仮想環境に依存しない動的解析システム”egg”の設計と実装

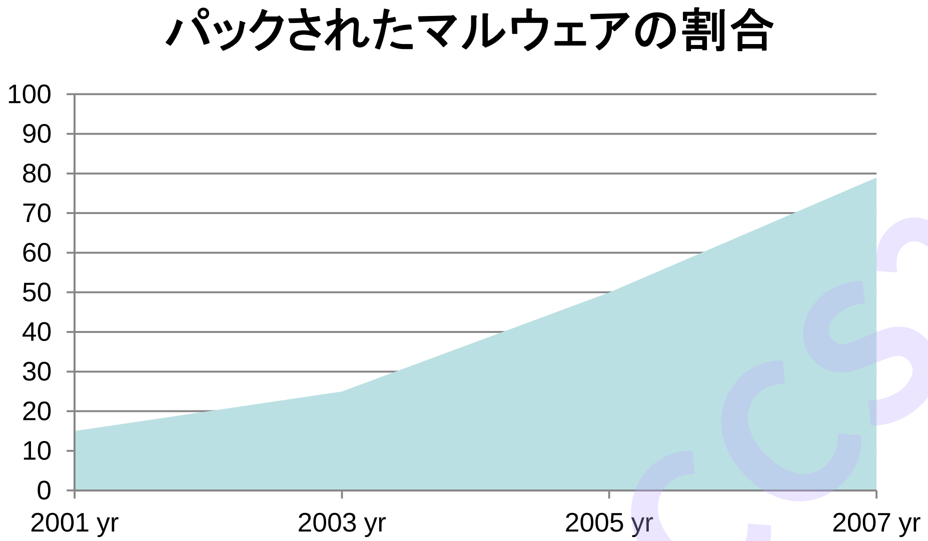
Fourteenforty Research Institute, Inc.
http://www.fourteenforty.jp

株式会社フォティーンフォティ技術研究所

アジェンダ

- 背景と問題
- “egg”とは
  - デモ: 基本的な機能
- 実装 (Ring-0における汚染追跡)
  - デモ: 汚染追跡
- “egg”の制約と今後
- まとめ

背景：増え続ける新種のマルウェア



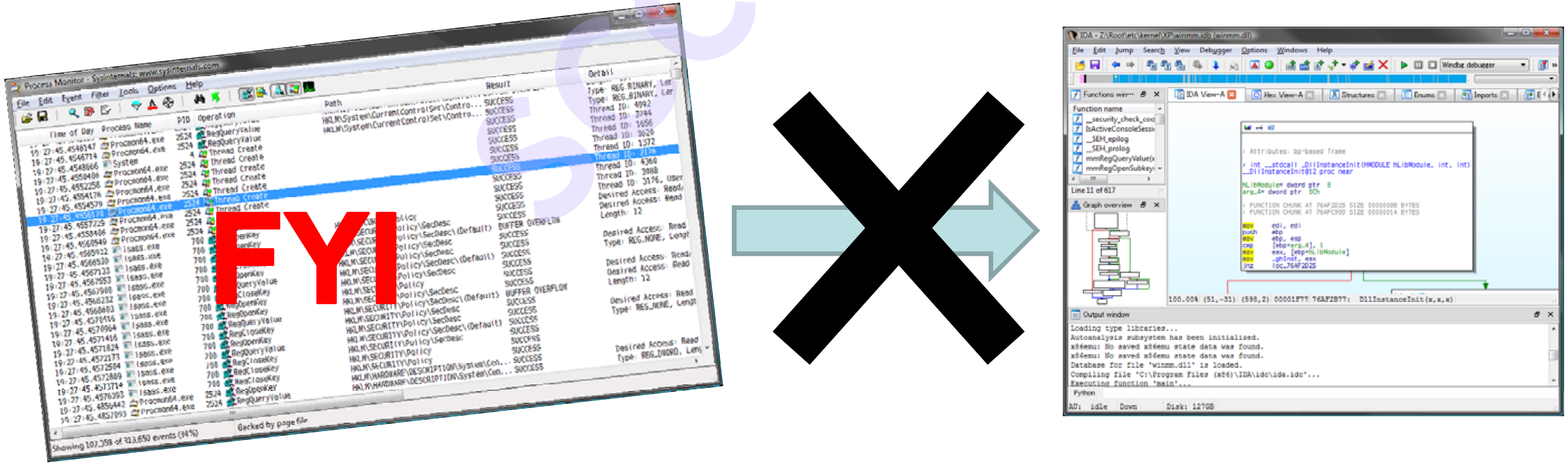
2007年時点で80%のマルウェアがパックされている

- それぞれのマルウェアを手動で解析することは不可能
- 自動解析が重要度が増している.

Source:
2001-2005 : McAfee Sage vol.1 issue 1
2007 : Panda Research (http://research.pandasecurity.com/malwareformation-statistics/)

問題：従来の動的解析システムの欠点

- より詳細な解析に進める際に有用な情報が少ない
- カーネルモードコードを解析できない
- プロセスを超えて拡散するマルウェアの解析が困難



問題：先進的なシステム(仮想マシンを用いるもの)

- 先進的なシステムでは先ほどの問題は解決されている☺
  - Anubis
  - Ether
    - これらはカーネルモードコードを命令単位で解析できる
  - BitBlaze and Renovo
    - これらは加えてプロセスを超えて拡散するマルウェアを「汚染追跡(Taint tracing)」と呼ばれる技法を使って解析できる

- しかしながらこれらのシステムは、仮想マシン検知技術によってマルウェアから検知されてしまう☹

問題のまとめ

動的解析システムの種類	従来システム	先進的システム (仮想マシンを用いるもの)
有用な情報の取得	不十分	良い
カーネルモードコードの解析	不十分	良い
拡散するマルウェアの解析	不十分	良い
仮想マシン検知技術への耐性（影響のなさ）	良い（影響を受けない）	不十分（影響を受ける）

- これらの問題を解決するために”egg”を開発

“egg”とは？

- “egg”はWindowsデバイスドライバーとして実装された動的解析システム.

- “egg”は以下の特長を持つ:

1. より詳細な情報を得ることが可能.
2. カーネルモードコードの解析が可能.
3. 拡散するマルウェアを自動的に解析可能

- もちろん、仮想マシン検知技術の影響を受けない
- さらに多くのアンチデバッグ技術によって検知されない

”egg” はどのような情報を収集するか

1. API 引数（IN, OUT, INOUT）および戻り値

```
BOOL WINAPI ReadFile(
  __in          HANDLE hFile,
  __out         LPVOID lpBuffer,
  __in          DWORD nNumberOfBytesToRead,
  __out_opt     LPDWORD lpNumberOfBytesRead,
  __inout_opt   LPOVERLAPPED lpOverlapped
);
```

