Future
Technology Days

どう変わる？
Windows7 & Vista アプリケーション開発 ～セキュリティ対応～

NECラーニング
Microsoft MVP for Visual Basic
山崎 明子

アジェンダ

- Windows 7とVistaのセキュリティに関する互換性問題
  - リソースの管理
  - セキュリティの基本、IL
- UAC
  - UACの目的、動作
  - UACの仕組み、昇格
- UAC以外のセキュリティ機能
  - UIPI
  - IE保護モード
  - セッション0の分離
  - WRP
- セキュリティを考慮した設計

アジェンダ

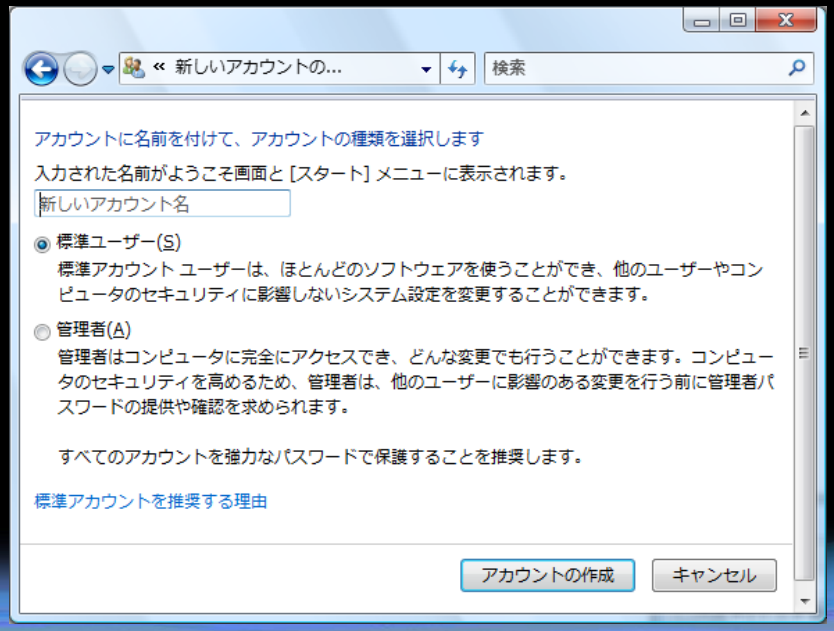
- Windows 7とVistaのセキュリティに関する互換性問題
  - リソースの管理
  - セキュリティの基本、IL
- UAC
  - UACの目的、動作
  - UACの仕組み、昇格
- UAC以外のセキュリティ機能
  - UIPI
  - IE保護モード
  - セッション0の分離
  - WRP
- セキュリティを考慮した設計

Windows 7とVistaの互換性問題
ほぼ同じ

- リソースの管理
- 整合性レベル - Integrity Level (IL)
- ユーザーアカウント制御(UAC)
- ユーザーインタフェース特権の分離(UIPI)
- Internet Explorerの保護モード
- セッション0の分離
- Windowsリソース保護(WRP)
- その他セキュリティ以外の互換性問題

Windows 7とVistaのユーザーの種類

- Administrator
  - ビルトインのAdministrator
- Administrators グループのユーザー
  - 管理者権限を持つユーザー
- 標準ユーザー
  - 管理者権限を持たないユーザー
- 動作を検証するのも、この3ユーザーで！



互換性問題

- どのような問題があるか？
- 問題の洗い出し
  - 実際のユーザー、シナリオで動作検証
    - 管理タスクを行うアプリケーション
      - AdministratorとAdministratorsグループのユーザで検証
    - それ以外のアプリケーション
      - 標準ユーザーで検証
- 問題の切り分け→原因判明
  - 対応策
  - 回避策
  - Windows の問題
    - フィードバックを！

リソースの管理

- フォルダ構造が整理
  - 主な変更点
    - ユーザー毎
    - ユーザー共通
  - 環境変数
  - ディレクトリの接合
  - リソース管理の考え方

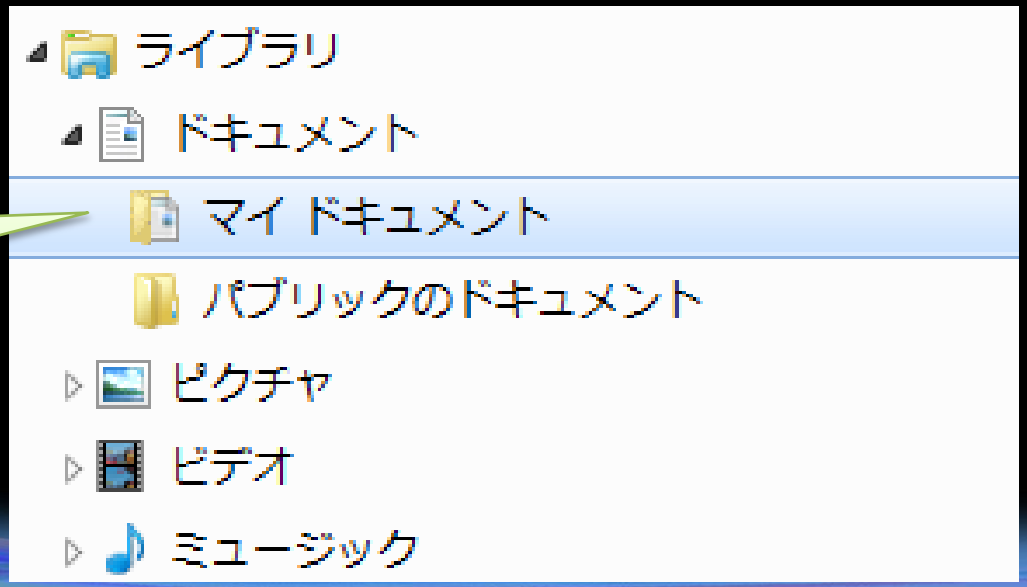
主な変更点　ユーザー共通

項目	XP	Vista	Windows 7
ユーザー プロファイルのルート 名称	C:¥Documents and Settings	ユーザー C:¥Users	
All Users プロファイルの名称	All Users	パブリック Public	
All Users プロファイルのアプリケーション データの場所と名称	All Users ¥Application Data	C:¥ProgramData	

主な変更点　ユーザー別

項目	XP	Vista	Windows 7
ドキュメント/ピクチャ/ミュージック など	マイ ドキュメント My Documents	ドキュメント Documents	マイ ドキュメント Documents　※
ローミングありのアプリケーションデータ	%UserProfile%¥Application Data	%UserProfile%¥AppData¥Roaming	
ローミングなしのアプリケーションデータ	%UserProfile%¥Local Settings¥Application Data	%UserProfile%¥AppData¥Local	

※Windows7のマイドキュメント



環境変数

- 新規追加
  - 名称や場所の変更用

%AppData%	AppData¥Roaming
%LocalAppData%	AppData¥Local
%ProgramData%	ProgramData
%Public%	Public

- パスの変更
  - 既存の変数利用を考慮

%AllUsersProfile%	ProgramData
%Temp% or %TMP%	AppData¥Local¥Temp
%UserProfile%	Users¥ユーザ名

- 変更なし

%CommonProgramFiles%	Program Files¥Common Files
%ProgramFiles%	Program Files
%WinDir%	Windows

ディレクトリの接合

- ユーザープロファイル内の、変更されたフォルダやファイルへの書き込みアクセスを行った場合、新しいフォルダへリダイレクトされる

  例)

