多段パックされたマルウェアからのコード取得

中村徳昭† 森井昌克†
伊沢亮一‡ 井上大介‡ 中尾康二‡

神戸大学大学院工学研究科†
独立行政法人情報通信研究機構‡

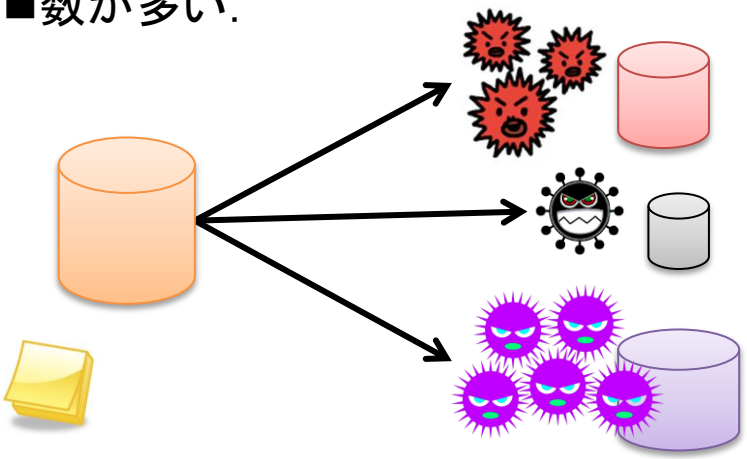
研究背景

◆インターネットの普及に伴いマルウェアが蔓延

被害が多発！

迅速な解析が求められているが，解析には多大な労力が必要

■数が多い．



大まかな機能で分類し，まとめて解析

■解析を妨害するために
暗号化(パッキング)が施されている．

パッキングの手法は巧妙化の傾向に！

研究目的

パッキング

実行可能な形式を保ったまま実行ファイル本来のプログラムコード（以下，オリジナルコードと呼ぶ）を暗号化・圧縮すること．

■通常パッキング　：　オリジナルコードをすべて復号後，順に実行

■多段パッキング　：　オリジナルコードを複数のセクションに渡って実行

◆問題点

展開の処理が複雑で完全なオリジナルコードを取得するのは難しい