”egg” はどのような情報を収集するか

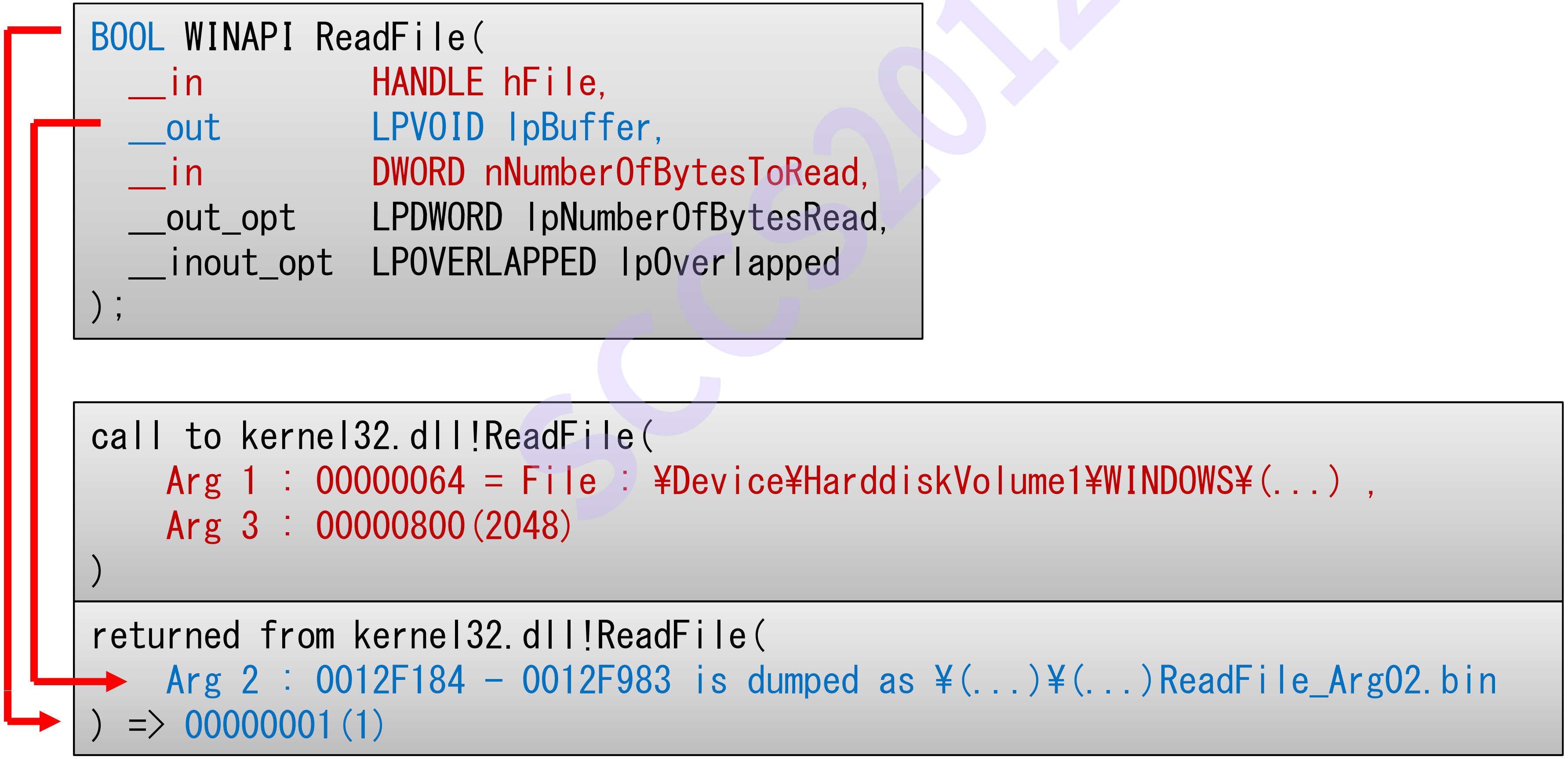
1. API 引数（IN, OUT, INOUT）および戻り値

```
BOOL WINAPI ReadFile(
  __in          HANDLE hFile,
  __out         LPVOID lpBuffer,
  __in          DWORD nNumberOfBytesToRead,
  __out_opt     LPDWORD lpNumberOfBytesRead,
  __inout_opt   LPOVERLAPPED lpOverlapped
);
```

```
call to kernel32.dll!ReadFile(
    Arg 1 : 00000064 = File : ¥Device¥HarddiskVolume1¥WINDOWS¥(...) ,
    Arg 3 : 00000800(2048)
)
```

”egg” はどのような情報を収集するか

1. API 引数（IN, OUT, INOUT）および戻り値



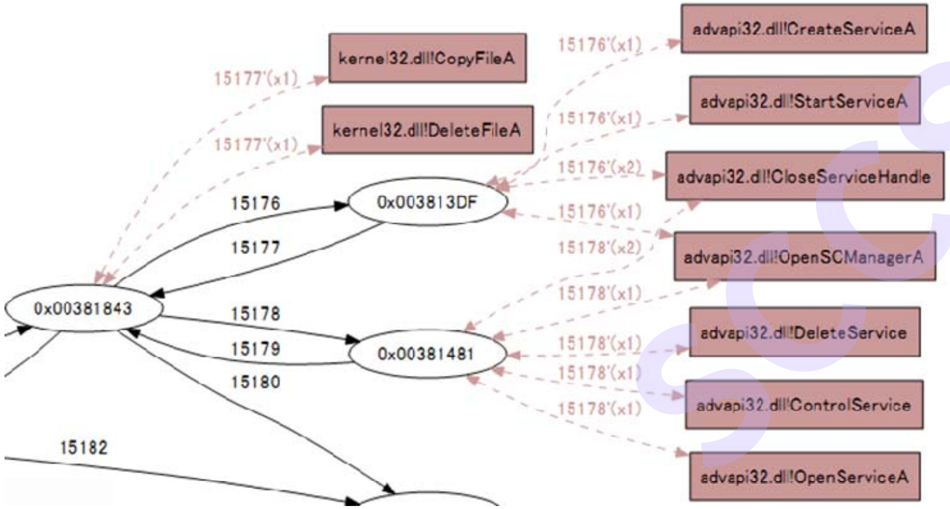
```
BOOL WINAPI ReadFile(
  __in          HANDLE hFile,
  __out         LPVOID lpBuffer,
  __in          DWORD nNumberOfBytesToRead,
  __out_opt     LPDWORD lpNumberOfBytesRead,
  __inout_opt   LPOVERLAPPED lpOverlapped
);
```

```
call to kernel32.dll!ReadFile(
    Arg 1 : 00000064 = File : ¥Device¥HarddiskVolume1¥WINDOWS¥(...) ,
    Arg 3 : 00000800(2048)
)
```

```
returned from kernel32.dll!ReadFile(
    Arg 2 : 0012F184 - 0012F983 is dumped as ¥(...)¥(...)ReadFile_Arg02.bin
) => 00000001(1)
```

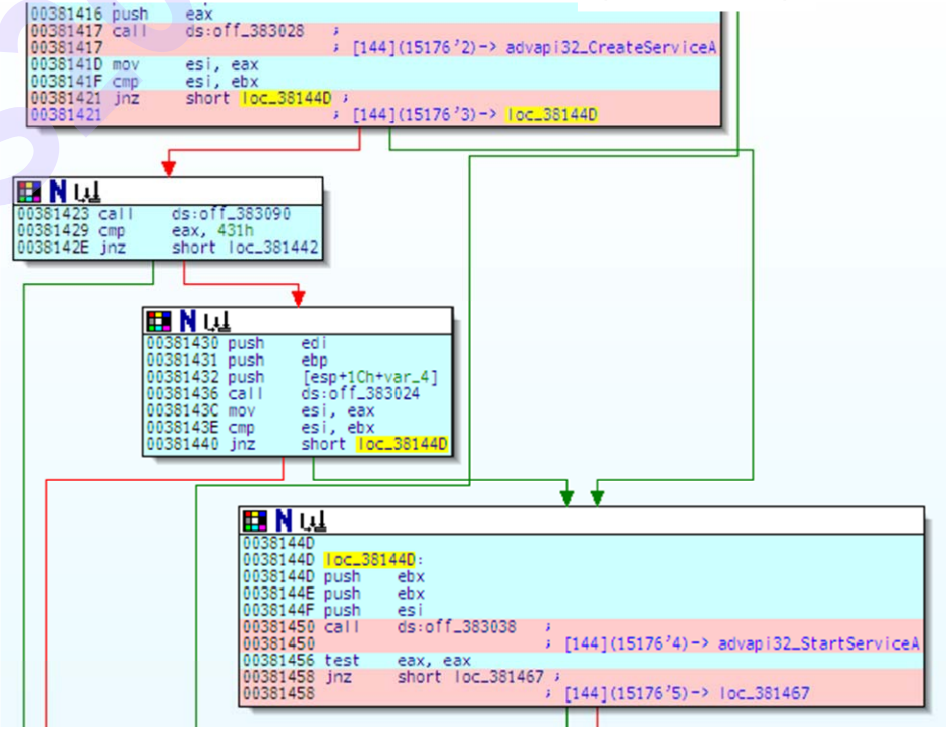
”egg” はどのような情報を収集するか

2. コールグラフ
3. 分岐情報



コールグラフ
(made with Graphviz)

分岐情報
(with IDA Pro)



```
00381416 push    eax
00381417 call    ds:off_383028   ;
00381417                         ; [144](15176'2)-> advapi32_CreateServiceA
0038141D mov     esi, eax
0038141F cmp     esi, ebx
00381421 jnz     short loc_38144D ;
00381421                         ; [144](15176'3)-> loc_38144D
```

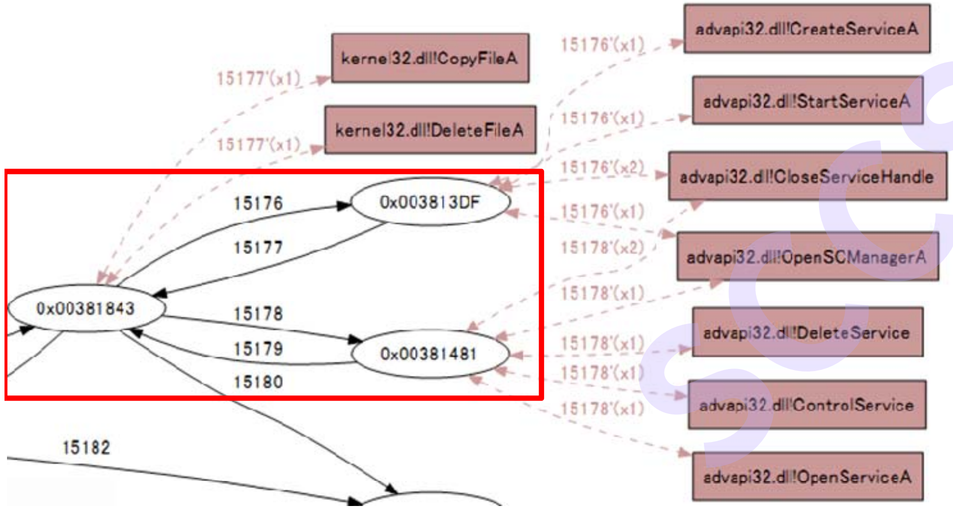
```
00381423 call    ds:off_383090
00381429 cmp     eax, 431h
0038142E jnz     short loc_381442
```

```
00381430 push    edi
00381431 push    ebp
00381432 push    [esp+1Ch+var_4]
00381436 call    ds:off_383024
0038143C mov     esi, eax
0038143E cmp     esi, ebx
00381440 jnz     short loc_38144D
```

```
0038144D
0038144D loc_38144D:
0038144D push    ebx
0038144E push    ebx
0038144F push    esi
00381450 call    ds:off_383038   ;
00381450                         ; [144](15176'4)-> advapi32_StartServiceA
00381456 test    eax, eax
00381458 jnz     short loc_381467 ;
00381458                         ; [144](15176'5)-> loc_381467
```