アクセス先	%UserProfile%¥Application Data¥sample.ini
リダイレクト先	%UserProfile%¥AppData¥Roaming¥sample.ini

- ただし、接合ディレクトリ (変更されたフォルダやファイル) への読み取りアクセスは拒否される

リソース管理の考え方

- マシン全体のデータの保存先

プログラム ファイル	%ProgramFiles%
システム ファイル	%WinDir%
共有データ	%ProgramData%
レジストリ	HKLM

※ 書込みや編集には管理者権限が必要

- ユーザーごとのデータの保存先

ユーザーのファイルの保存先	マイ ドキュメントなど
ローミングなしのデータ	%LocalAppData%
ローミングありのデータ	%AppData%
レジストリ	HKCU

整合性レベル
- Integrity Level (IL)

- アクセスを制御するメカニズム
  - 対象：
    - セキュリティ保護が可能なオブジェクト
    - プロセス、ファイル、レジストリ・・・
  - Windows 7/Vistaの様々な場面で使用
    - ユーザアクセス制御、UIPI、保護モードのIEなど・・・

IL	権限	フォルダやレジストリの例
高	管理者権限 – システム用の領域への書き込みが可能	%PrograFiles% や %WinDir% HKLM
中	標準権限 – ユーザー用の領域への書き込みが可能	%UserProfile% HKCU
低	信頼できない権限 – 安全な領域への書き込み のみ可能	%UserProfile%¥AppData¥LocalLow HKCU¥Software¥AppDataLow

アジェンダ

- Windows 7とVistaのセキュリティに関する互換性問題
  - リソースの管理
  - セキュリティの基本、IL
- UAC
  - UACの目的、動作
  - UACの仕組み、昇格
- UAC以外のセキュリティ機能
  - UIPI
  - IE保護モード
  - セッション0の分離
  - WRP
- セキュリティを考慮した設計

ユーザーアカウント制御
- User Account Control (UAC)

- UACの目的
  - UACの真の目的
- 実現のアプローチ
  - トークンのフィルター
  - 権限の昇格
  - 盾(シールド)アイコン
- Windows 7におけるUAC
- 権限昇格の方法
- 検証のポイント
- 対応のポイント

UAC登場の背景

- XPの時も・・
  - すべてのユーザーを制限付きユーザーとして動作させることで攻撃の危険性を削減できることはわかっていた
- しかし・・
  - Windows XP の制限付きユーザーでの問題
    - 正しく動作しないアプリケーションがある
    - Windows の多くのタスクの実行に管理者権限が必要
- かといって・・
  - 適切にログオン/ログオフを繰り返すのは面倒
- そのため・・
  - 最初から管理者でログオン
  - 攻撃を受けたときの被害は甚大

UACの目的

- XPのセキュリティ問題をVistaで解決!?
- **アプリケーションやコンピュータの使い勝手をできるだけ損なわずにセキュリティを強化**
- 新たな問題
  - 権限昇格を確認するためのUACダイアログが何度も出る・・・
- Windows 7で改善!
  - 詳細は後ほど・・

UACの真の目的

- 最終ゴール:
  - できる限り、すべてのアプリケーションを標準ユーザーとして実行できるようにする
- なぜ?
  - 安全性:
    - 脆弱なアプリ≠ 脆弱なマシン
  - TCO:
    - 簡単には壊れない
  - 運用性:
    - ポリシーに従った運用が可能

UACの実現方法
トークンのフィルタリングを活用

- トークンとは
  - ユーザーや所属グループ、保持している特権などの情報により、ユーザーを表したもの
- トークンのフィルター
  - ログオン時に標準ユーザー以外のユーザーのトークンを分割する
    - フィルタ済みトークン (標準権限のみ)
      - 管理者グループを含むすべてのユーザーが通常動作で利用する
    - フル トークン (保持している全権限)
      - 権限昇格を行った場合に利用する
- 例外
  - ビルトインの Administrator は常にフルトークンで動作する
  - Windows 7/Vistaでは Administrator は 既定で無効

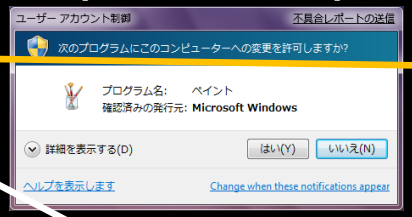
管理者承認モードにおける トークンのフィルターの動作

2) トークンのフィルタ

フル トークン
(管理者権限)

フィルタ済み
トークン
(標準権限)

権限昇格により
フルトークン
(管理者権限) で実行



- ✓ HKLM , %WinDir% や %ProgramFiles% への書き込み
- ✓ 日付の変更
- ✓ アプリケーションのインストール
- ✓ デバイスの削除
- ✓ ドライバの変更

1) ログオン

Windows Vista™

Windows® 7

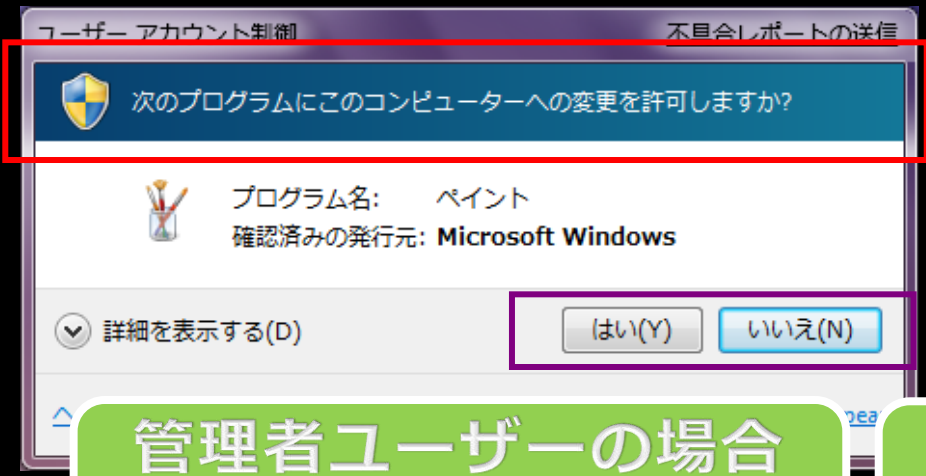
管理者ユーザー

フィルタ済み
トークン (標準権限)
で実行

- ✓ %UserProfile% や HKCU への書き込み
- ✓ タイムゾーンの変更
- ✓ メモ帳の実行
- ✓ システムの設定の参照

UACダイアログによる権限の昇格

- 管理者権限が必要なアプリケーションやタスクを実行する際は、ダイアログを表示し権限の昇格を確認させる
- 発行元の種類によって帯の色が変更
  - 青、グレー、黄色、赤