不完全でも取得して利用することを考える

研究目的

多段パックされた検体からオリジナルコードを汎用的に取得，マルウェアの類似度判定に利用

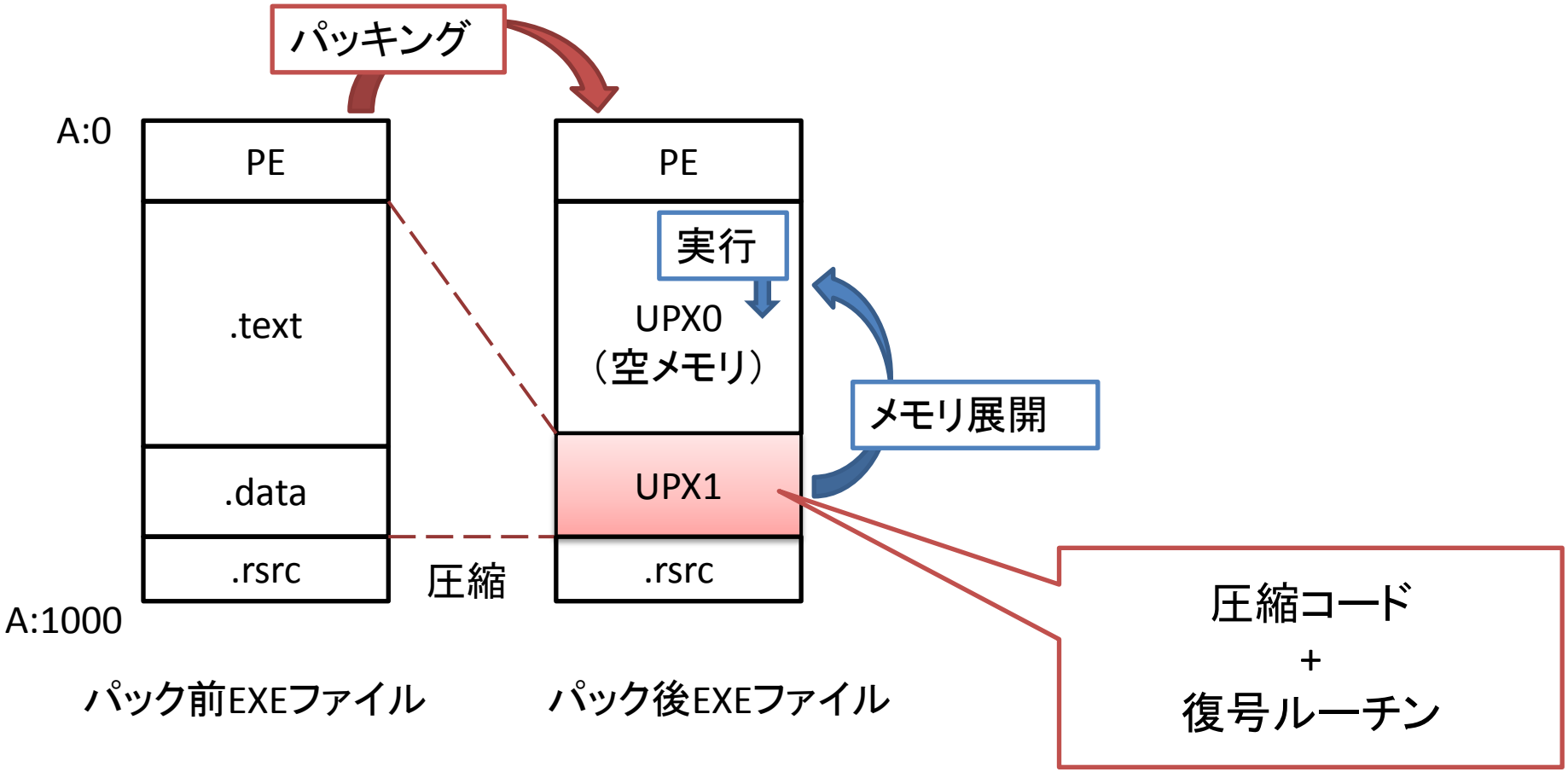
発表の流れ

1. パッキングの構造
   - 通常パッキング
   - 多段パッキング
2. 多段パッキングに対するオリジナルコード取得
   - オリジナルコード取得法
   - 評価実験1
3. コードを用いた類似度判定
   - 類似度判定法，類似度可視化
   - 評価実験2
4. まとめ

発表の流れ

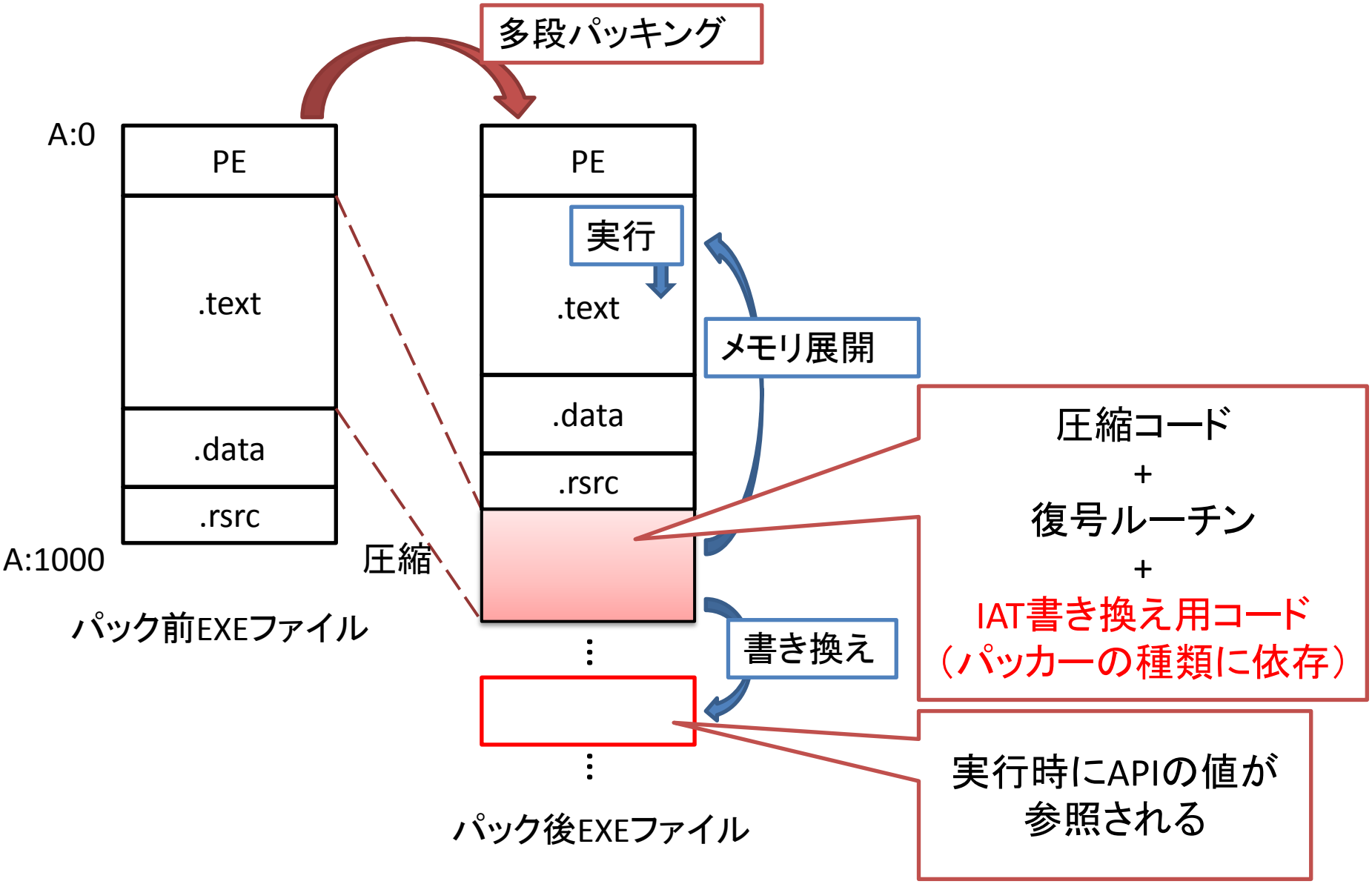
1. パッキングの構造
   - 通常パッキング
   - 多段パッキング
2. 多段パッキングに対するオリジナルコード取得
   - オリジナルコード取得法
   - 評価実験1
3. コードを用いた類似度判定
   - 類似度判定法，類似度可視化
   - 評価実験2
4. まとめ