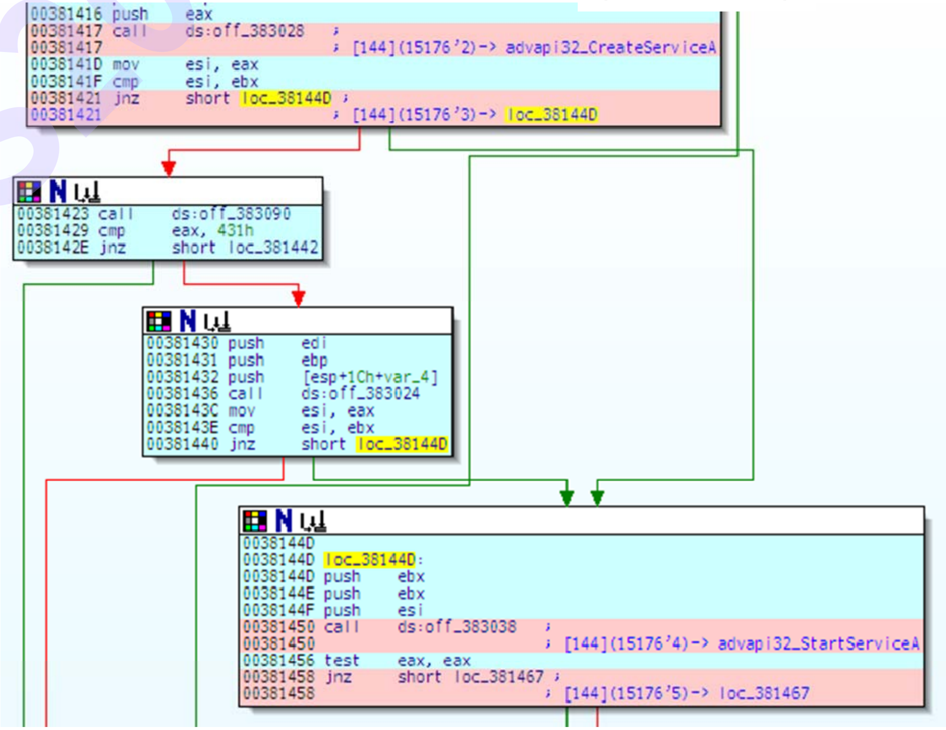
”egg” はどのような情報を収集するか

2. コールグラフ
3. 分岐情報



コールグラフ
(made with Graphviz)

分岐情報
(with IDA Pro)



```
00381416 push    eax
00381417 call    ds:off_383028   ;
00381417                         ; [144](15176'2)-> advapi32_CreateServiceA
0038141D mov     esi, eax
0038141F cmp     esi, ebx
00381421 jnz     short loc_38144D ;
00381421                         ; [144](15176'3)-> loc_38144D
```

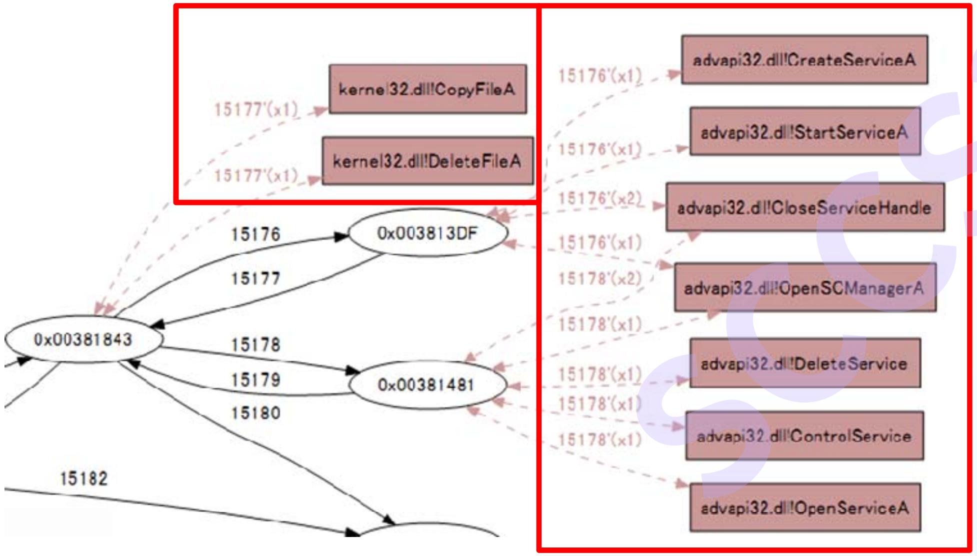
```
00381423 call    ds:off_383090
00381429 cmp     eax, 431h
0038142E jnz     short loc_381442
```

```
00381430 push    edi
00381431 push    ebp
00381432 push    [esp+1Ch+var_4]
00381436 call    ds:off_383024
0038143C mov     esi, eax
0038143E cmp     esi, ebx
00381440 jnz     short loc_38144D
```

```
0038144D
0038144D loc_38144D:
0038144D push    ebx
0038144E push    ebx
0038144F push    esi
00381450 call    ds:off_383038   ;
00381450                         ; [144](15176'4)-> advapi32_StartServiceA
00381456 test    eax, eax
00381458 jnz     short loc_381467 ;
00381458                         ; [144](15176'5)-> loc_381467
```

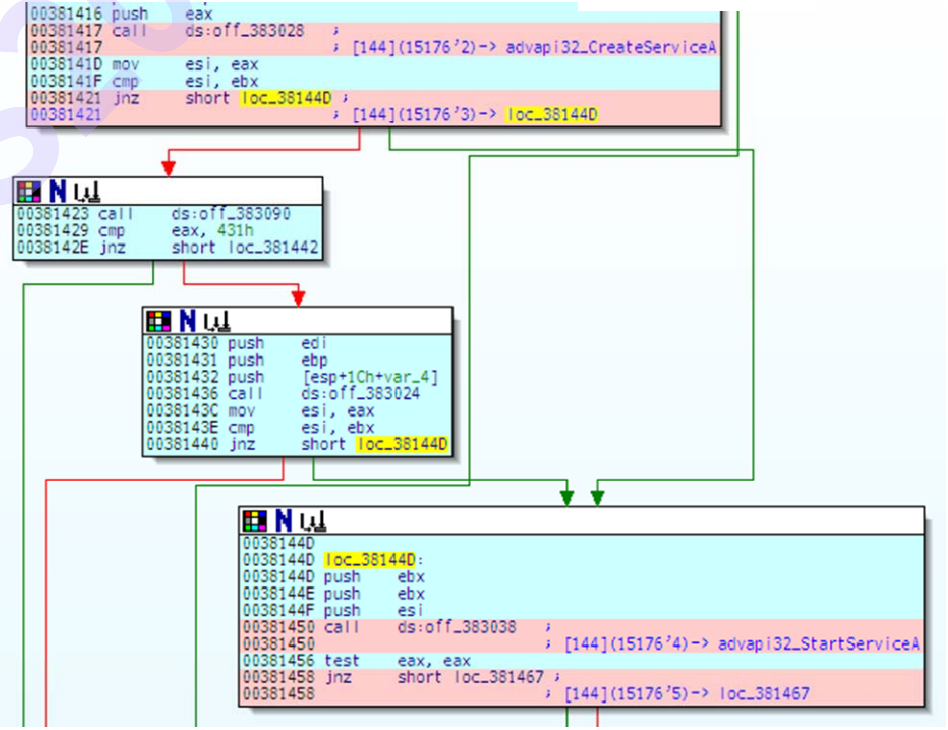
”egg” はどのような情報を収集するか

2. コールグラフ
3. 分岐情報



コールグラフ
(made with Graphviz)

分岐情報
(with IDA Pro)



```
00381416 push    eax
00381417 call    ds:off_383028   ;
00381417                         ; [144](15176'2)-> advapi32_CreateServiceA
0038141D mov     esi, eax
0038141F cmp     esi, ebx
00381421 jnz     short loc_38144D ;
00381421                         ; [144](15176'3)-> loc_38144D
```