管理者ユーザーの場合
昇格の許可

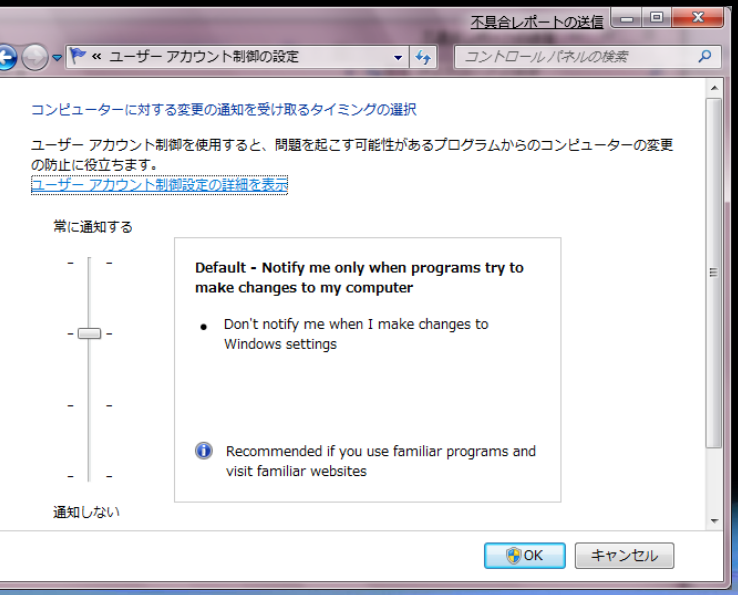
標準ユーザーの場合
管理者ユーザーの資格情報

盾(シールド)アイコン

- 管理タスクを明確に特定する
  - 昇格が必要なタイミングをユーザーが予測できる
  - UAC が無効の場合にも表示される
  - 状態は 1 つのみ
- 昇格のステップ

Windows 7 におけるUAC

- VistaのUACとほぼ同じ
  - On/Off可能
    - Offは推奨されない
  - セキュリティポリシーで詳細に設定可能
- 改善点=Vistaと異なる点
  - 必要最低限のUACダイアログの出現
  - UACの設定がシンプルに、さらに可視化
  - コントロールパネル
    - システムとセキュリティ
      - Action Center
        - UACのレベルの設定が可能



UACのレベルの設定

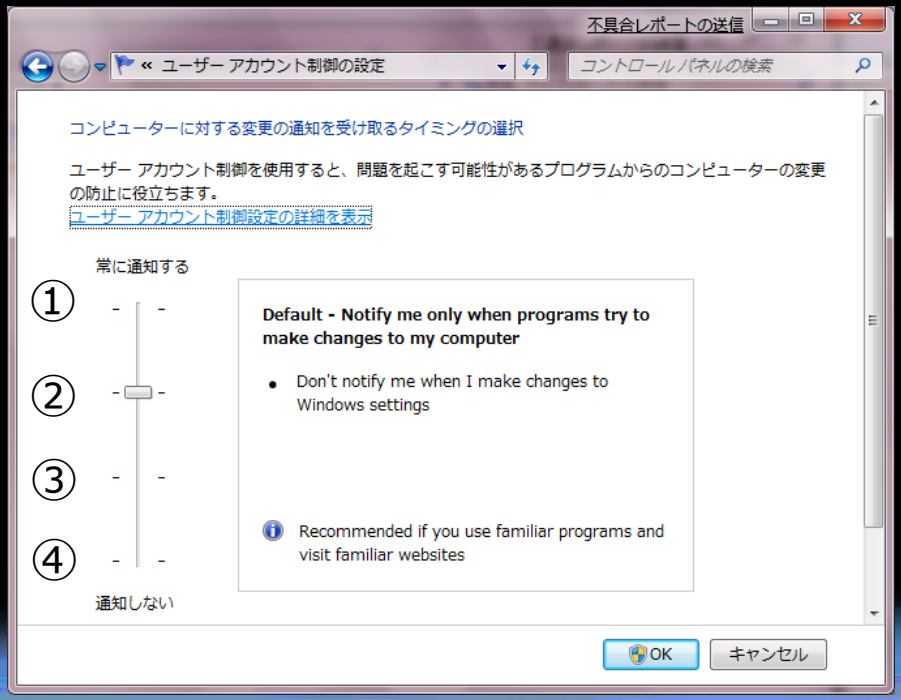
レベルは4段階

① 常に通知
- 常にUACダイアログ表示
- Vistaと同じ（Vistaは、OnとOffのみ）

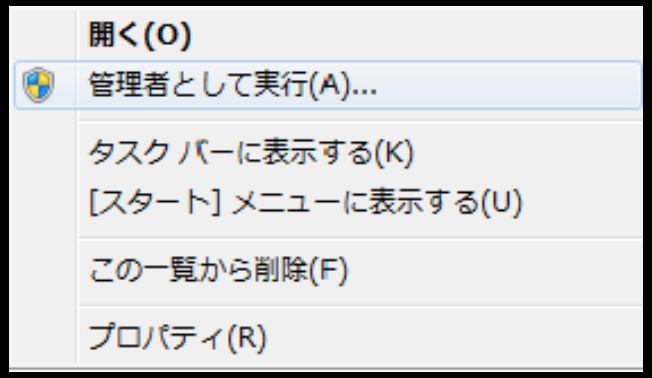
② コンピュータへの変更の時だけ表示
- Windowsの設定変更では表示されない
- デフォルト

③ ②と同じタイミング
- ただしUACダイアログ表示中も他の操作OK

④ 通知しない
- UAC無効
- 推奨されない



権限昇格の方法



- コンテキスト メニューの「管理者として実行」から昇格を行う
- アプリケーション互換性テクノロジを利用
  - [互換性] タブ
  - 互換フィックス
- アプリケーション マニフェストで実行権限を指定
  - level=“requireAdministrator“

権限昇格の方法 – マニフェスト

- EXE の実行権限を定義
  - {ファイル名}.exe.manifest (XML ファイル) 内に記述

```
<requestedExecutionLevel level="asInvoker" uiAccess="false"/>
```

  - Level

asInvoker	親プロセスと同じ権限で動作
highestAvailable	ユーザーが取得可能な権限で動作
requireAdministrator	管理者権限で動作

  - uiAccess

false	通常のアプリケーション
true	UIPI ※の制限を回避

※UIPIは後述

- 添付方法
  - EXE に埋め込む
  - EXE と同じフォルダに配置

インストーラーの検出機能

- 経験則に基づき、特定のパターンに合致するプログラムをインストーラとして検出し、自動的に権限昇格ダイアログを表示
  - 例) ファイル名に *setup* 、 *install*が含まれている、など
- 適用条件
  - 32 ビット プログラムである
  - 標準ユーザーで実行されている
  - Manifest により実行権限を指定していない

※ 既存アプリケーション向けの一時的な互換性機能であるため、依存しないようにする

ファイルとレジストリの仮想化

- 標準ユーザーによる、書き込み権限のない領域 への書き込みアクセスを、ユーザープロファイルの領域へリダイレクト

管理者権限の領域	リダイレクト後の標準権限の領域
%ProgramFiles%	%LocalAppData%¥VirtualStore¥Program Files
%ProgramData%	%LocalAppData%¥VirtualStore¥ProgramData
%WinDir%	%LocalAppData%¥VirtualStore¥Windows
HKLM¥Software	HKCU¥Software¥Classes¥VirtualStore¥Machine¥Software

  - 読み取りアクセス時は、ユーザーごとの領域を先にチェック
- 適用条件
  - 32 ビット プログラムである
  - 標準ユーザーで実行されている
  - Manifest により実行権限を指定していない
- ※ 既存アプリケーション向けの一時的な互換性機能であるため、依存しないようにする

検証のポイント

- 標準ユーザーで検証
  - 効率的に問題を発見
- 問題発生のタイミング
  - 管理者権限が必要な操作を行った際
    - アプリケーションのインストールや更新
- どのような問題？
  - アクセス拒否のエラーが表示され、操作に失敗する
  - 「管理者権限が必要」というエラーが表示され、操作を続行できない
  - 何も起こらず、操作に失敗する
  - 起動のたびに権限昇格ダイアログが表示される
- 問題の切り分け
  - 権限昇格して問題が発生しなければ UAC の可能性が高い