通常パック(UPX)の構造



UPX1セクションに復号ルーチンと圧縮データが存在し，
UPX0セクションに.text，.data等のセクションがまとめて展開される．

多段パックの構造

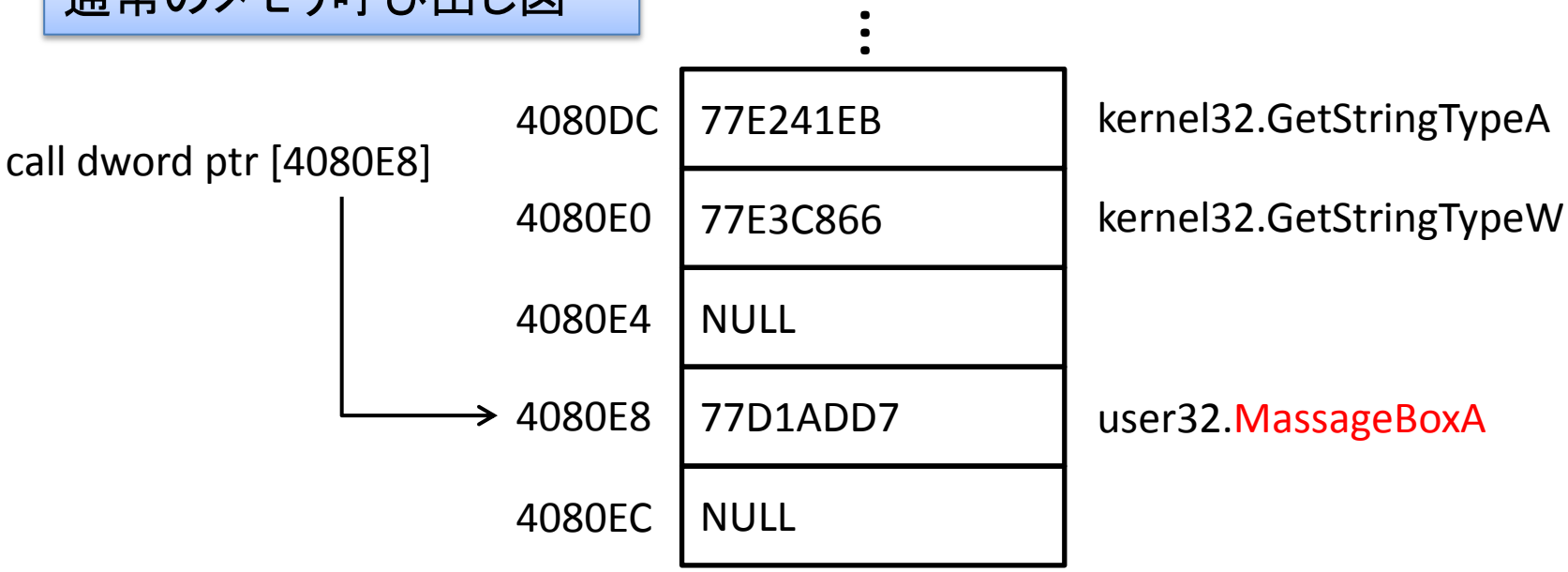

多段パッキング
A:0
PE
.text
.data
.rsrc
A:1000
圧縮
パック前EXEファイル
PE
実行
.text
.data
.rsrc
メモリ展開
書き換え
パック後EXEファイル
圧縮コード
+
復号ルーチン
+
IAT書き換え用コード
（パッカーの種類に依存）
実行時にAPIの値が
参照される