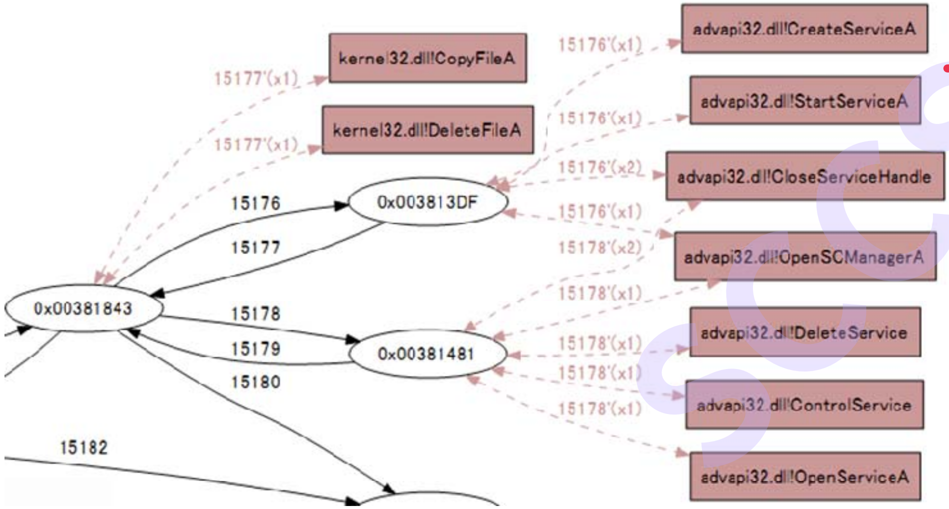
```
00381423 call    ds:off_383090
00381429 cmp     eax, 431h
0038142E jnz     short loc_381442
```

```
00381430 push    edi
00381431 push    ebp
00381432 push    [esp+1Ch+var_4]
00381436 call    ds:off_383024
0038143C mov     esi, eax
0038143E cmp     esi, ebx
00381440 jnz     short loc_38144D
```

```
0038144D
0038144D loc_38144D:
0038144D push    ebx
0038144E push    ebx
0038144F push    esi
00381450 call    ds:off_383038   ;
00381450                         ; [144](15176'4)-> advapi32_StartServiceA
00381456 test    eax, eax
00381458 jnz     short loc_381467 ;
00381458                         ; [144](15176'5)-> loc_381467
```

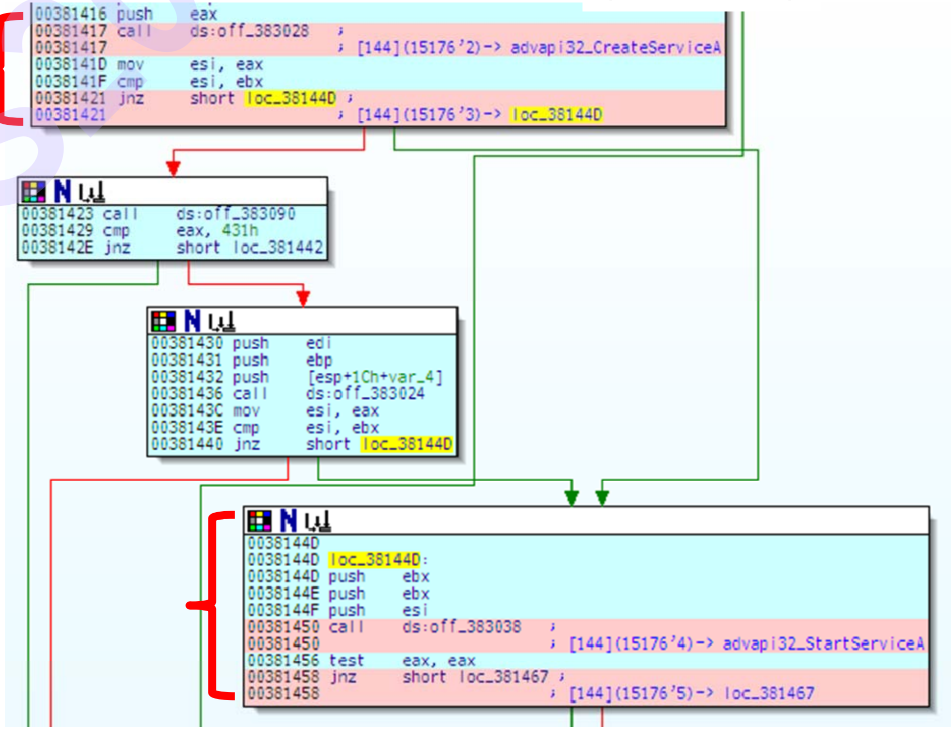
”egg” はどのような情報を収集するか

2. コールグラフ
3. 分岐情報



コールグラフ
(made with Graphviz)

分岐情報
(with IDA Pro)



```
00381416 push    eax
00381417 call    ds:off_383028   ;
00381417                         ; [144](15176'2)-> advapi32_CreateServiceA
0038141D mov     esi, eax
0038141F cmp     esi, ebx
00381421 jnz     short loc_38144D ;
00381421                         ; [144](15176'3)-> loc_38144D
```

```
00381423 call    ds:off_383090
00381429 cmp     eax, 431h
0038142E jnz     short loc_381442
```

```
00381430 push    edi
00381431 push    ebp
00381432 push    [esp+1Ch+var_4]
00381436 call    ds:off_383024
0038143C mov     esi, eax
0038143E cmp     esi, ebx
00381440 jnz     short loc_38144D
```

```
0038144D
0038144D loc_38144D:
0038144D push    ebx
0038144E push    ebx
0038144F push    esi
00381450 call    ds:off_383038   ;
00381450                         ; [144](15176'4)-> advapi32_StartServiceA
00381456 test    eax, eax
00381458 jnz     short loc_381467 ;
00381458                         ; [144](15176'5)-> loc_381467
```

デモ：基本的な機能

- sample.exeの解析

- Sample.exe はもともとのbeepドライバ(beep.sys)を上書きする
- その後beepサービスを再起動することで上書き後のドライバを
- カーネルに読み込ませる

- “egg”は sample.exe と、上書きされたドライバの両方を解析する
.

詳細なコード解析の実装

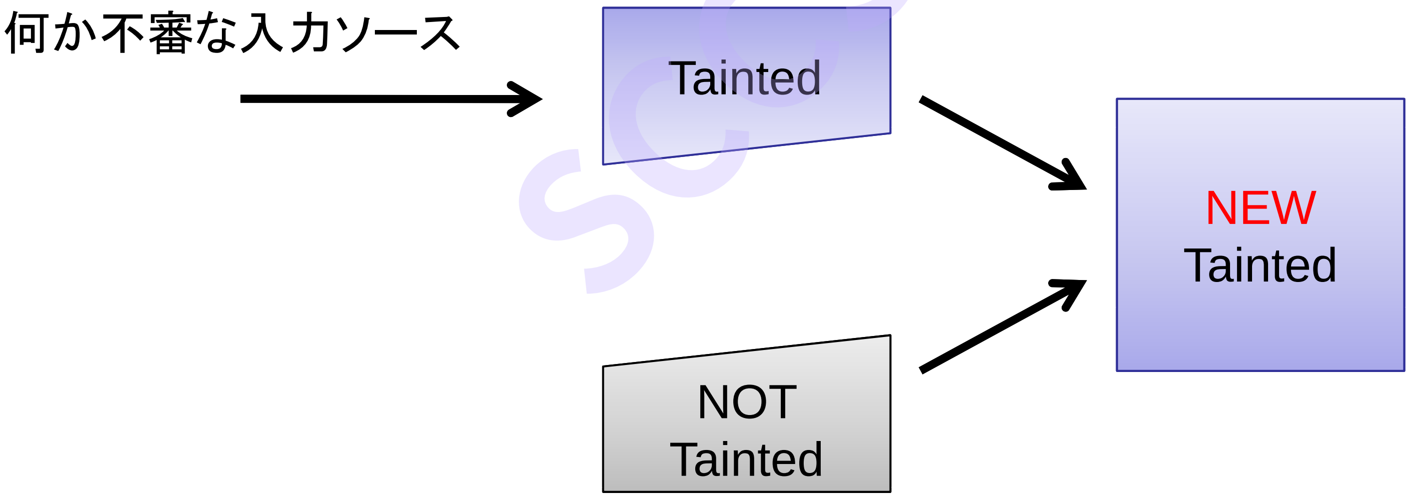
- X86のページ保護とトラップフラグを利用
- →"Stealth Breakpoints"と呼ばれる技法

