Windows 上における危険な処理の承認機構の提案

100430100　早川 顕太
渡邊研究室

1. はじめに

マルウェアは多様化が進み，不正インストールやスパムメールの送信，情報漏えいといった様々な活動を行う．これらの活動はバックグラウンドで行われるためユーザがその危険な処理に気づくことができないという課題がある．

本稿では，Windows 上において危険な処理のユーザへの承認機構を提供することにより，マルウェアがバックグラウンドで行う危険な処理を防止する．

2. 承認機構の既存技術

承認機構の既存技術としては，Windows Vista 以降に導入されたユーザアクセス制御（UAC；User Access Control）が挙げられる．しかしながら，UAC はプログラムの起動時に行われる承認機構であって，プログラムの実行中に行われる動的な承認機構ではない．従って，昇格プロンプトによりユーザに承認を求めた時点では，実際に行われる処理の内容やそのタイミングをユーザが把握することができない．また，標準ユーザの権限で行えるカレントユーザのみへのインストールや，メールの送信などを防ぐとこができないという課題がある．

3. 提案方式

3.1 概要

提案方式は，Windows 上における危険な処理のユーザへの動的な承認機構である．図 1 に提案システムの概要を示す．アプリケーションが危険な処理を行うために発行する Windows API を提案システムがフックすることにより，その危険な処理が行われる直前に，ユーザへの承認ダイアログを表示する．ユーザは行われようとしている危険な処理が自身の意図したものであるかどうかによって，その処理の許可／拒否を選択する．ユーザの応答により，提案システムはその処理を続行するか，あるいは処理を中断させてアプリケーションにエラーを返す．これにより，マルウェアがバックグラウンドで行う危険な処理を，ユーザは自身の意図していないものとして拒否することが可能になる．

3.2 危険な処理の定義

マルウェアにより悪用される可能性がある処理で，かつ，正規ソフトウェアがバックグラウンドで行わないような処理を危険な処理として定義した．正規ソフトウェアがバックグラウンドで行う処理に対しても，ユーザに承認を求めてしまうと，ユーザはその処理の許可／拒否を正しく判断できない恐れがある．このことを踏まえ，現在，危険な処理として表 1 を定義した．

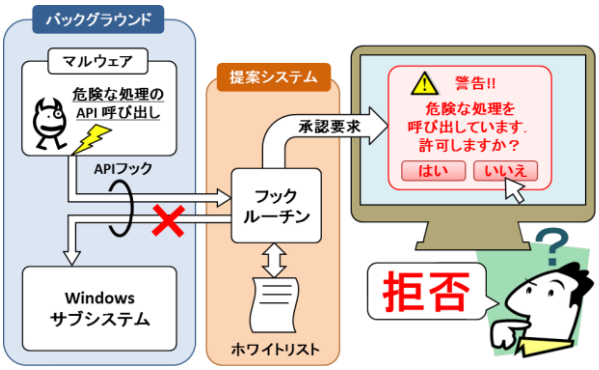


図 1. 提案システムの概要

表 1. 危険な処理の定義

危険な処理	想定される被害
実行ファイルの作成	不正インストール
自動実行への登録	不正インストール
メールの送信	スパムメール
他プロセスの強制終了	セキュリティの無効化
HTTP の POST	掲示板への不正投稿
キー入力の取得	キーロギング

3.3 ホワイトリストの導入

提案方式はホワイトリストを導入し，ユーザビリティの向上を図る．プログラムを一定の期間使用し，ユーザがそれを安全であると判断した場合，そのプログラムの絶対パスと常に許可したい処理（複数可）をホワイトリストへ登録する．それ以降，承認ダイアログは省略され，ユーザビリティが向上される．

4. プロトタイプシステムの実装と実験

提案方式の有用性を評価するため，プロトタイプシステムを実装した． API フックライブラリである Detours ライブラリを用いてフック用の DLL を作成し，これを DLL インジェクションにより監視対象のプロセスへと強制的に注入することで API フックを行う．なお，現在，実装済みの危険な処理は表 1 の内，実行ファイルの作成と，自動実行への登録のみである．

次に，実験により，独自に入手した 53 体のマルウェアをプロトタイプシステム上で動作させた．その結果，31 体のマルウェアについてバックグラウンドで行われる危険な処理を検出でき，提案方式の有用性を示すことができた．

5. まとめ

本稿は，Windows 上においてバックグラウンドで行われる危険な処理を防止するため，危険な処理のユーザへの承認機構を提案した．今後は，未実装である他の危険な処理についても実装・検証を行う予定である．

Windows 上における危険な処理の承認機構の提案

名城大学　理工学部　情報工学科
渡邊研究室
100430100　早川 顕太

Watanabe Lab.