Import Redirection

Import Redirection

IATの値をヒープに作成したダミーコードのアドレスに書き換える手法

通常のメモリ呼び出し図

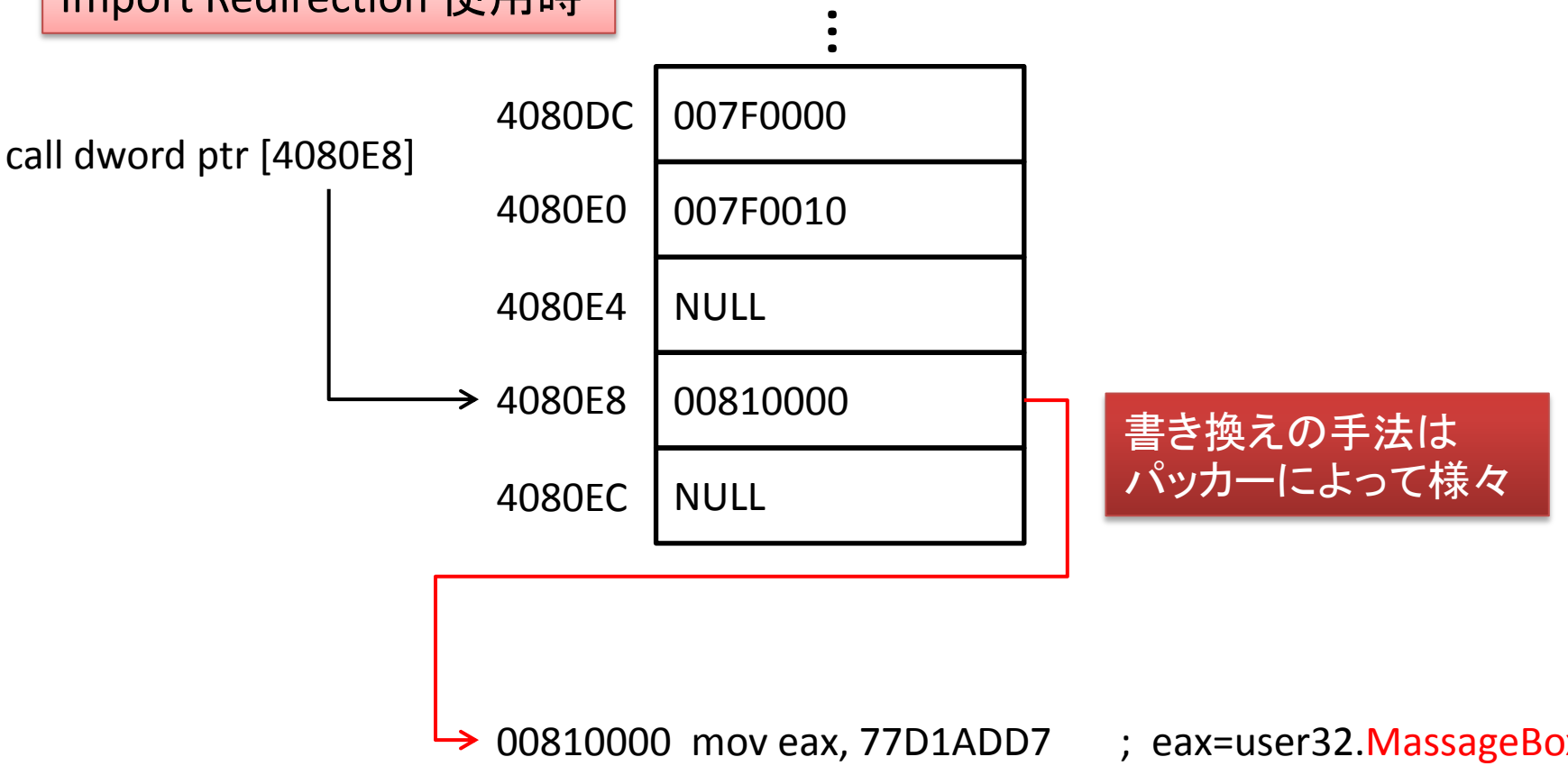


Import Redirection

Import Redirection

IATの値をヒープに作成したダミーコードのアドレスに書き換える手法

Import Redirection 使用時



Import Redirectionによる影響

元EXEのオリジナルコード

```
⋮
PUSH DWORD PTR DS:[1014D6C]
CALL DWORD PTR DS:[<&USER32.MessageBoxW>]
JMP calc.01005A73
PUSH 1
CALL DWORD PTR DS:[<&USER32.GetClipboard>]
MOV ESI,EAX
CMP ESI,EBX
MOV DWORD PTR SS:[EBP-C],ESI
JE calc.010059CA
PUSH ESI
CALL DWORD PTR DS:[<&KERNEL32.GlobalLock>]
⋮
```