対応のポイント

- 基本
  - マニフェストにより実行権限を明確に定義
    - OS の一時的な回避策に依存しない
  - 標準権限で動作させることを優先
  - 管理者権限が必要な場合、必要性を吟味した上で、適切に権限の昇格を要求
    - 管理者タスクのみのプログラムとなる場合
    - マニフェストによりプログラム自体を管理者権限として起動する

対応のポイント - 標準権限

- Windows のリソースを適切に使用する
  - マシン全体のデータとユーザーごとのデータの保存先を区別して利用する
  - ※「リソースの管理」を参照
- ユーザー毎の処理はインストール後に行う
  - 権限の昇格により、インストールが別のユーザーとして実行されている可能性がある
  - ユーザーごとの設定変更は初回起動時に行う
- セルフ アップデートには注意が必要
  - 標準権限で動作している場合、アップデートできない
- Windows インストーラ 3.1 以降を利用する
  - UAC との高い連動性
  - 標準権限での更新も可能な場合がある

対応のポイント – 権限昇格

- 管理者権限が必要な操作を分離する
  - 別プログラムとして分離する場合
    - マニフェストで管理者権限 を定義する
    - ShellExecute() で親プログラムから起動する
  - COM オブジェクトとして分離する場合
    - CoCreateInstanceAsAdmin()※を使用し、親プログラムから管理者権限で起動する ※SDKのサンプル
- 管理者権限が必要な操作にはシールド アイコンを配置する
  - ユーザーへ一貫性のある UI を提供する
  - BCM_SETSHIELD

アジェンダ

- Windows 7とVistaのセキュリティに関する互換性問題
  - リソースの管理
  - セキュリティの基本、IL
- UAC
  - UACの目的、動作
  - UACの仕組み、昇格
- UAC以外のセキュリティ機能
  - UIPI
  - IE保護モード －Appendix－
  - セッション0の分離
  - WRP
- セキュリティを考慮した設計

ユーザーインターフェース特権の分離 (UIPI)

- 上位権限のプロセスを、デスクトップに共存する下位権限のプロセスの攻撃から保護する
  - ウィンドウ メッセージを介したシャッター アタック
  - 悪意のある DLL の注入など
- プロセスの権限は UI 特権レベルで管理する
  - UI 特権レベルは整合性レベル (IL) に基づいて分類される

検証のポイント

- 現象
  - 上位権限のプロセスのハンドルの検証に失敗する
  - 上位権限のウィンドウへの SendMessage() や PostMessage() に失敗する
    - API は成功してもウィンドウ メッセージは削除
  - 上位権限のプロセスへのスレッド フックやジャーナル フックに失敗する
  - 上位権限のプロセスへの DLL の注入に失敗する
- 影響を受けるアプリケーションの例
  - オン スクリーン キーボードなどのアクセシビリティ ソフトウェア
- 以下に該当する場合、UIPI の可能性が高い
  - 権限が同じプロセス間では問題が発生せず、上位権限との間で問題が発生する
  - かつ、「管理者として実行」、またはビルトインの Administrator で起動した場合は問題が発生しない

対応のポイント

- 権限を昇格し、自身も上位権限になる
  - UAC の対応方法と同じ
  - ダイアログ上で権限昇格の確認が必要
- 実行権限を昇格せずに、上位権限のプロセスとやり取りする
  - マニフェストで以下を指定
    - level=“asInvoker”
    - uiAccess=“true”
  - プログラムをコード署名
  - %WinDir% または %ProgramFiles% からプログラムを実行

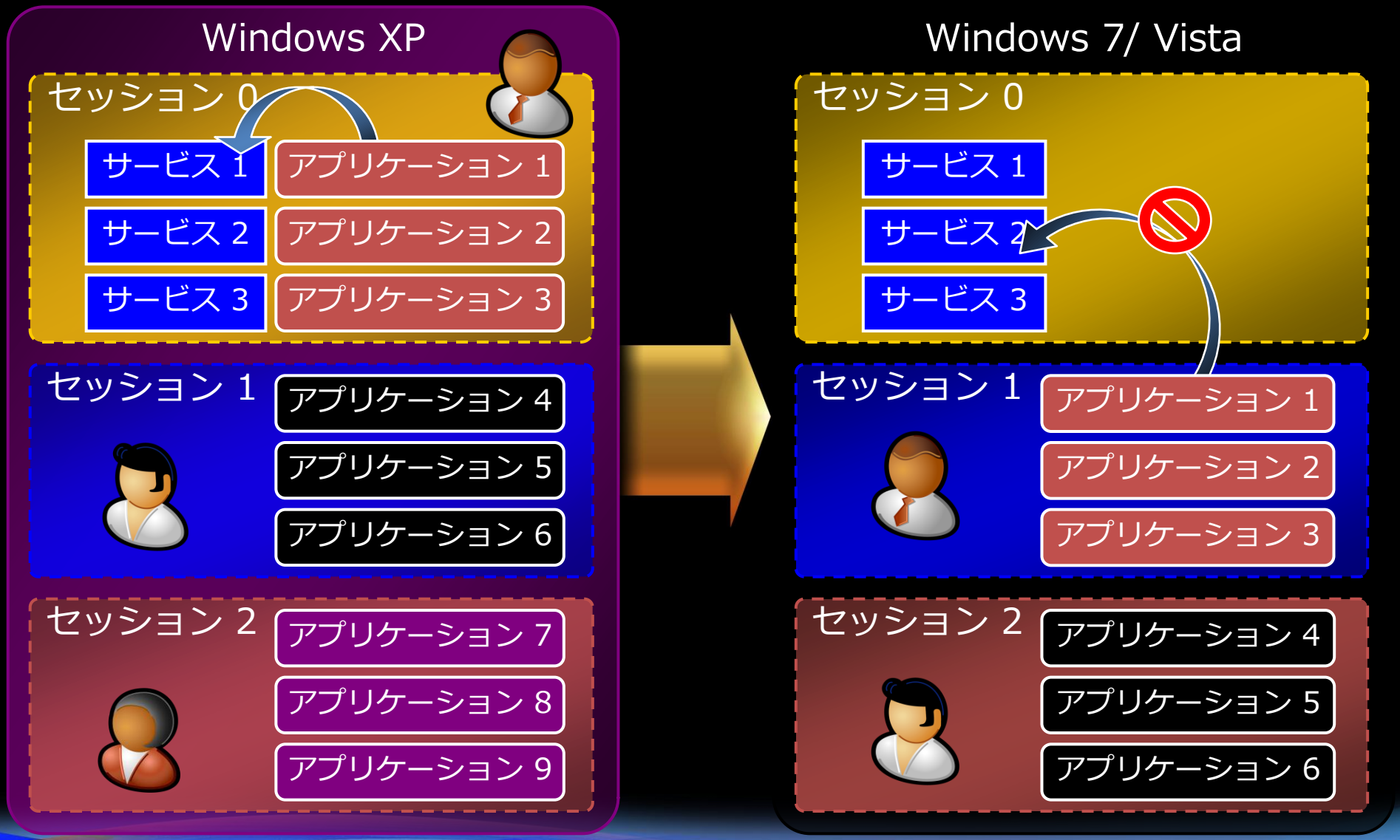
セッション 0 の分離

- セッション 0 の分離とは
- 動作のイメージ
- 検証のポイント
- 対話型サービス検出
- 対応のポイント

セッション 0 の分離とは

- セッション 0 をアプリケーションから分離することで、権限の昇格を狙った悪意のあるプログラムからサービスを保護する
  - セッション 0 では、システム プロセスとサービスのみが動作する
  - ユーザーのセッションは 1 からを利用する

