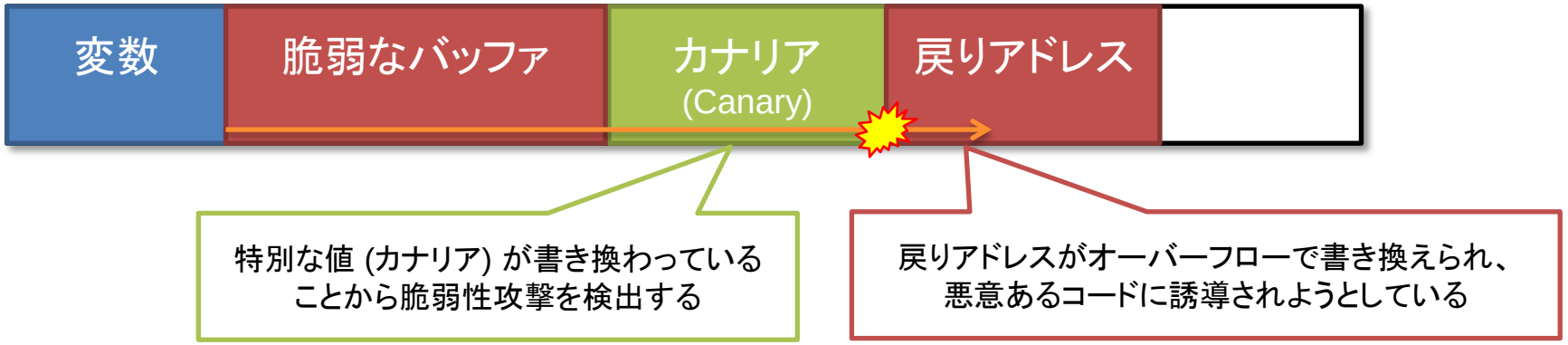


- Type Confusion (異なる型の取り違え) による例
  - 文字列を介して vtable や C++ オブジェクトの中身を書き換える
  - 無効な動作などによって悪意あるコードに誘導される (実装依存)
- 脆弱性攻撃は確かに困難になっているが、不可能になったわけではない
  - 弊社にとっても防御は難しくなっていくと予想はされるが、
    現時点では FFR yarai の ZDP で防御される可能性が高い
  - 最新の回避手法と Flash Player の脆弱性 (CVE-2011-0609) を
    組み合わせた脆弱性攻撃が ZDP によって検出されることを確認

まとめ：ユーザーのために

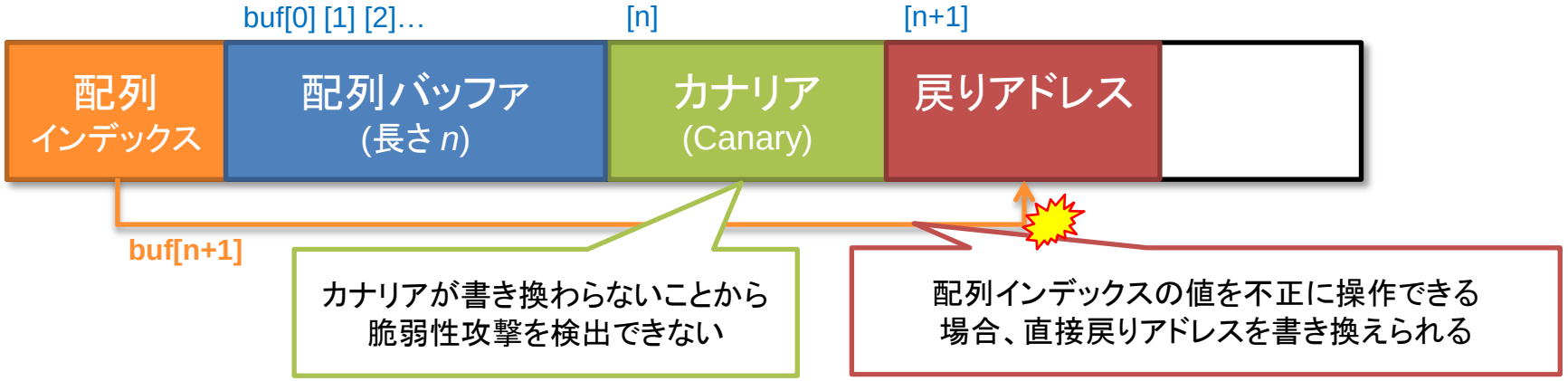
- Windows 8 では脆弱性緩和機能が大幅に強化された
  - アップグレードを行うことで、脆弱性攻撃に対して基礎的な防御力をつけることが可能
  - さらに EMET などを用いてシステムを設定することでセキュリティを強化することができる
- ただし、脆弱性攻撃自体が不可能となったわけではない
  - 脆弱性緩和機能を回避する最新の手法が存在する
  - 依然として、脆弱性攻撃が成功する可能性を考慮に入れることは必要
    - セキュリティ対策が不要となったわけではない（マルウェア対策としても Windows Defender は基礎的）