- 1命令実行されるごとに解析処理を動作させることができる
- カーネルモードとユーザーモードを解析可能
- ユーザーモードから透過的に動作する

Stealth Breakpoints
http://www.acsac.org/2005/abstracts/72.html

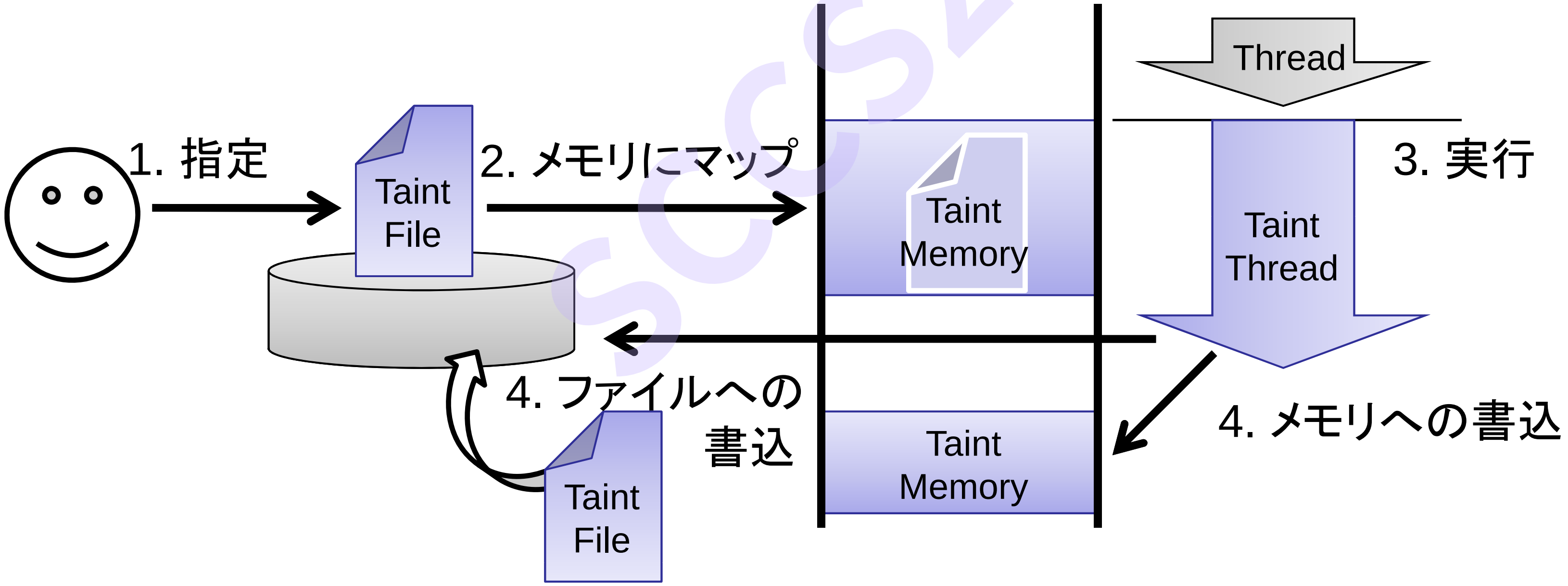
汚染追跡(Taint tracing)とは何か

- 不審な要素を追跡する技法
- 不審な要素を「汚染」としてマークする.
- 「汚染」された要素を使用した要素も「汚染」とみなすことで、「汚染」を自動的に追跡していく