動作のイメージ


Windows XP
セッション 0
サービス 1
サービス 2
サービス 3
アプリケーション 1
アプリケーション 2
アプリケーション 3
セッション 1
アプリケーション 4
アプリケーション 5
アプリケーション 6
セッション 2
アプリケーション 7
アプリケーション 8
アプリケーション 9
Windows 7/ Vista
セッション 0
サービス 1
サービス 2
サービス 3
セッション 1
アプリケーション 1
アプリケーション 2
アプリケーション 3
セッション 2
アプリケーション 4
アプリケーション 5
アプリケーション 6

検証のポイント (1)

- ユーザー インターフェイス (UI) を作成しているサービス
  - 別セッションのユーザーから UI が見えないため、ユーザーからレスポンスを受信できない
- アプリケーションとサービスがウィンドウ メッセージをやり取りしている
  - セッションが異なるため、メッセージが受信ができない
- 画面のプロパティ情報を取得しているサービス
  - セッション 0 にグラフィックス能力がないため、正しい情報を得られない
- ローカル オブジェクトでサービスと同期をとるアプリケーション
  - セッションが異なるため、ローカル オブジェクトで同期をとることはできない
- 新規にフォントをインストールする必要があるサービス
  - ユーザーがインストールする場合、再起動が必要
- 外字(EUDC)を使用するサービス

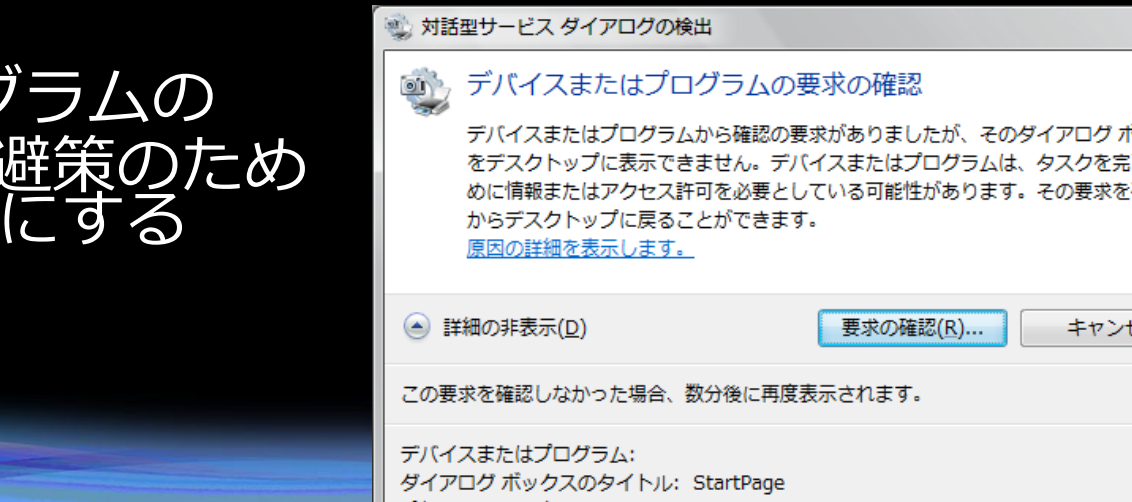
検証のポイント (2)

- 問題の切り分け
  - 対話型サービス検出サービスが作動した場合、この機能変更の影響を受けている可能性が高い
  - Windows XP のユーザーの簡易切り替えを使用し、2 人目以降のログイン ユーザーで問題が発生する場合、影響を受けている可能性が高い

対話型サービス検出

- セッション 0 へのダイアログやウィンドウの作成を検出し、別セッションのユーザーに通知
  - ユーザーはデスクトップをセッション 0 へ切り替え、ダイアログ等を処理することが可能となる
  - 処理が行われない場合、5 分以内にユーザーに再通知

※ サービス プログラムの
一時的な問題回避策のため
依存しないようにする



対応のポイント (1)

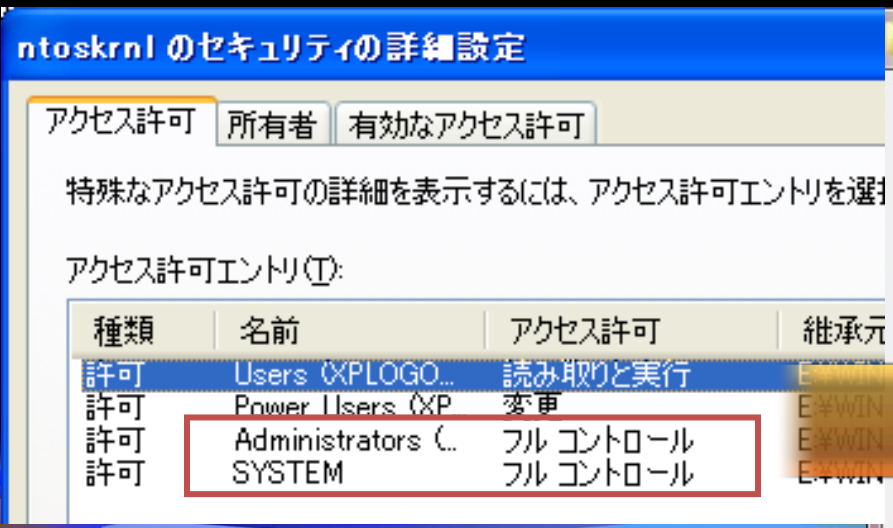
- アプリケーションとの通信方法
  - クライアント サーバー型の通信メカニズムを使用する
    - RPC や名前付きパイプなど
  - ウィンドウ メッセージは使用しない
- 簡単なメッセージ ボックスを利用している場合
  - WTSSendMessage() を使用し、レスポンスを取得する
- 複雑な UI を利用している場合
  - CreateProcessAsUser() でユーザーのセッションにプロセスを作成する
  - 作成したプロセスから UI を表示する
  - クライアント サーバー型の通信メカニズムでレスポンスを取得する
- サービスとオブジェクトを共有する方法
  - オブジェクトをグローバル名前空間に作成する

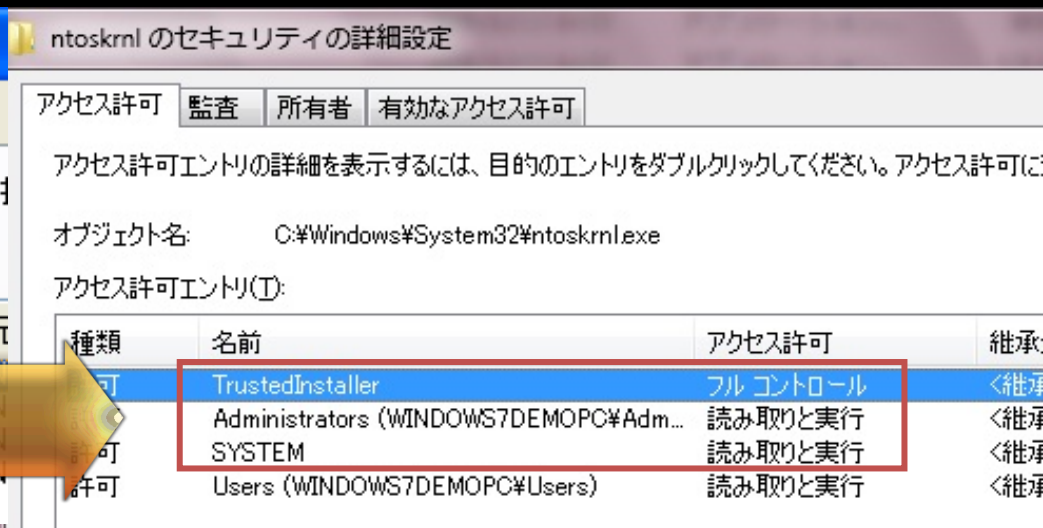
対応のポイント (2)