多段パックから取得できたコード

```
⋮
PUSH DWORD PTR DS:[1014D6C]
CALL DWORD PTR DS:[10011A8]
JMP calc.01005A73
PUSH 1
CALL DWORD PTR DS:[10010EC]
MOV ESI,EAX
CMP ESI,EBX
MOV DWORD PTR SS:[EBP-C],ESI
JE calc.010059CA
PUSH ESI
CALL DWORD PTR DS:[1001088]
⋮
```

未知の多段パックに対しては，IATの再構築は行えず
API名は取得できない

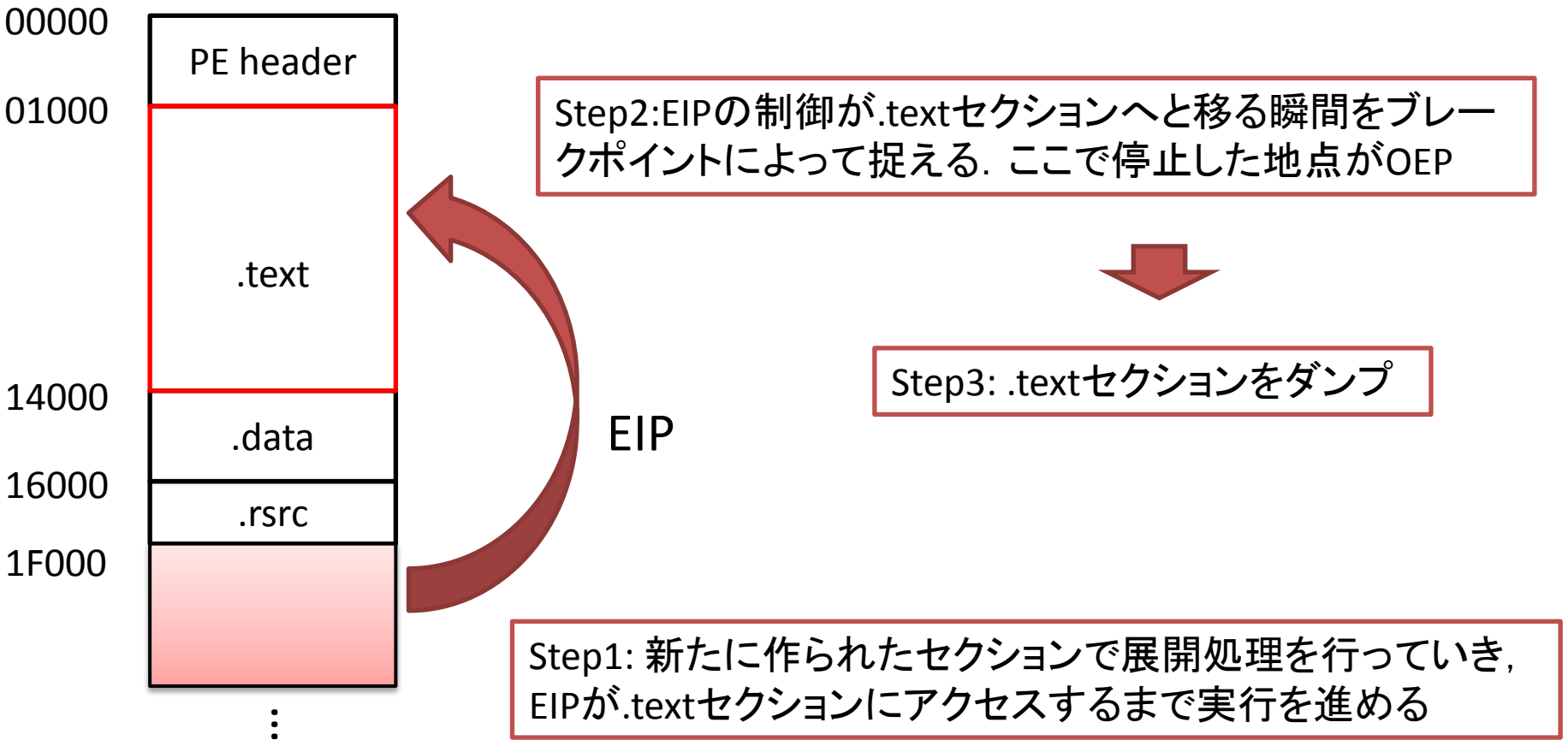
コード取得率はアドレス中のオペコードで比較して評価する

発表の流れ

1. パッキングの構造
   - 通常パッキング
   - 多段パッキング
2. 多段パッキングに対するオリジナルコード取得
   - オリジナルコード取得法
   - 評価実験1
3. コードを用いた類似度判定
   - 類似度判定法，類似度可視化
   - 評価実験2
4. まとめ