研究背景(1/2)

- マルウェアは多様化が進み，様々な活動を行う
  - 不正インストール，情報漏えい，スパムメール，etc
  - これらの活動はバックグラウンドで行われる

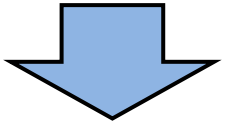
ユーザは気づくことができない

- マルウェア検出技術

	検出方法	未知 マルウェア	暗号化 マルウェア
パターンマッチング法	静的なシグネチャ検出	×	×
静的ヒューリスティック法	静的な振る舞い検出	○	×
ビヘイビア法	動的な振る舞い検出	○	○

研究背景(2/2)

- ビヘイビア法の課題
  - (課題1)振る舞い定義の難しさ
    - 多様化したマルウェアの活動を一概に定義できない
  - (課題2)正規ソフトウェアとの分離
    - マルウェアの活動の多くが正規ソフトウェアにも観測されるため，誤検知が発生しやすい



本提案

危険な処理のユーザへの承認機構を提案

- 振る舞い定義の難しさ　⇒　**複数の処理を承認対象とする**
- 正規ソフトウェアの分離　⇒　**ユーザの判断を借りる**

※対象OSはWindows

承認機構に関連した既存技術

- **ユーザアカウント制御（UAC；User Account Control）**
  - Windows Vista以降，標準搭載
  - 管理者としてログインしても，標準ユーザの権限で動作
  - 昇格プロンプトにより，ユーザの承認を得れば，管理者権限で起動
    ⇒ UACはプログラム起動時の承認機構