- 画面のプロパティの取得方法
  - ユーザーのセッションで行う
- フォントをインストールする必要があるサービス
  - 再起動
  - サービスアプリケーションからフォントをインストール
- 外字(EUDC)を使用するサービス
  - 以下の手順が必要
    - 外字フォントをインストール
    - サービスアカウント用のレジストリに外字を登録
    - EnableEUDC()を呼び出し

Windowsリソース保護 (WRP)

- Windows のリソースやコンポーネントを アクセス制御 (読み取り専用) で保護する
  - 保護対象はシステム ファイルやフォルダ、レジストリ キー
  - TrustedInstaller (Windows モジュール インストーラ サービス) にのみフル コントロール権限が許可される
  - ビルトインの Administrator や管理者ユーザー、システム アカウントでも既定では読み取りと実行権限のみ





検証のポイント

- 互換性への影響
  - アプリケーションのインストール時などに、ファイルやレジストリが書込/更新できない
- WRPのセキュリティ機能
  - アクセス拒否エラーの表示をOS 側で自動的に抑制
    - 関数は成功するが、リソースの変更は出来ない
  - 条件
    - インストーラの実行権限がマニフェストで定義されていない
    - 管理者権限で動作している
    - WRP で保護されたリソースの作成や変更、削除によるエラーである

対応のポイント

- やってはいけないこと
  - 再配布パッケージやサービス パック等以外で システム ファイルやコンポーネントのインストールや更新を行う
  - 再配布パッケージの分解や再パッケージ化によるリソースの配布
- できること
  - WRPで保護されているかを判別する

ファイル	SfcIsFileProtected()
レジストリ	SfcIsKeyProtected()

アジェンダ

- Windows 7とVistaのセキュリティに関する互換性問題
  - リソースの管理
  - セキュリティの基本、IL
- UAC
  - UACの目的、動作
  - UACの仕組み、昇格
- UAC以外のセキュリティ機能
  - UIPI
  - IE保護モード
  - セッション0の分離
  - WRP
- セキュリティを考慮した設計

可能な限り管理者権限を持たずに実行

- 低い権限を利用
  - 攻撃からソフトウェアを守ることができる
  - 管理者権限をもつことは危険
- 昇格しないようにマニフェストを利用
  - asInvokerを設定
    - Visual Studio 2008 では、マニフェストをデフォルトで生成
  - ファイル名に"setup" や "install"を含めない
- できる限り、昇格を要求しない

特権の利用法
必要最低限の利用を！

- 必要なときだけ、要求
  - 「とりあえず、全部」はやめる
    - 設定を変えるときにだけ必要
    - 参照するときは不要
- 管理者権限が必要な場面
  - ソフトウェアのインストール
  - デバイスの設定
  - ファイアウォールの設定など・・

特権の要求

- プロセス起動時にのみ昇格可能
  - プロセス実行中の昇格・降格は不可
- 管理者権限取得の3つの方法
  1. プロセス起動時
     - マニフェストで指定
     - 直接呼出し or ShellExecute
  2. アウトオブプロセスのCOMオブジェクト
     - CoCreateInstanceAsAdmin
     - シールドアイコンの設定
  3. タスクやサービスを利用

悪い手順：COM経由で特権を要求する

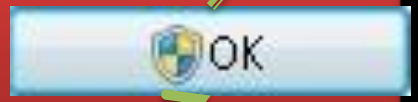
- 一般的なシナリオ
  - 低い権限のアプリケーションが、昇格したCOMオブジェクトを作るためにCoCreateInstanceAsAdminを利用する
  - 低い権限のアプリケーションが、何かをさせるCOMオブジェクトにメッセージを送る
- 問題
  - 低い権限のプロセスが 踏み台になる
    - マルウェアが低い権限のプロセスに、ボタンをクリックしたというメッセージを送る
    - 昇格したCOMオブジェクトに悪いことをさせることができる