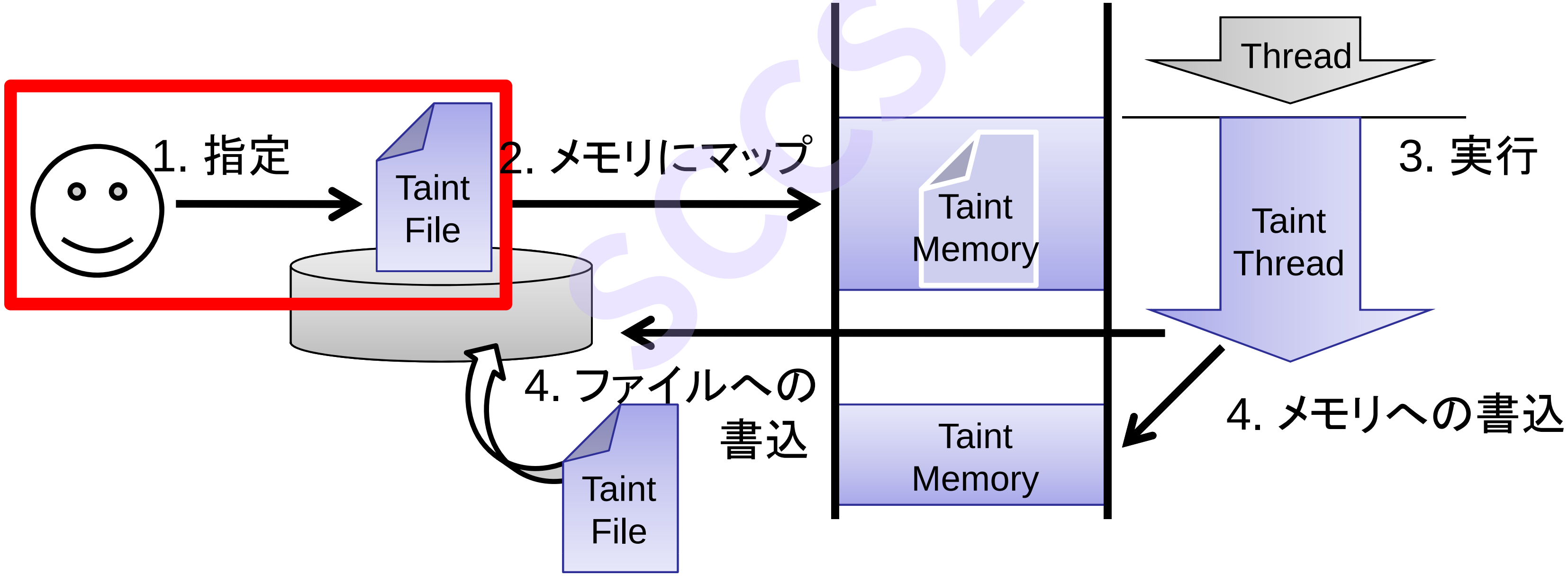
“egg”における汚染追跡の全体像

- egg は汚染追跡をユニークな方法で実装する
- egg の場合、“要素”は、ファイル、仮想メモリ、スレッド

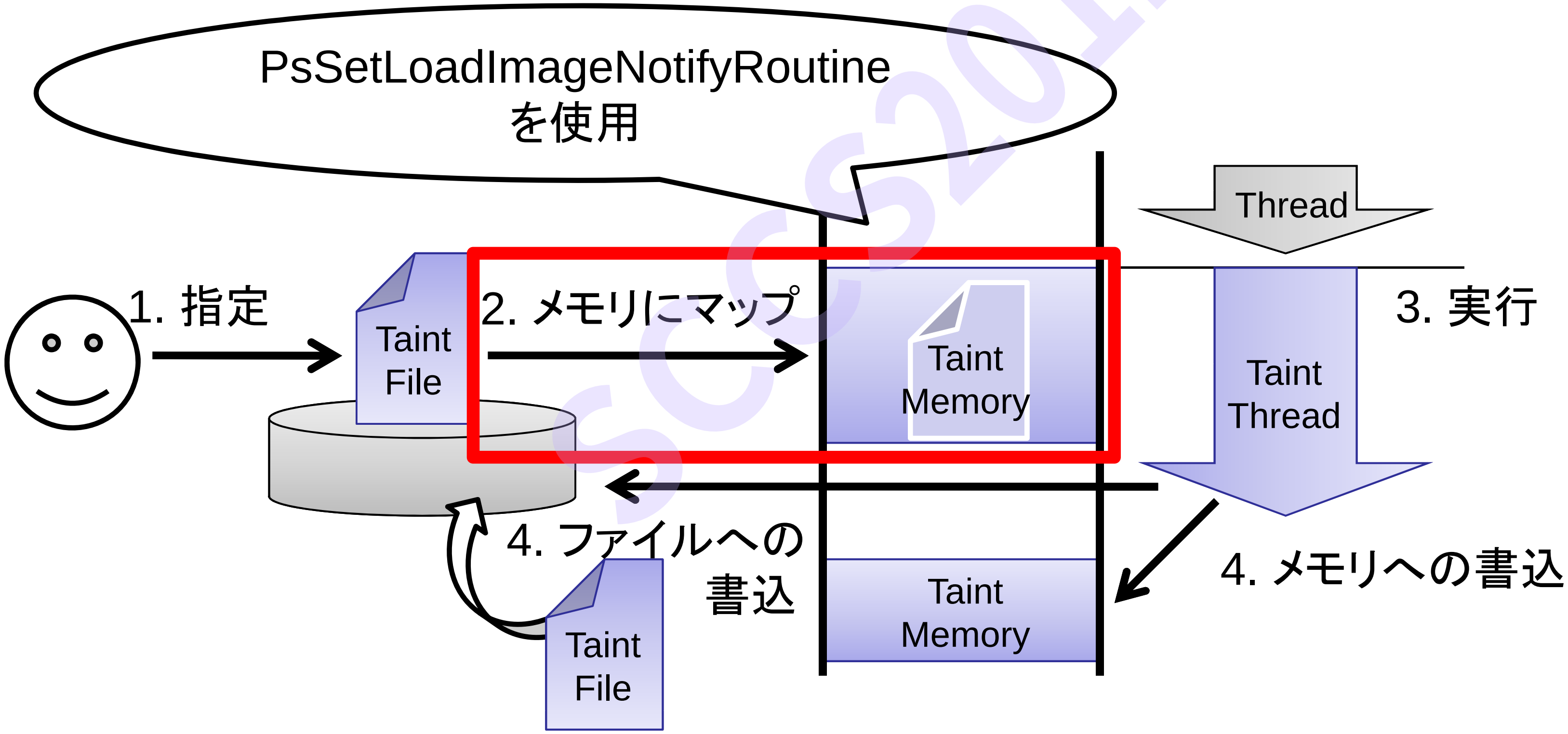


“egg”における汚染追跡の全体像

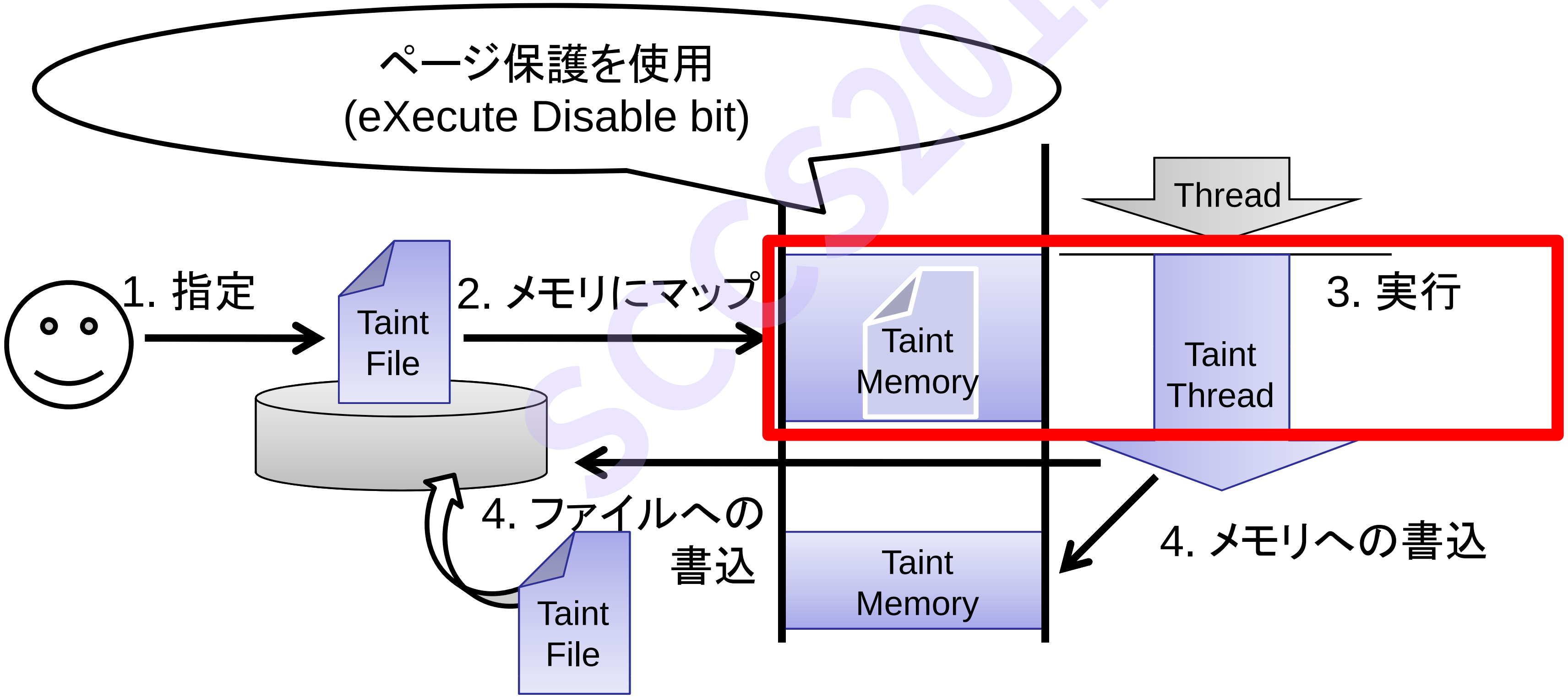
- egg は汚染追跡をユニークな方法で実装する
- egg の場合、“要素”は、ファイル、仮想メモリ、スレッド