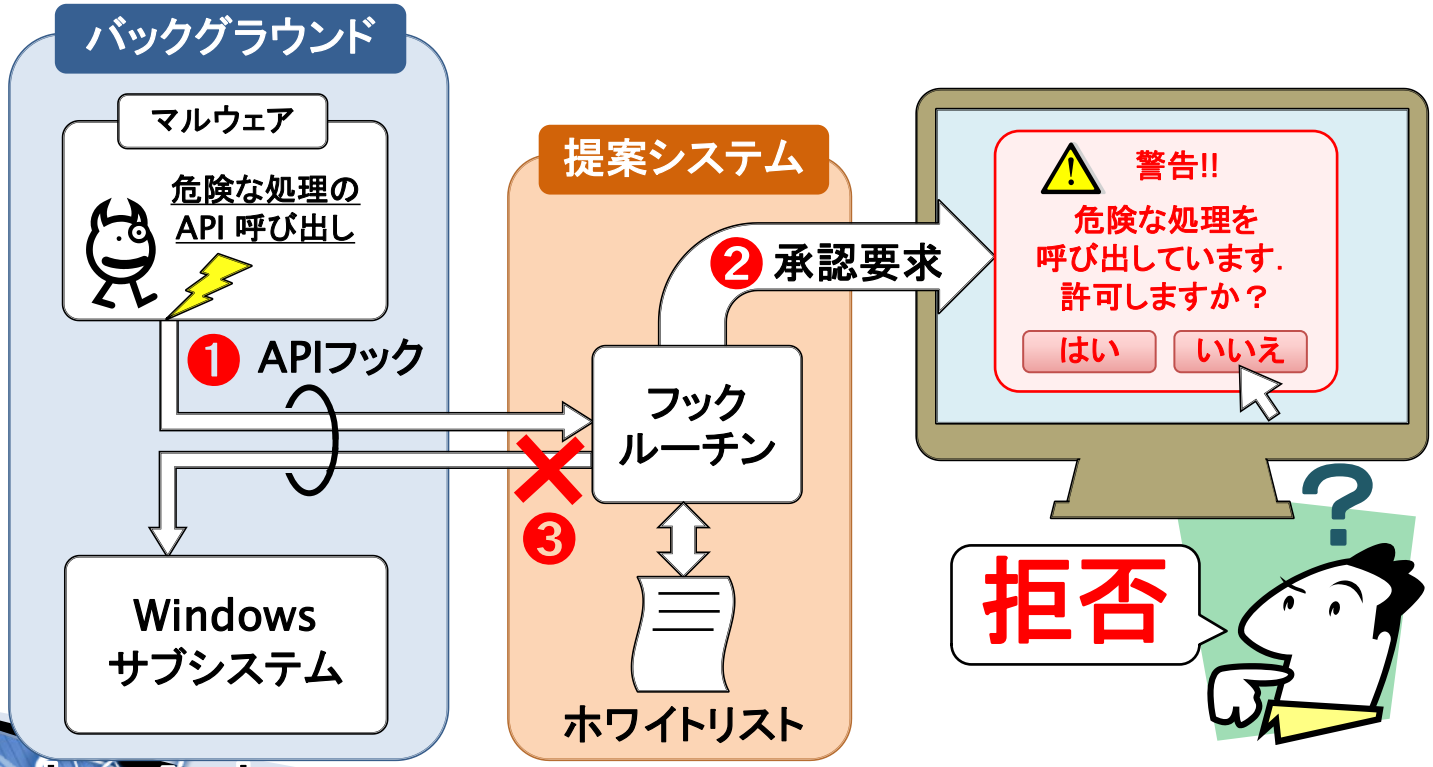
課題

◆**プログラム実行中の承認機構ではない**
- 実際にどんな処理がいつ行われるのか分からない

◆**標準ユーザの権限で行える悪意のある活動を防げない**
- カレントユーザのみへの不正インストール
- スパムメールの送信など

提案方式

- **プログラムの実行中に行われる危険な処理の承認機構**
  - ❶ プログラムが危険な処理を行うために発行するWindows APIをフック
  - ❷ 承認ダイアログによる，ユーザへの承認要求
  - ❸ ユーザの応答により，処理を続行／中断



危険な処理の定義

- マルウェアにより悪用の恐れがある処理
- かつ，正規ソフトウェアがバックグラウンドで行わないような処理
  ⇒　ユーザによる誤検知をなくす

危険な処理	想定される被害
実行ファイルの作成	不正インストール
OS起動時の自動実行へ登録	不正インストール
他プロセスの強制終了	セキュリティ無効化
キー入力の取得	キーロギング
HTTPのPOST	掲示板への不正投稿
メール送信	スパムメール，脅迫メール