オリジナルコードの取得

EIP（インストラクションポインタ）：現在処理を実行しているアドレスの値を格納

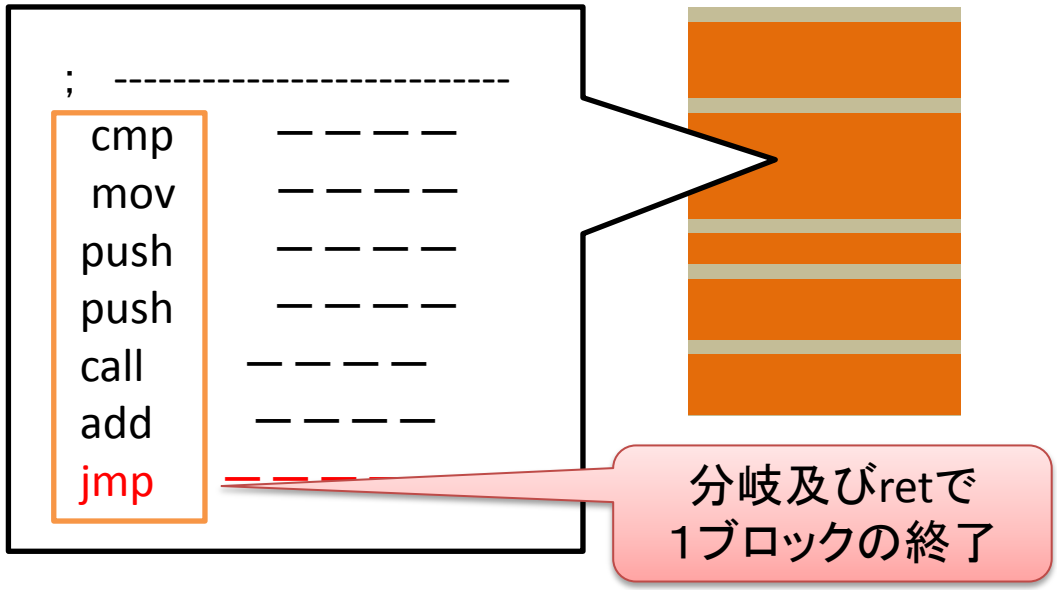


多段パック後EXEファイル

基本ブロック分割

基本ブロック： 分岐命令で区切られた連続した命令列.

基本ブロックの例



コード列

実行コード

```
push 0x0
lea eax, ptr [ebp-0x230]
push eax
push dword ptr [ebp-0x210]
call 0x7c94eb3b
mov edi, edi
push ebp
mov ebp, esp
push ebx
xor ebx, ebx
cmp byte ptr [0x7c9be0a0], bl
push edi
mov edi, dword ptr [ebp+0xc]
jnz 0x7c97d95d
movzx eax, word ptr [edi]
lea eax, ptr [eax+eax*1+0x2]
cmp eax, 0xffff
jnbe 0x7c97d968
cmp byte ptr [ebp+0x10], bl
push esi
mov esi, dword ptr [ebp+0x8]
lea ecx, ptr [eax-0x2]
mov word ptr [esi], cx
jnz 0x7c952ef9
```

分岐命令の例

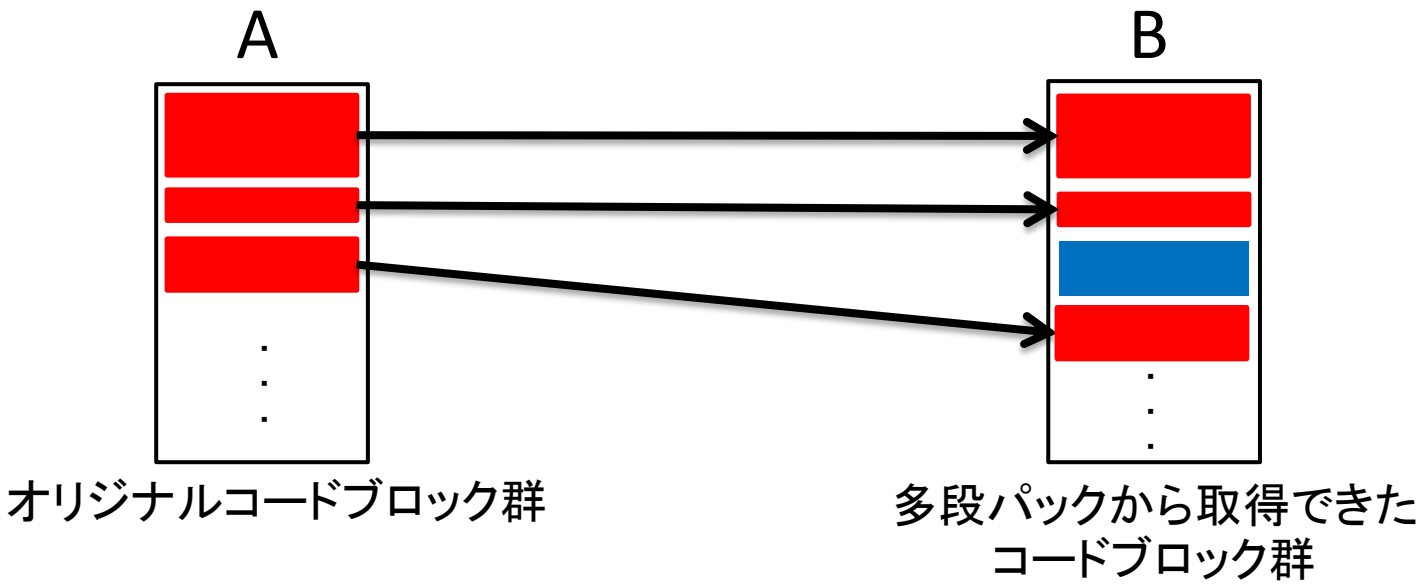
jz	jump if zero
jne	jump if not zero
jl	jump if less
jge	jump if greater or epual
. . .	. . .