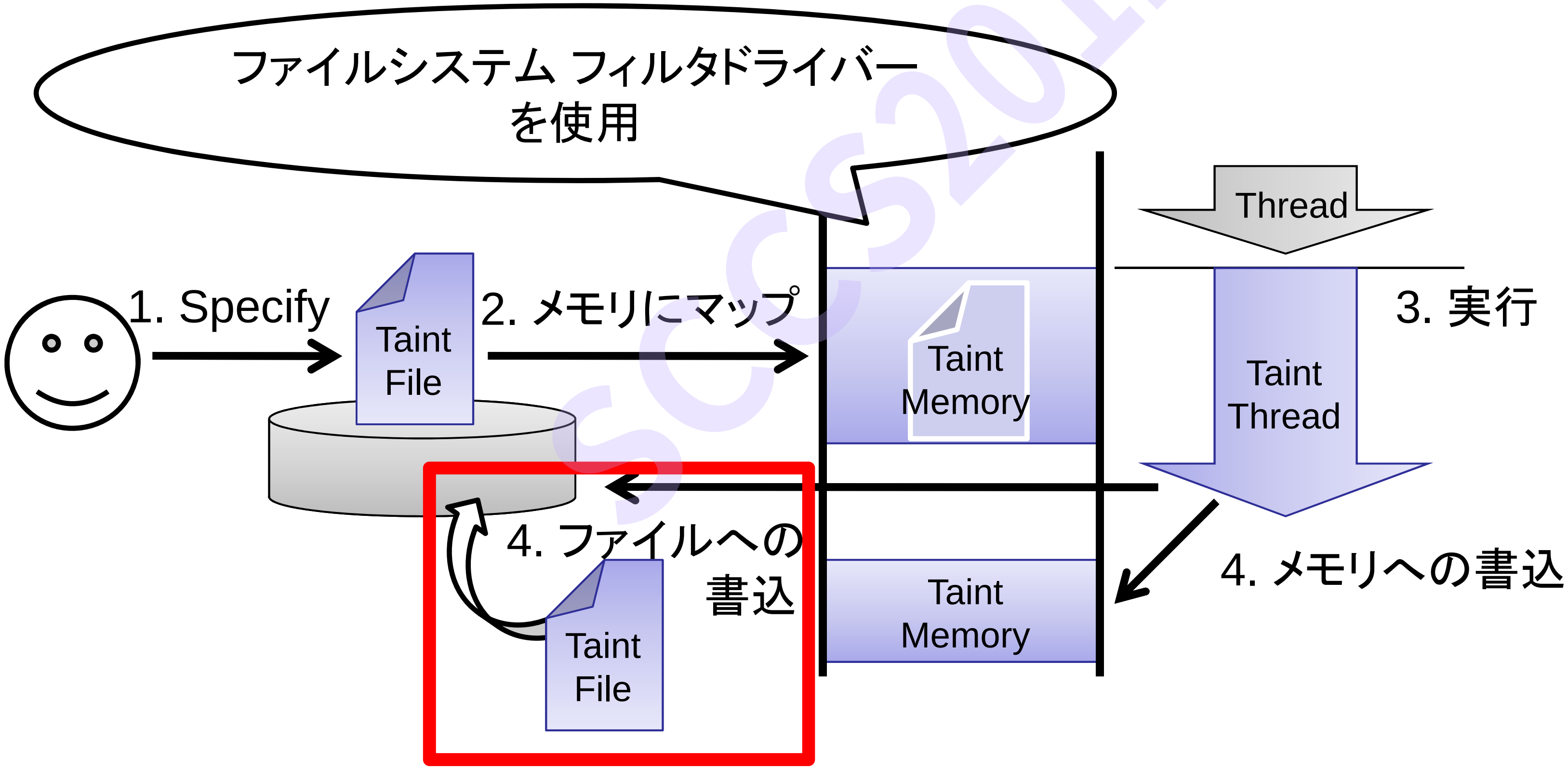
Ring-0における汚染追跡の実装



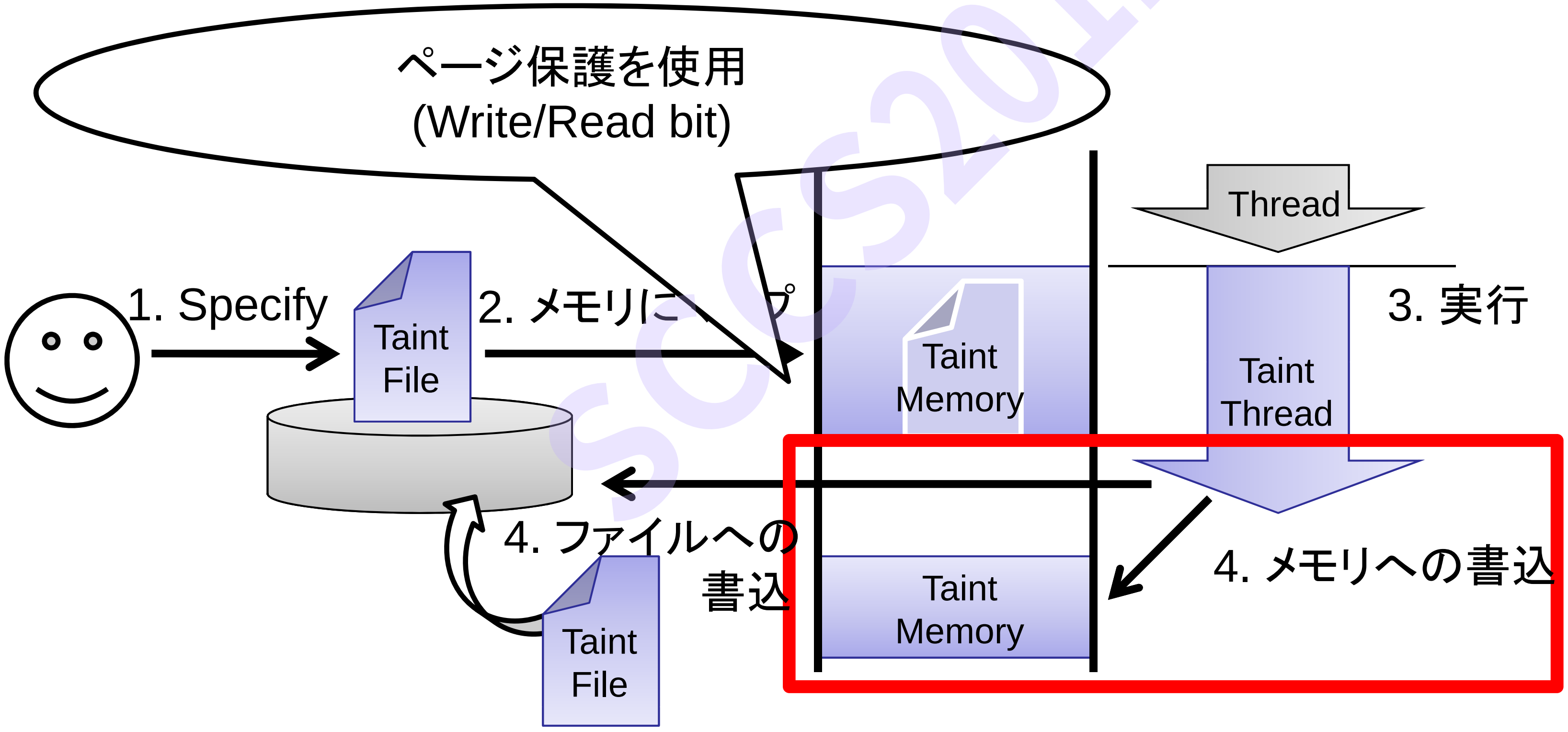
Ring-0における汚染追跡の実装



Ring-0における汚染追跡の実装

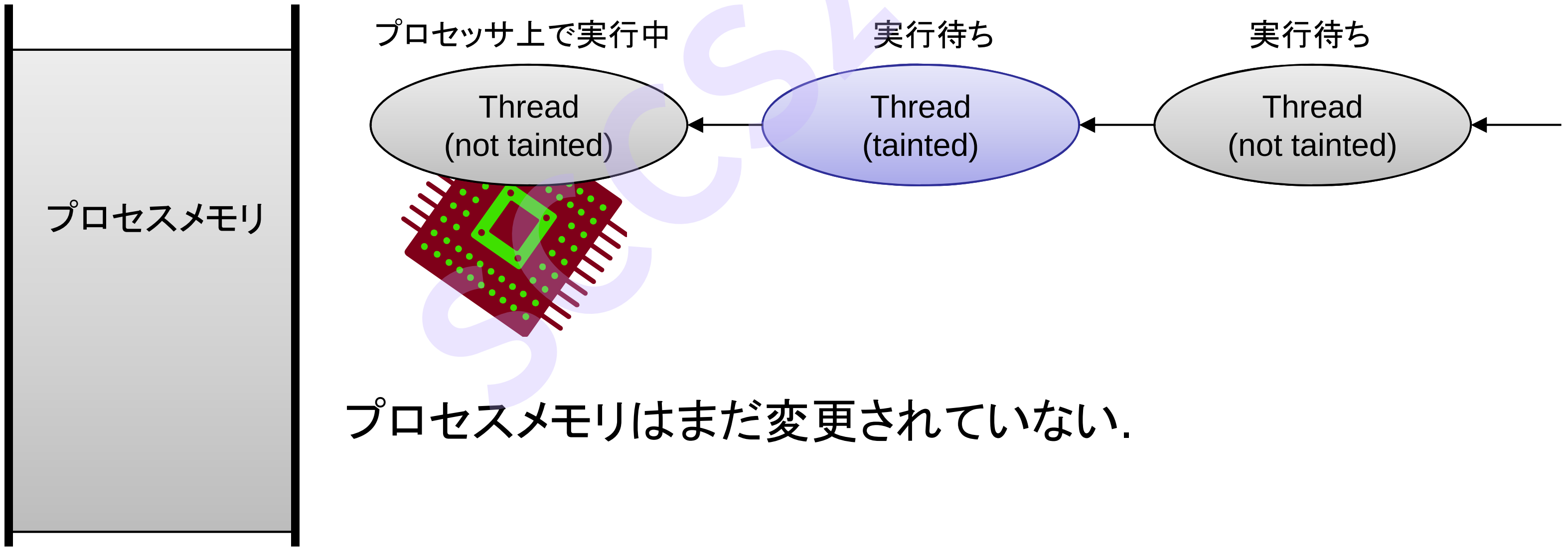


Ring-0における汚染追跡の実装



Ring-0における汚染追跡の実装

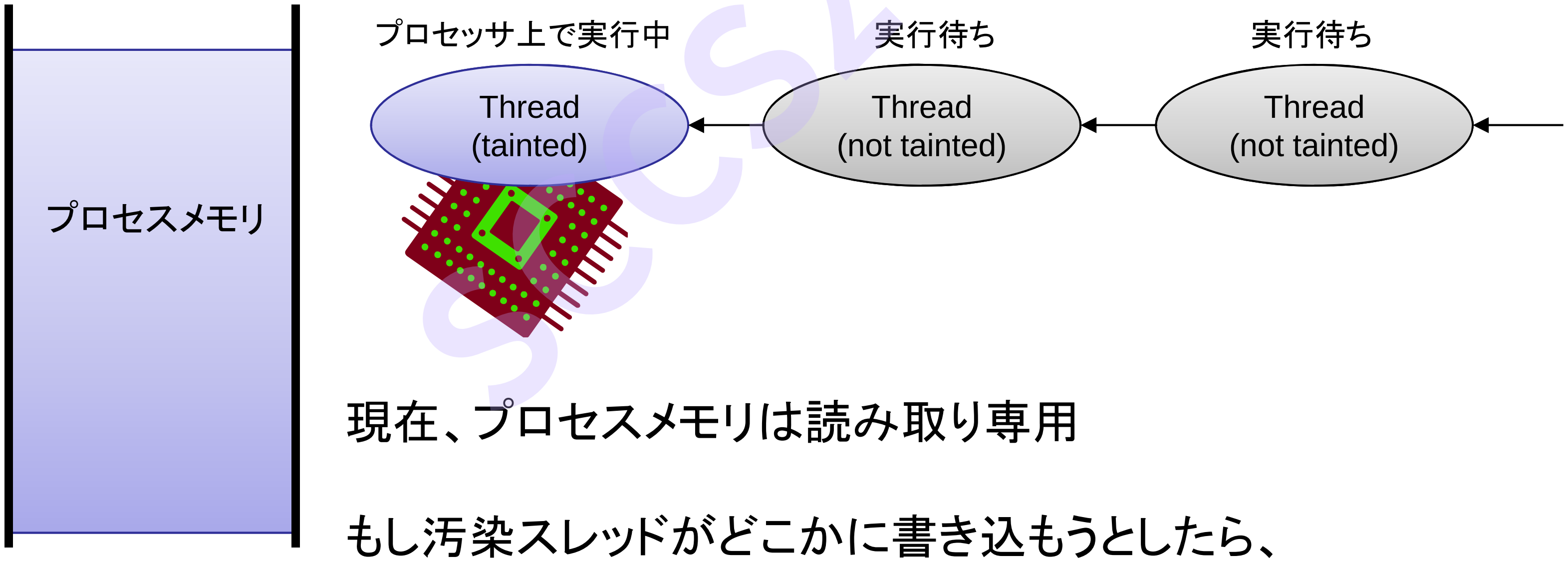
- スレッド安全性を確保するため、eggはWindowsカーネルのスレッドスイッチ関数(SwapContext)をフックする.
- これによりeggはスレッドスイッチの発生を知ることができる.



プロセスメモリはまだ変更されていない.

