

SandMan入門

Nicolas RUFF / EADS-IW SE/CS
nicolas.ruff [à] eads.net

Matthieu SUICHE
msuiche [à] gmail.com - <http://www.msuiche.net/>

概要

- 当社はWindowsの「ディスクの休止」機能を詳細に分析してきた
 - 一般名「ハイバネーション」
- ハイバネーションファイルの読み書きのための、Cライブラリを記述
- 以下の環境で一部のアプリケーションを実証する予定:
 - 攻撃コンピューティング
 - 防御コンピューティング
 - 法医学

ハイバネーションとは何か?

- Microsoftの「ディスクの休止」機能の名称
 - 今日の全OSで利用可能
- システム状態がディスクに完全にバックアップされる
 - メモリおよびプロセッサの状態を含む
 - 「ゼロパワー」スリープモード
- Windows 2000以降に導入
- コマンドライン制御:
 - POWERCFG /HIBERNATE
 - SHUTDOWN /H

デバッグの問題点

- デバッグで電力遷移が問題になる
 - ハイバネーション状態への移行はデバッガに有利ではない
 - KdDeleteAllBreakpoints() など
 - ハイバネーション状態からの復旧はNTLDRにより行われる
 - デバッガアタッチフックの前 ([1]を参照)
- ただし全体に占める割合は非常に小さい
 - 「デッドリスティング」分析が可能
 - エントリポイントからのコールチェーン:
 - NtShutdownSystem()
 - NtSetSystemPowerState()
 - PopSleepSystem()
 - PopInvokeSystemStateHandler()
 - PopSaveHiberContext()
 - <コアプロセッシング>

ファイル形式

フィールド	コンテンツ
ヘッダー	PO_MEMORY_IMAGE構造
ページリスト	不確定 - ローダー用の「フリーページ」のリストのなる場合がある
プロセッサ状態	CONTEXT + SPECIAL_REGISTERS構造
メモリ範囲アレイ#1	ヘッダー: リストエントリカウント + 次のリストオフセット + チェックサム リスト: 最大255エントリ リストエントリ: 開始ページ + エンドページ + チェックサム
Xpressブロックアレイ#1	マジック: “\x81\x81xpress” (Windows > 2000) ヘッダー: サイズ + チェックサム + その他 コンテンツ: 圧縮データ
メモリ範囲アレイ#2	(...)

ファイル形式 – 詳細

- ヘッダー
 - PO_MEMORY_IMAGEがシンボルのデバッグ時にエクスポートされる
 - ただし、この構造はWindowsのバージョンごとに変化する
 - マジックバイトは以下になる:
 - **hibr**: ハイバネーションファイルは有効、システムをブート時に再開する
 - Vistaは大文字を使用 (HIBR)
 - **wake**: ハイバネーションファイルは無効、システムを新規に起動する
 - **link**: サポートされるが、現時点で更新されていない

ファイル形式 - 詳細

- メモリ範囲アレイ
 - リストカウントは32ビットで格納されるが、カウントは常に0xFF（最後の範囲を除く）
 - ページはオーダーされない
- Xpressブロック
 - 1 Xpressブロック = 0x10物理ページ（最後のブロックを除く）
 - Windows 2000: RtlCompressBuffer() を使って圧縮
 - 内部的「LZNT1」と呼ばれる圧縮方式
 - その他の圧縮方式も利用できるが使用されない
 - 当社ではサポートするが☺
 - Windows > 2000: 内部関数XpressEncode() を使って圧縮
 - RtlCompressBuffer(COMPRESSION_ENGINE_HIBER)は使われたことはない

ファイル形式 - 詳細

- 各アレイについて:
 - Σ (メモリ範囲内のページ) == Σ (Xpressブロック内のページ)
 - Xpressブロックはそれぞれ 0x10 ページを保持
- その他のランダムノート:
 - ほとんどのチェックサムはゼロに設定される (Windows > 2000)
 - すべてページアラインされる
 - 1ページ = 0x1000 バイト
 - 多少のバリエーションのある OS フィンガープリント機能が可能
 - PO_MEMORY_IMAGE ヘッダーの変更により必要
 - 圧縮方式の選択に必要 (Windows 2000)
 - 特殊なメモリレイアウト (/PAE) を構成レジスタを通じて検出できる