承認ダイアログのイメージ

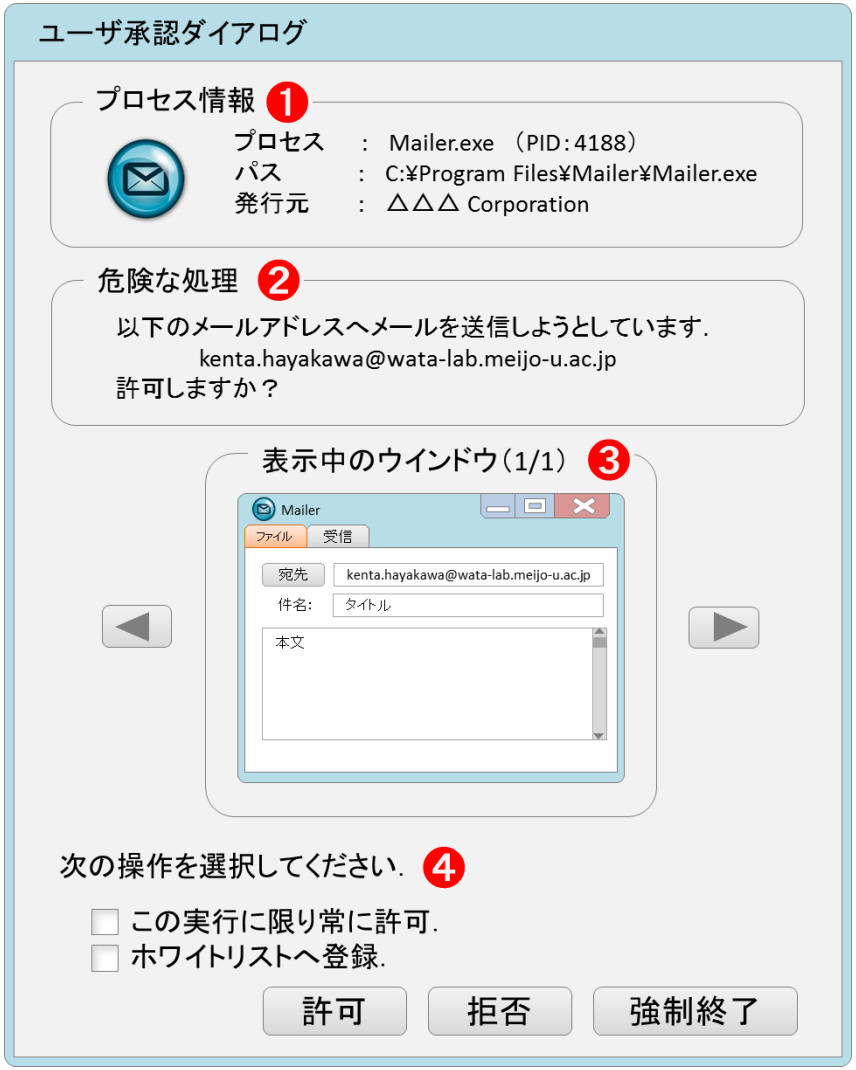
表示する情報

❶プロセスに関する情報
- プロセス名
- プロセスID
- イメージアイコン
- イメージの絶対パス
- 電子署名の発行元

❷ 行われる処理内容

❸ プロセスが表示中のウインドウ

❹ ユーザの選択肢



ホワイトリストを導入

- 登録されたプログラムは，承認ダイアログの表示を省略

ユーザビリティの向上を図る

- ホワイトリストの登録内容
  - プログラムの絶対パスと許可される処理（複数可）

プログラム	許可される処理
Explorer	（実行ファイルの作成）
タスクマネージャ	（他プロセスの強制終了）
Internet Explorer	（実行ファイルの作成）＋（HTTPのPOST）

プロトタイプシステムの実装

- **Detoursライブラリを採用**
  - Microsoft Researchが提供するユーザモードの APIフックライブラリ
  - このライブラリを使用したDLLを作成し，これを監視対象のプロセスへ強制注入することでAPIフックを実現
  - 危険な処理として，現在実装済みである次のAPIをフック

危険な処理	フック対象となるAPI	APIの引数の条件
実行ファイルの作成	NtCreateFile	ファイルを新たに生成し，ファイルの拡張子が”.exe”である場合
	NtSetInformationFile	拡張子が”.exe”以外のファイル”を.exe”へリネームした場合
OS起動時の自動実行への登録	NtSetValueKey	自動実行に関するレジストリの内，マルウェアがよく利用するレジストリエントリへ書き込む場合

実験

- **実験目的**
  - プロトタイプシステムにより，マルウェアが
    バックグラウンドで行う危険な処理を検出できるか調査
- **実験環境**
  - 仮想マシンVMWare上の32ビット版Windows XP SP3
  - プロトタイプシステムを導入
- **使用したマルウェア**
  - 以下のマルウェア収集サイトから独自に入手した
    実行可能なマルウェア　53体
    - Offensie Computing（http://www.offensivecomputing.net/）
    - VX Vault（http://vxvault.siri-urz.net/）

実験結果

- 53体中31体のマルウェアで，承認ダイアログが表示された
  ※この内，2体はExplorerにスレッドを注入して行わせる

これらのマルウェアは提案方式が有効

マルウェア検体数（計53体）	結果（実行ファイルの作成と自動実行への登録のみ）
31	承認ダイアログが表示される
12	承認ダイアログは表示されず，実行を続ける
2	マルウェアによりシステムが操作不能に陥る
8	エラーにより実行不可

- 今後，他の危険な処理も実装することで，さらなる検出率の向上が期待できる

まとめ

- Windows上における危険な処理のユーザへの動的な承認機構を提案
- プロトタイプシステムと危険な処理の一部を実装し，半数以上のマルウェアを検出することに成功
- 今後の予定
  - 他の危険な処理に対し，フックするAPIの検討・実装を行い，さらなる検出率の向上を目指す