Ring-0における汚染追跡の実装

- 汚染スレッドがアクティブになったとき、
  eggはすべてのプロセスメモリを読み取り専用に変更する

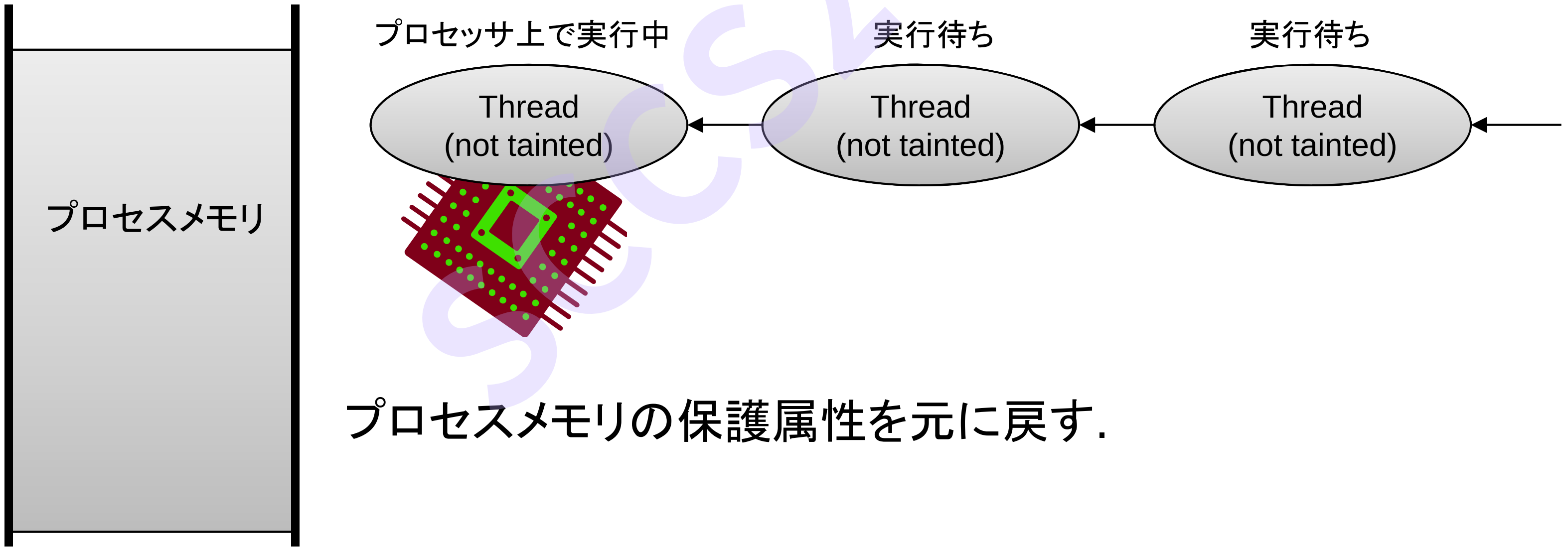


現在、プロセスメモリは読み取り専用

もし汚染スレッドがどこかに書き込もうとしたら、
プロセッサは例外を発生させる。
egg はこの例外を汚染イベントとしてキャッチする。

Ring-0における汚染追跡の実装

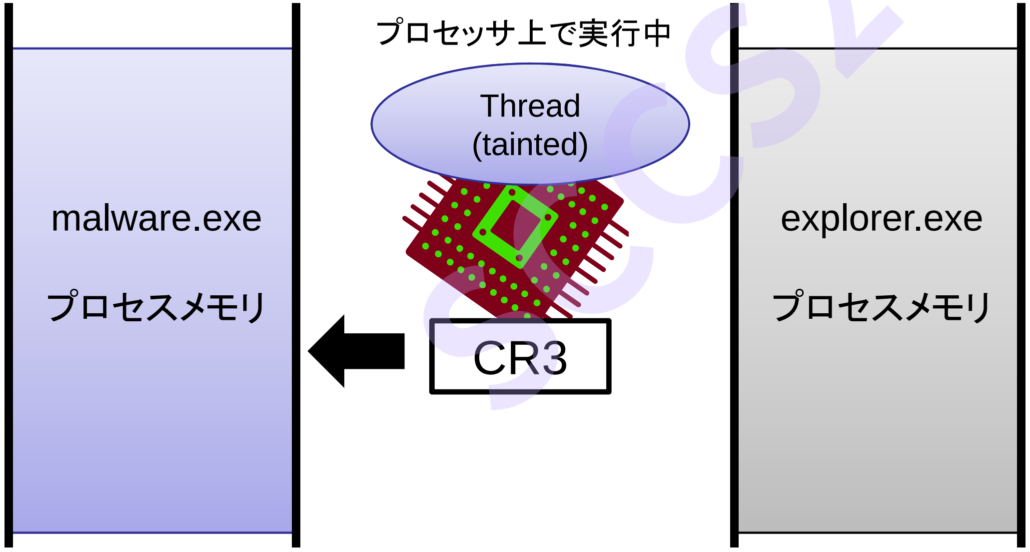
- 汚染スレッドがアクティブではなくなったとき、
  eggはすべてのプロセスメモリの保護属性を元に戻す



プロセスメモリの保護属性を元に戻す.

プロセス間メモリ操作の追跡

- プロセス間メモリ操作を追跡するために、eggはプロセスメモリ空間の切り替え関数(KiSwapProcess)をフックする
- これによりeggはプロセス間のメモリ操作を知ることができる

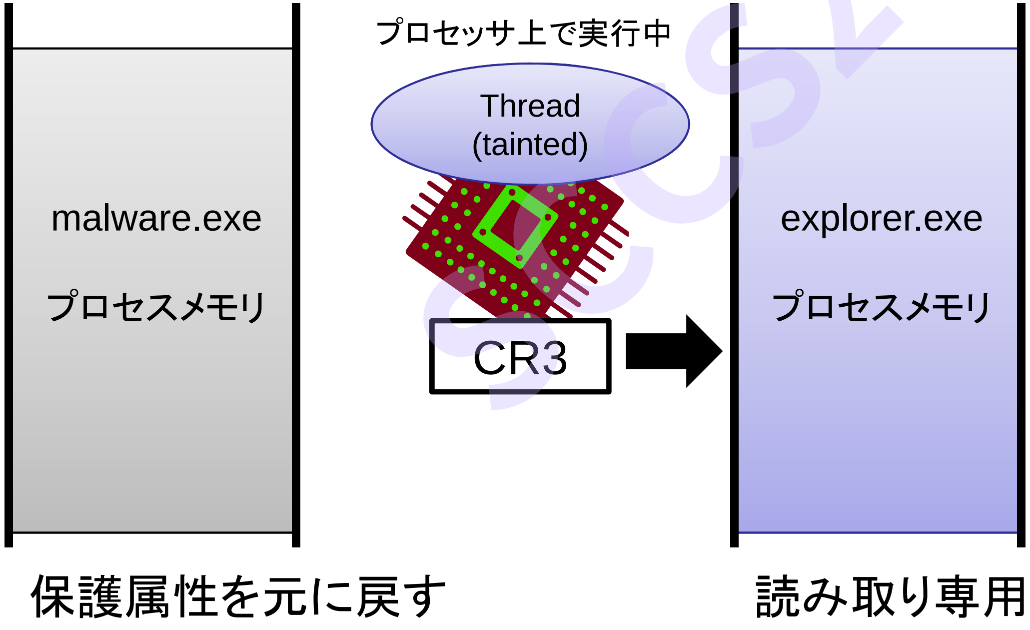


読み取り専用

保護属性はまだ変更されていない

プロセス間メモリ操作の追跡

- 汚染スレッドが別のプロセスメモリで実行されたとき、eggはそのプロセスメモリを読み取り専用に変更する



eggは他のプロセスメモリに対する操作（例：WriteProcessMemory）でも追跡することができる。

デモ：汚染追跡

- サンプルマルウェアはスレッドインジェクションを行う

- サンプルマルウェアは“injector.exe”
- VirtualAllocEx、WriteProcessMemory、CreateRemoteThreadを用いてnotepad.exeにスレッドをインジェクトする
- インジェクトされたスレッドはAllocConsoleとWriteConsoleをループの中で呼び出す
- eggはこのインジェクトされたスレッドも解析する

同レベル特権の問題

- egg はカーネルモードの解析について制限を持つ
  - カーネルモードマルウェアからeggは可視であり、破壊可能

- この制限は仮想マシン検知技術の影響を回避するためのトレードオフ

まとめ

動的解析システムの種類	Egg	従来システム	先進的システム
有用な情報の取得	良い	不十分	良い
カーネルモードコードの解析	可	不十分	良い
拡散するマルウェアの解析	良い	不十分	良い
仮想マシン検知技術への耐性 （影響のなさ）	良い （影響を受けない）	良い （影響を受けない）	不十分 （影響を受ける）

- eggを使用することで解析の所要時間を抑えることができる
- 将来のバージョンでは、より高速かつ安定した動作を目指す.

ありがとうございました

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