補助資料：Visual C++ 2010 の Enhanced /GS (1)



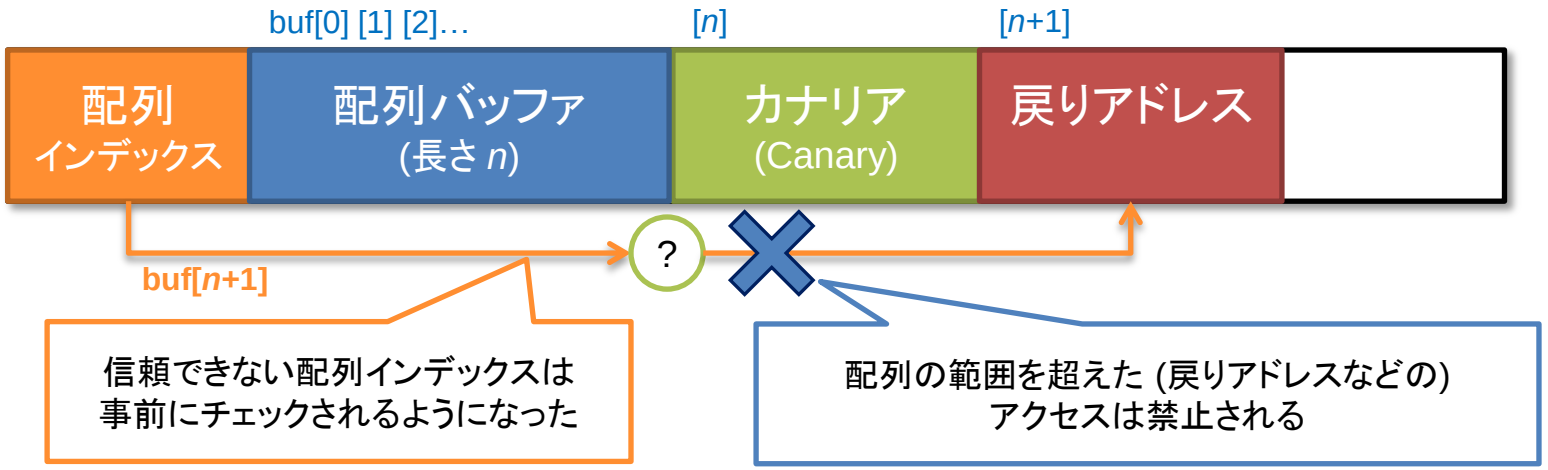
- 従来の /GS (スタックオーバーフロー防止機能)
  - 悪用されやすい配列などのバッファの後にカナリアという攻撃を検出するための値を配置しておく
  - これが書き換わったときに脆弱性攻撃が行われたとしてプログラムを停止させる
  - 古典的なスタックオーバーフローへの対策として有効

補助資料：Visual C++ 2010 の Enhanced /GS (2)



- 従来の /GS（スタックオーバーフロー防止機能）
  - ただし、配列のインデックスなどを不正に操作できる場合、
    カナリアを飛び越えて直接戻りアドレスなどを書き換えることが可能
  - カナリア自身は書き換わらないため、脆弱性攻撃を検出できない

補助資料：Visual C++ 2010 の Enhanced /GS (3)



- Visual C++ 2010 の Enhanced /GS (スタックオーバーフロー防止機能)
  - 信頼できない配列インデックスなどに対するチェックを追加
  - 確保した配列のサイズを超えるアクセスを禁止
  - これによってカナリアを書き換えないスタックオーバーフロー攻撃も防ぐことが可能になる