ご静聴ありがとうございました

付録

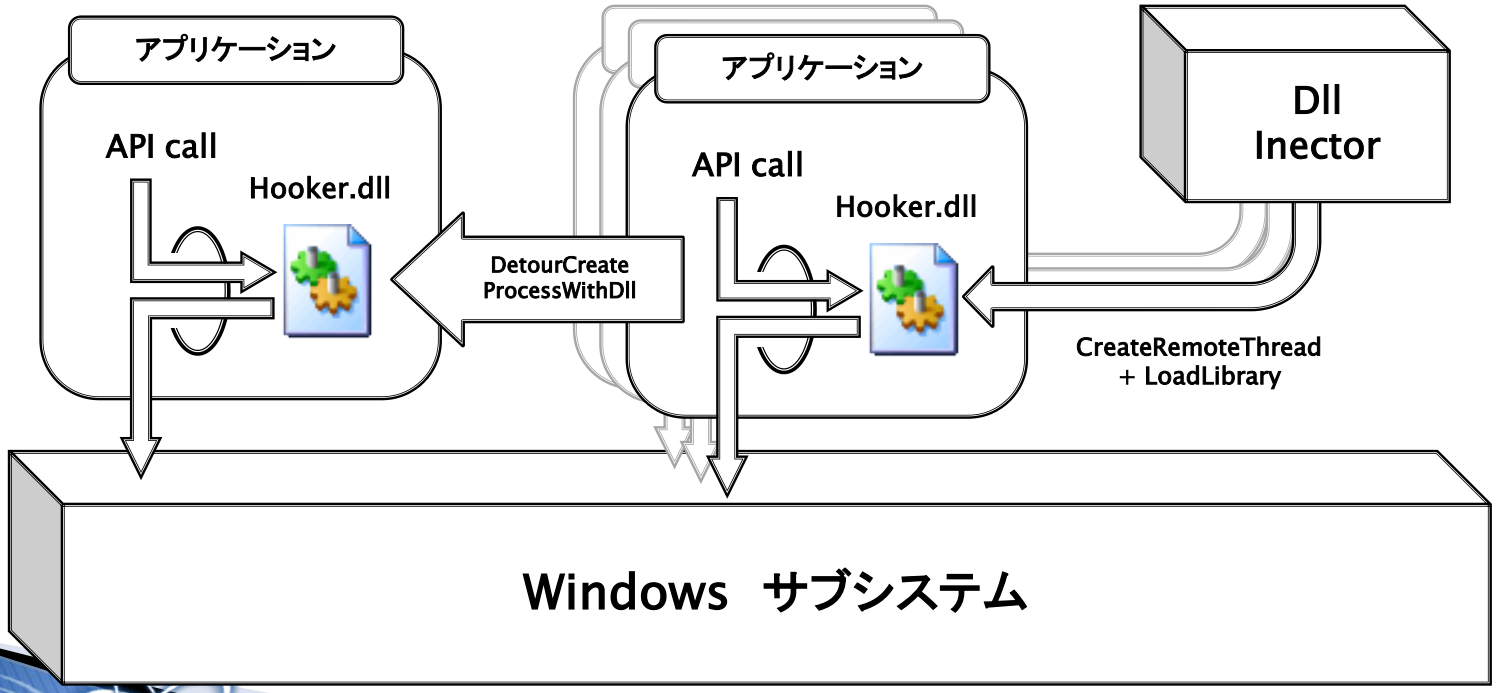
2012年の遠隔操作事件

- **脅迫メールの送信**
  - 被害者が無料ソフトをダウンロードした際，それに同梱されたマルウェアも同時にインストール
  - マルウェアによる遠隔操作により，脅迫メールの送信
- **掲示板へ脅迫文の投稿**
  - 脅迫メールと同様にマルウェアによる遠隔操作
  - クロスサイトリクエストフォーフェリ（CSRF）
    - 攻撃者が用意した悪意のあるWebページにアクセスすると，自動的に，任意のWebページへ任意のHTTPリクエストをさせることができる
    - 被害者はCSRFが仕掛けられたWebページのリンクを踏み，掲示板へ脅迫文が投稿された