オリジナルコード取得率の算出

◆オリジナルコードの取得率は基本ブロックの一致率によって定義する.

ブロック一致率: $sim(A;B)$

$$sim(A,B) = \frac{\text{Bとのブロック一致数}}{\text{マルウェアA総ブロック数}}$$



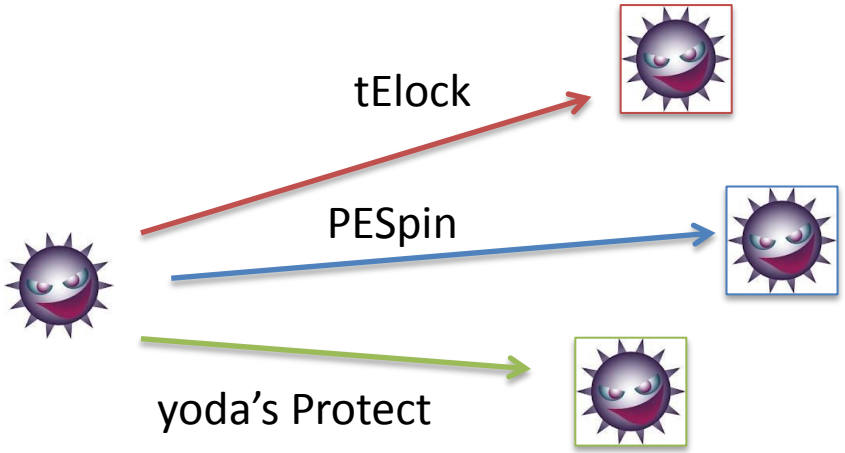
実験１:オリジナルコード取得率の算出

◆実験１:

１．168検体を用意して，それぞれ3種類の多段パッカー（tElock,PESpin,yoda’s Protect）でパッキングを行う（計504パターン）.