悪い手順

予期せぬ相手から
実行させられる

高い特権の
COM オブジェクト

低い特権の
アプリケーション



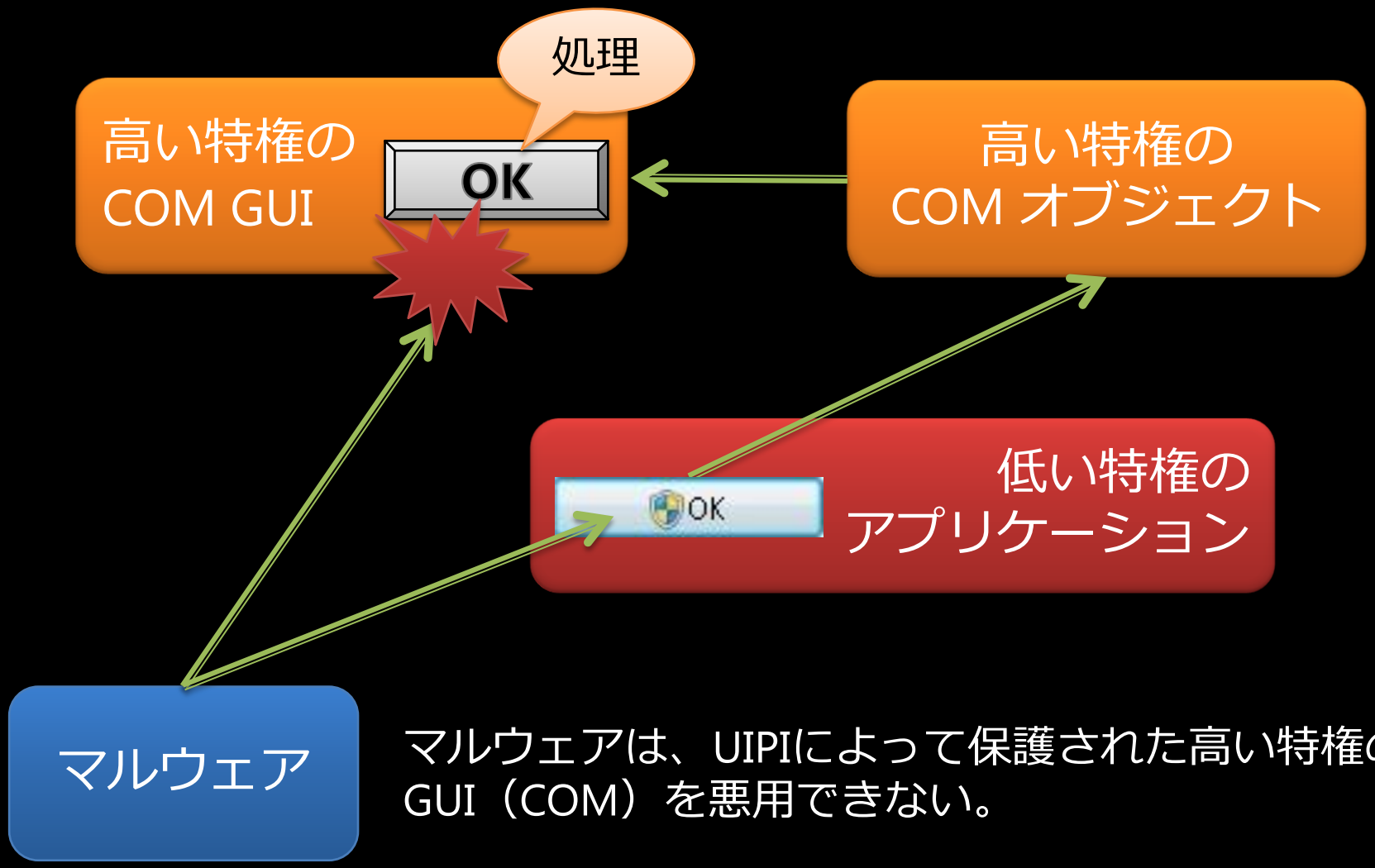
マルウェア

マルウェアが低い特権のアプリケーションを介して、
高い特権のCOMオブジェクトを悪用

よりよい手順：COM経由で特権を要求する

- 権限の低いプロセスから管理者権限が必要な処理を実行したい時
  - 昇格したCOMオブジェクトを作るためにCoCreateInstanceAsAdminを利用する
  - ユーザーがクリックするためのWindowをCOMオブジェクトに描画させる
- それによって・・・
  - Window上のボタンをクリックすることで処理が開始
  - UIPIがWindowを保護
    - これによって権限の低いプロセスは、高いプロセスにクリックメッセージを送れない
- **注意！**
  - 昇格したCOMオブジェクトは危険がいっぱい

よりよい手順



マルウェアは、UIPIによって保護された高い特権のGUI（COM）を悪用できない。

特権が必要な処理を実行するために、タスクやサービスを利用する

- 問題点
  - 攻撃の機会を増やす
  - GUIがない
  - アプリケーションから呼び出せない
- 良い点
  - UACダイアログが出ない
  - デスクトップ上で実行されているGUIのアプリケーション自体は巧妙な攻撃に対し脆弱であるが、サービスを利用するときの制約によって、守られる。

昇格要求は一度だけでOKに

- 問題
  - インストール中に、昇格要求が何度も発生
  - その結果、UACダイアログが何度も表示
- 解決
  - インストーラーの見直し
    - 管理者ブローカーのために一つのプロセスを昇格
    - 昇格が必要なプロセスは、特権ブローカーが起動
  - その結果、UACダイアログは一度だけ表示

※エクスプローラーも同様の問題があった
- Vista SP1で解決

セキュリティを意識した設計

- 管理者権限が、本当に必要か？
- とにかく標準権限での動作を優先する
- 管理者権限が必要なタスクのみ権限の昇格を要求する
  - プロセス、COMオブジェクト、タスク、サービス
  - 権限の昇格が必要なタスクにはシールドアイコンを配置
- どうしても最初から管理者権限が必要なら、マニフェストファイルで設定

- 「Writing Secure Code for Windows Vista」
  - Michael Howard、David LeBlanc 著

まとめ

- Windows 7とVistaのセキュリティに関する互換性問題
  - リソースの管理
  - セキュリティの基本、IL
- UAC
  - UACの目的、動作
  - UACの仕組み、昇格
- UAC以外のセキュリティ機能
  - UIPI
  - IE保護モード
  - セッション0の分離
  - WRP
- セキュリティを考慮した設計

Appendix
IE8 保護モード

Internet Explorerの保護モード

- IEの保護モードの目的
- IE6のモデル
- 保護モードのIEのモデル
- ブローカープロセスの動作
- 互換性レイヤー
- 検証のポイント
- 対応のポイント

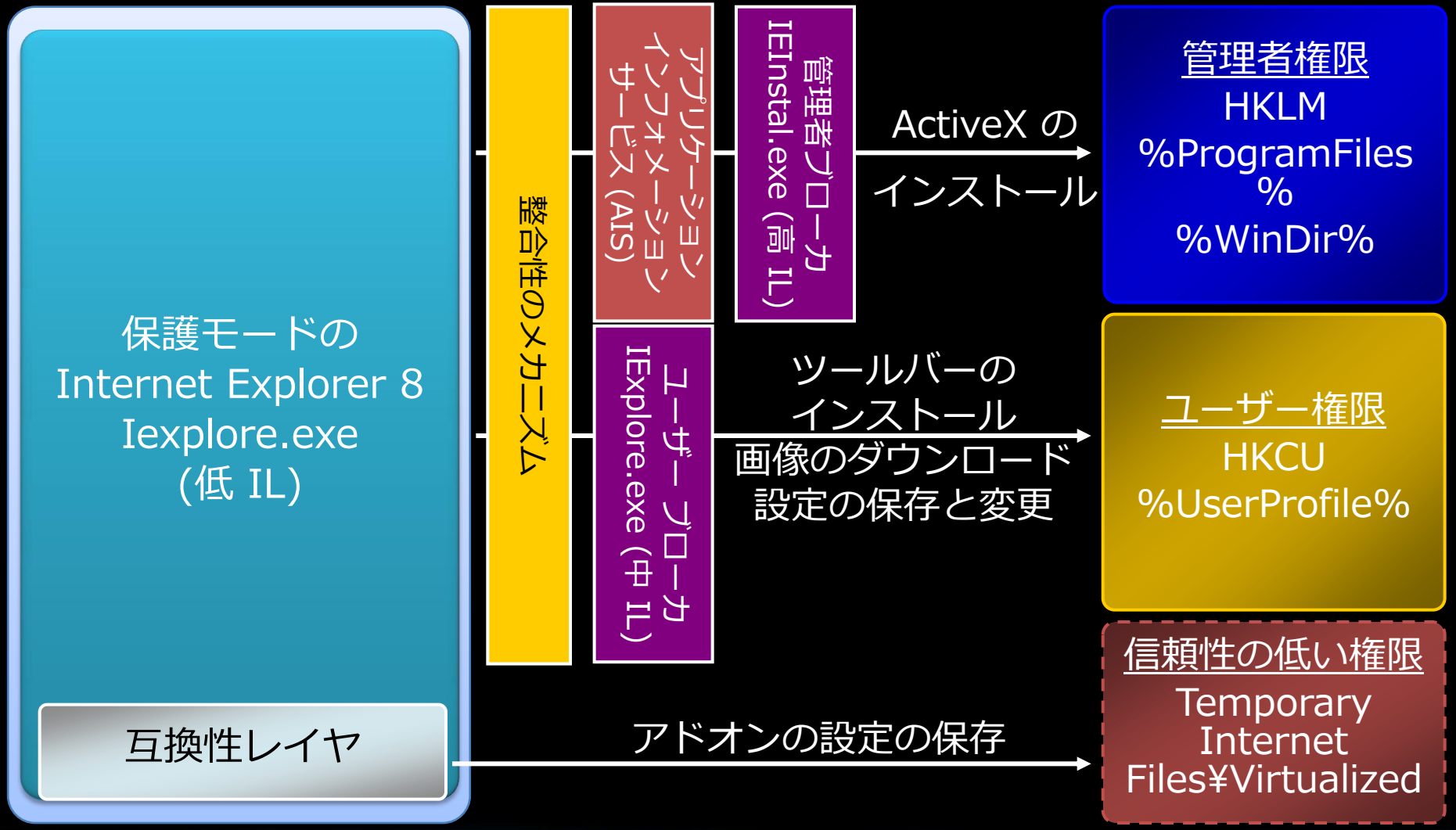
IEの保護モードの目的

- UACなどと連動し、IE を介した悪意のある攻撃からシステムを保護する
- 保護モードの IE は、低ILで動作する
  - 上位プロセスにウィンドウ メッセージを送信することはできない (UIPI の制限)
  - 安全な領域にのみ書き込み可能 となる (IL の制限)
- 上位権限が必要な操作は、ブローカプロセスを介して行う