SandMan ライブラリ

- SandMan ライブラリのウィッシュリスト:
 - Windows のバージョンに関わりなく、任意のハイバネーションファイルを解析する機能
 - 32/64ビットを含む
 - 正しく文書化されたライブラリ
 - Python のバインディング
 - クールなサンプルアプリケーション
 - 「dd」形式のファイルに変換
 - 検索とパッチページ (高速ルックアップを含む)
 - ページの追加
 - スクリーンショット
 - 使いやすいGUI
- 現在提供できるライブラリを紹介します ☺
 - デモ

使用事例

- 攻撃コンピューティング
 - スリープ状態のマシンをパッチ
 - ターゲット#1 : nt!SeAccessCheck()
 - ターゲット#2 : msv1_0!MsvpPasswordValidate
 - データ抽出
 - ページ化されていないプールを含むすべてがディスクに格納
 - PatchGuardを削除、または符号なしドライバをローディング、希望者はいませんか? 😊

使用事例

- 防御コンピューティング
 - マルウェアの検出
 - コード非表示技術を思いつかない
 - ハイバネーションファイル内にコードがなければ、実行が再開されない
 - ただしハードウェアを信頼する必要がある
 - ボーナスクエスチョン: ハイバネーションの間のハイパーバイザページはどのようなようになるのか?
 - そのようなスライドでは、「ブルーピル」といった業界用語は使用しません ☺

使用事例

- ハイバネーションを通じた法医学
 - DFRWS 2005以降、ライブのメモリ分析への関心が高まっている
 - 攻撃型ワークの例:
 - Meterpreter (Metasploitプロジェクト)
 - Syscallプロキシ化 (CORE Impact)
 - 防御型ワークの例:
 - PTFinder (Andreas Schuster)
 - MemParser (Chris Betz)
 - Windows Memory Forensics Toolkit (Mariusz Burdach)
 - PMODump (TRUMANプロジェクト)
 - FATkit (4tphi)
 - Volatools (Komoku)
 - Volatility (Volatile Systems)
 - Oracleメモリ分析 (Black Hat 2007)
 - Etc.

使用事例

- ハイバネーションを通じた法医学
 - 依然としてメモリ収集が課題
 - ハードウェア技術が絶対的な前提条件
 - 専用のPCIハードウェア
 - IEEE 1394バス
 - ソフトウェア技術はWindowsのバージョンに依存
 - dd “\Device\PhysicalMemory”
 - ZwSystemDebugControl
 - ドライバのロード
 - 「現場で」最も一般的な技術はBSoD + 完全なメモリダンプ

使用事例

- ハイバネーションを通じて法医学
 - 長所:
 - ハードウェアの前提条件がない
 - 首尾一貫したシステム状態
 - アトミックなハイバネーション+ページファイル獲得
 - マシンの動作をシームレスに再開できる
 - リブートなしでハイバネーションを起動できる
 - ハイバネーションファイルを「dd形式の」メモリダンプに変換できる
 - プロセッサコンテキスト（CR3レジスタを含む）の利用が可能
 - 短所:
 - 物理メモリが100%保存される保証がない

使用事例

- ハイバネーションファイルによる法医学
 - ハイバネーションファイルは消去されない
 - 最初のページは0で埋められる
 - » 完全なメモリ再構築は避けられない
 - 機密データ（パスワードと鍵）を保持できる
 - ハイバネーションファイル内にスラックスペースが存在する
 - ハイバネーションファイルは事前に物理メモリサイズに割り当てられる
 - 効果的な使用は物理メモリの使い方による
 - Xpressブロックは「xpress」ヘッダーを指定して抽出できる

結論

- ハイバネーションファイルは実際には簡単な作業にすぎない
 - Windows 2000以降に導入
 - 実際には理解しやすい
- 利用（悪用）の可能性がある
 - 「最高のLiveKD」
 - Rootkit検出
 - 生きた記憶の法医学
- ショーをお楽しみください

(抜粋) 参考文献

- 「Windows起動方法」 (パート2)
 - <http://blogs.msdn.com/ntdebugging/archive/2007/06/28/how-windows-starts-up-part-the-second.aspx>
- Nirsoft: Vistaカーネル構造
 - http://www.nirsoft.net/kernel_struct/vista/
- プロセス、スレッド、仮想メモリ
 - <http://www.i.u-tokyo.ac.jp/edu/training/ss/msprojects/data/07-ProcessesThreadsVM.ppt>
- Intel - Volume 3 システムプログラミングガイド
 - 第3章 – 保護モードメモリ管理
- Microsoft Vista SP1の概要
 - <http://download.microsoft.com/download/9/0/d/90da9663-815a-4ce8-88c0-2b9f54c69efe/windows%20vista%20service%20pack%201%20beta%20overview.pdf>