＜参考サイト＞http://www.jnsa.org/secshindan/secshindan_1.html

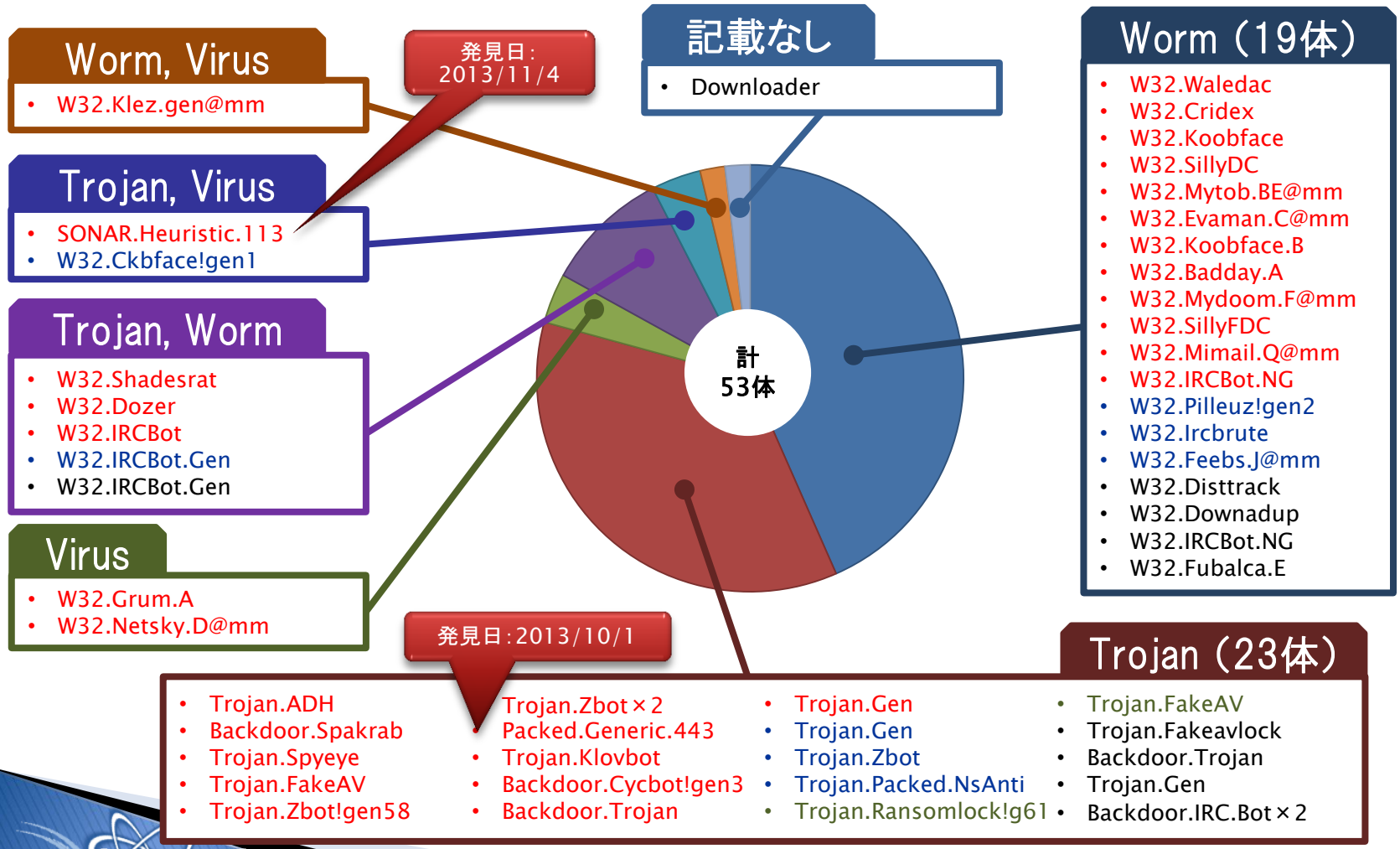
プロトタイプシステムの実装（詳細）

- Detoursライブラリを使用したフック用DLL（Hooker.dll）を作成
- これを監視対象のプロセスへ強制的に注入（DLLインジェクション）
  - （方法1）DLL注入済みプロセスが子プロセスの生成と同時にDLL注入
  - （方法2）駐在プロセス”DllInjector”のプロセス監視によるDLL注入



実験に使用したマルウェアの詳細

- Symantec社によるマルウェア名と種別



正規ソフトウェアによる実験（その1）

- **実験目的**
  - 正規ソフトウェア上で危険な処理がバックグラウンドで行われることがないかを調査

- **実験方法**
  - ProcessMonitorによる危険な処理の監視ツール
    - ファイルシステム，レジストリ，プロセスなどの活動をリアルタイムで表示する監視ツール
    - 一部はカーネルで動作するため，全てのプロセスの活動を漏れなく監視できる
    - 実行ファイルの作成と，自動実行への登録を検出できるようにフィルタを掛ける