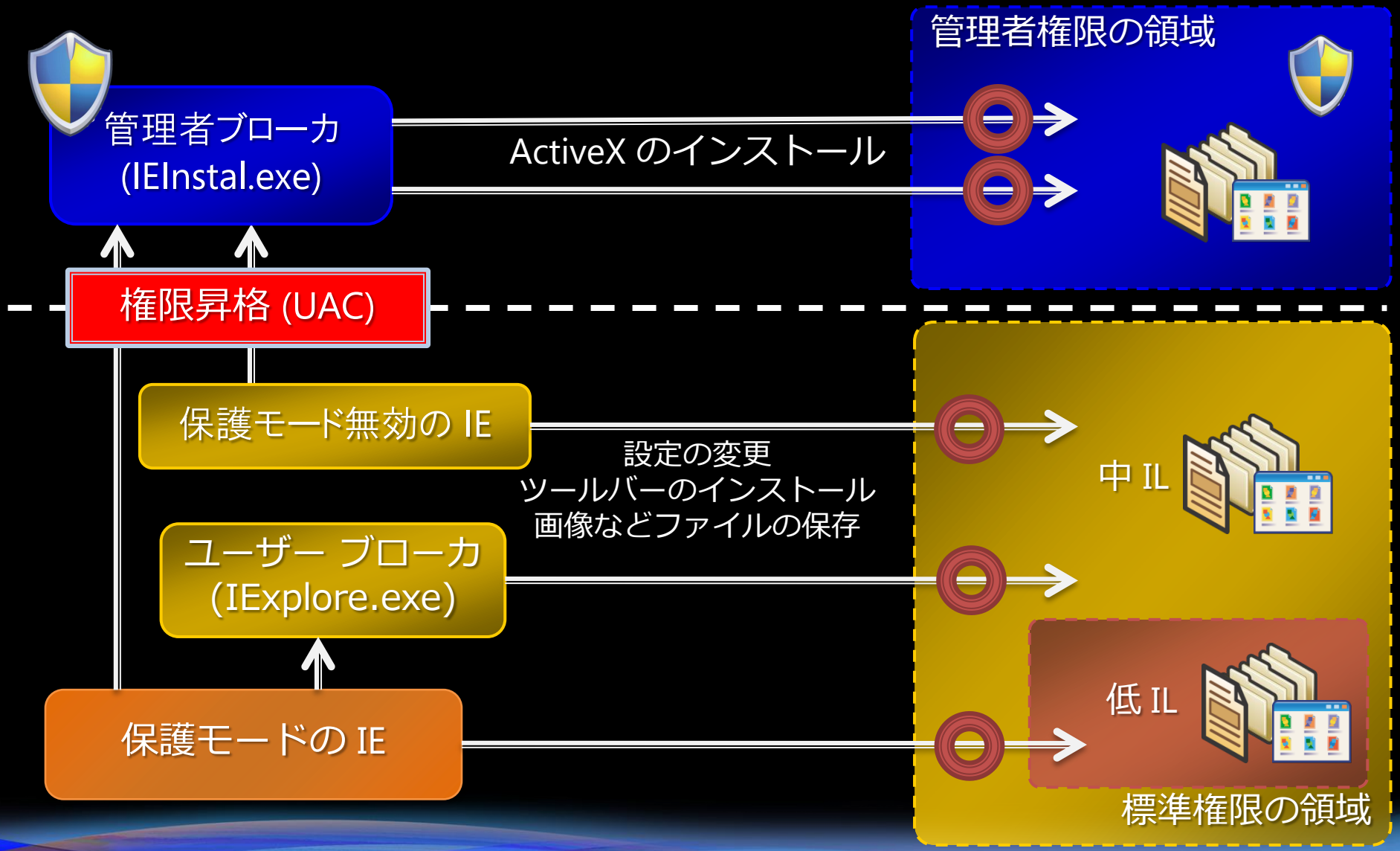
IE6のモデル


Internet Explorer 6
管理者権限で動作
ActiveX などの
インストール
悪意のある
プログラムの
インストール
ダウンロード設定の
保存と変更
悪意のある
プログラムの
インストール
Web コンテンツの
キャッシュ
管理者権限
HKLM
%ProgramFiles%
%WinDir%
ユーザー権限
HKCU
%UserProfile%
Temporary Internet
Files

保護モードのIE8のモデル



ブローカープロセスの動作



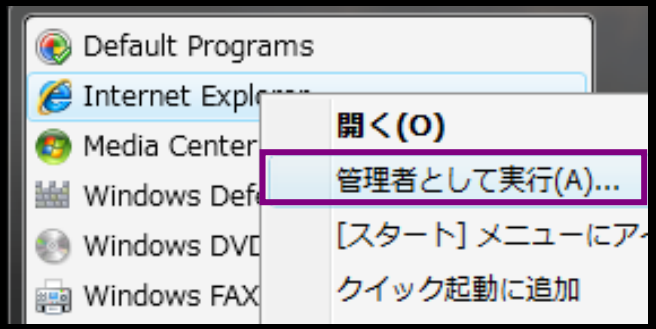
互換性レイヤー

- アドオンなどによる標準権限の領域への書き込みを、安全な領域へリダイレクト
- リダイレクト先

ファイル	%LocalAppData%¥Microsoft¥Windows¥Temporary Internet Files¥Virtualized
レジストリ	HKCU¥Software¥Microsoft¥Internet Explorer¥InternetRegistry

検証のポイント

- 互換性への影響
  - アドオンなどで上位権限のリソースに書き込みできない
  - IE にロードされたモジュールの設定情報等を、IE 以外のプロセスと共有できない
    - 互換性レイヤにより下層フォルダへリダイレクトされるため
- 問題の回避方法
  - 保護モードを無効化する
  - 管理者として実行する
- 問題の切り分け
  - 保護モードを無効にして問題が発生しなくなる場合、保護モードの可能性が高い
  - 保護モードを無効にしても問題が発生する場合、UAC などの可能性がある



対応のポイント

- 保護モードのIE7/8用のAPIを活用する
  - 保護モードで動作しているかどうかを判別する
    - IEIsProtectedModeProcess()
  - HKCU 下の書き込み可能なレジストリを取得する
    - IEGetWriteableHKCU()
  - ユーザー ブローカを介して、低 IL から中 IL へファイル保存を行う
    - 低 IL へ一時ファイルを作成する
    - IEShowSaveFileDialog() を呼び出し、保存先を指定する
    - 引数に一時ファイルのパスを指定し、IESaveFile() を呼び出す
- ブローカ プロセスを作成し上位権限の操作を行う
- 低 IL のフォルダやレジストリで情報を共有する
  - 信頼性の低い場所であることに注意する
    - %UserProfile%¥AppData¥LocalLow
    - %Temp%¥Low
    - HKCU¥Software¥AppDataLow

Windows® 7

Future Technology Days


Microsoft®