正規ソフトウェアによる実験（その2）

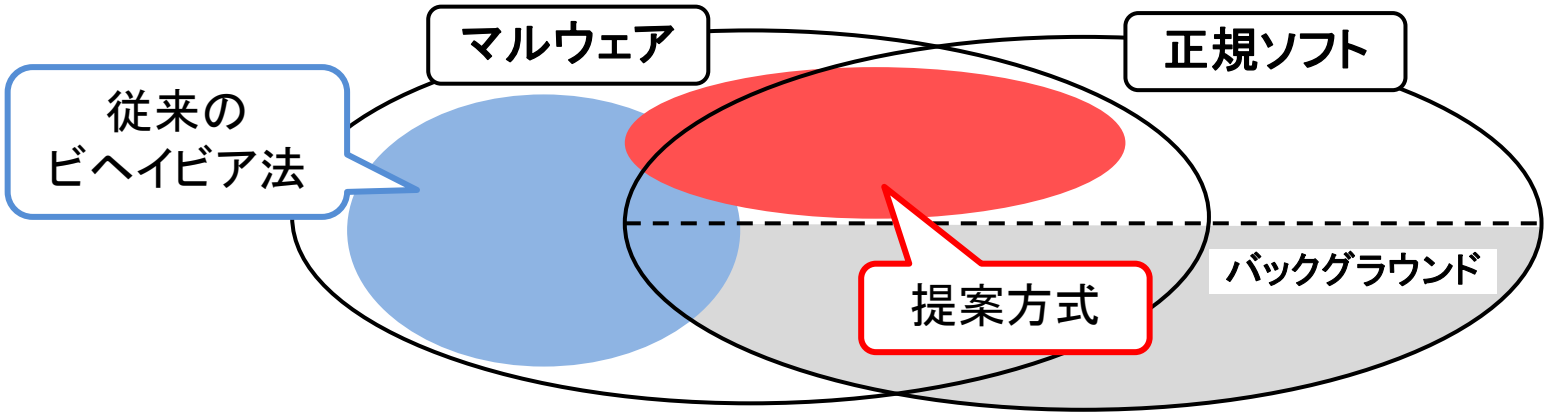
- **実行ファイル作成が観測された際のイベント**
  - インストーラ（Dropbox，Skype，etc）の実行
  - Explorerによる実行ファイルのコピー
  - Webブラウザ（IE，Chrome）による実行ファイルのDL
  - Lhaplusによる実行ファイルを含むZIPの解凍
  - copyコマンドによる実行ファイルのコピー
  - Visual Studio C++ 2010（link.exe）によるビルド
- **自動実行への登録が観測された際のイベント**
  - インストーラ（Dropbox，Skype，etc）の実行
  - Skypeの設定による自動実行への変更
  - レジストリエディタやregコマンドによる自動実行に関するエントリへの書き込み
  - scコマンドによるサービスの登録

→ いずれも，ユーザの意図したタイミングで行われている

→ ユーザは承認ダイアログで，正しく許可を発行できる

従来のビヘイビア法との比較

- 各手法が検出する振る舞いの範囲



- マルウェアと正規ソフトウェアの振る舞いの共通部は…

従来のビヘイビア法

正規ソフトウェアで検出した場合，誤検知になってしまう

提案方式（承認機構）

ユーザの判断を借りることで，一部を誤検知なしに検出可能

提案方式の課題

提案方式の原理的な課題

- 改造された正規ソフトウェアが，ユーザが危険な処理を意図したタイミングと同時に，悪意を働く場合，ユーザは承認ダイアログで許可を与えてしまうという問題
- 例えば・・・
  - インストーラを改造し，マルウェアを同時にインストールする
  - メーラを改造し，メール送信時に本文を改変してメールを送信する
- 根本的な解決は難しい　⇒　正規の方法でソフトウェアを入手するなどのユーザによる対策

実装における課題

- APIフック回避
  - プロトタイプシステムはユーザモードのAPIフックであるため，APIフックの対策が可能
    ⇒　カーネルモードのAPIフックを採用することで，APIフックの回避がより困難になる
- 承認ダイアログ回避
  - プログラム的に承認ダイアログの許可ボタンをクリックできてしまう問題
    ⇒　UACのセキュアデスクトップのようなものを実装する必要がある
- ホワイトリストによる回避
  - マルウェアが自身をホワイトリストに登録してしまう問題
    ⇒　管理者権限がなければホワイトリストに登録できないようにすることで解決