検体実行，提案手法によるコード取得

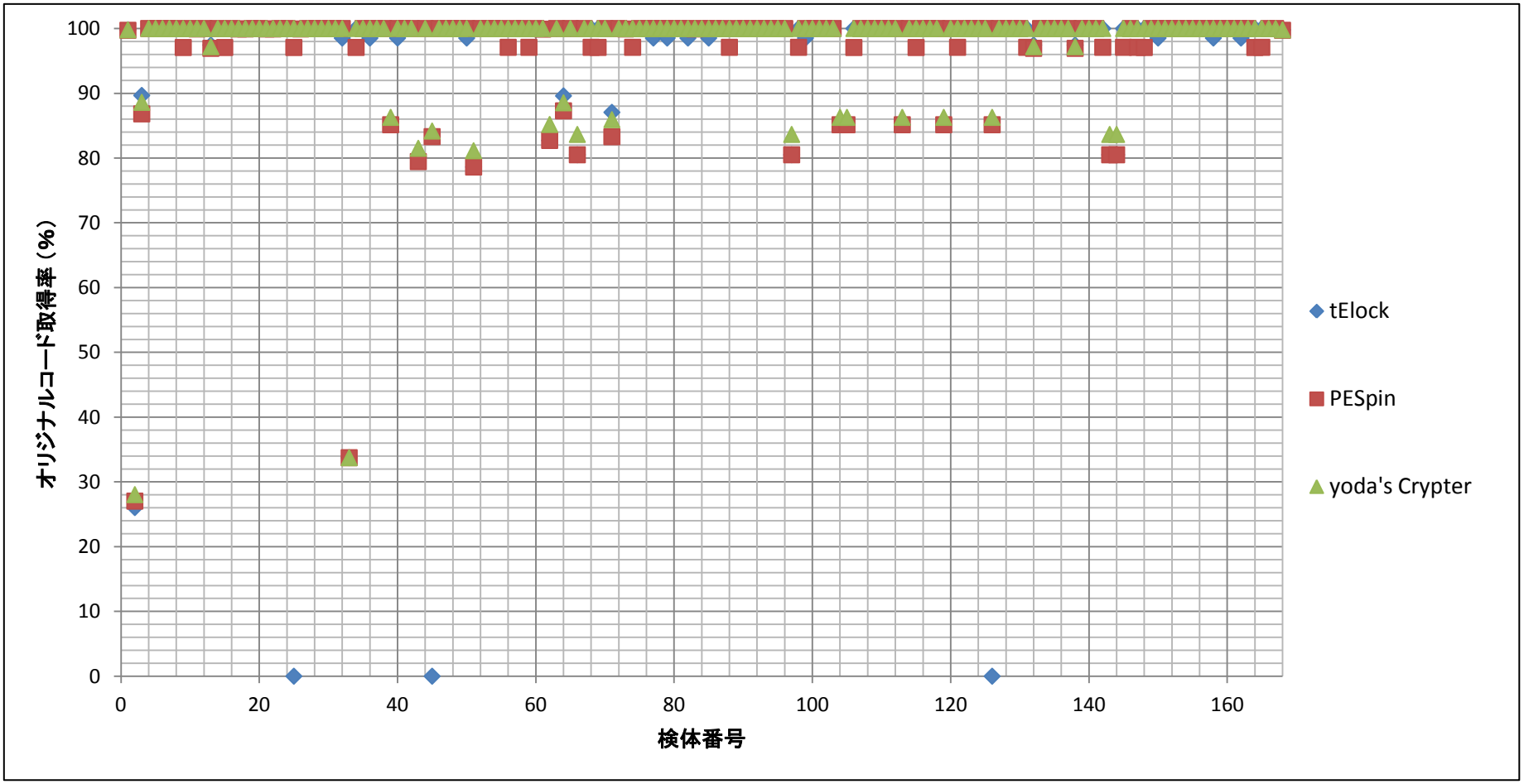
２．元検体と基本ブロック単位の比較を行うことで，オリジナルコード取得率を算出する.



実験結果

実験結果1：オリジナルコード取得率

ほとんどの組み合わせで，パック以前のオリジナルコードブロックを取得することができた．



発表の流れ

1. パッキングの構造
   - 通常パッキング
   - 多段パッキング
2. 多段パッキングに対するオリジナルコード取得
   - オリジナルコード取得法
   - 評価実験1
3. コードを用いた類似度判定
   - 類似度判定法，類似度可視化
   - 評価実験2
4. まとめ

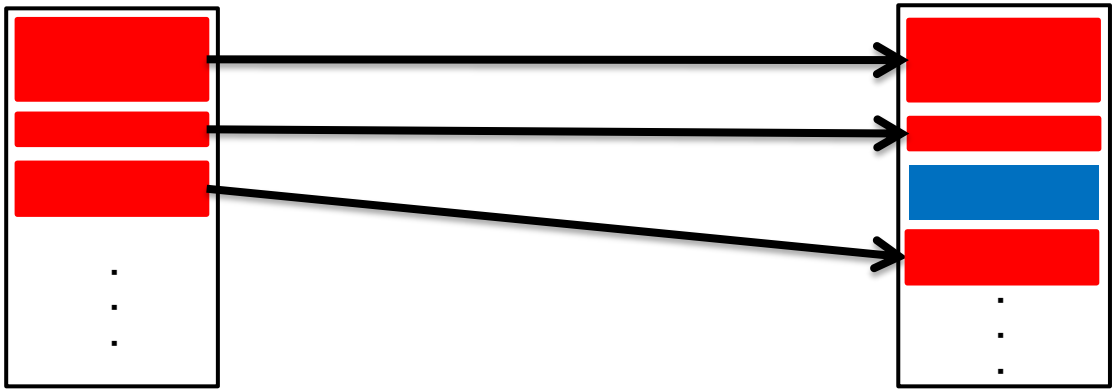
類似度判定

◆類似度判定も基本ブロックの一致率によって定義する.

◆ $sim(A, B)$ が高い ⇒ AがBの機能を多く有している.

ブロック一致率: $sim(A; B)$

$$sim(A, B) = \frac{\text{Bとのブロック一致数}}{\text{マルウェアA総ブロック数}}$$



マルウェアAのコードブロック群　　マルウェアBのコードブロック群

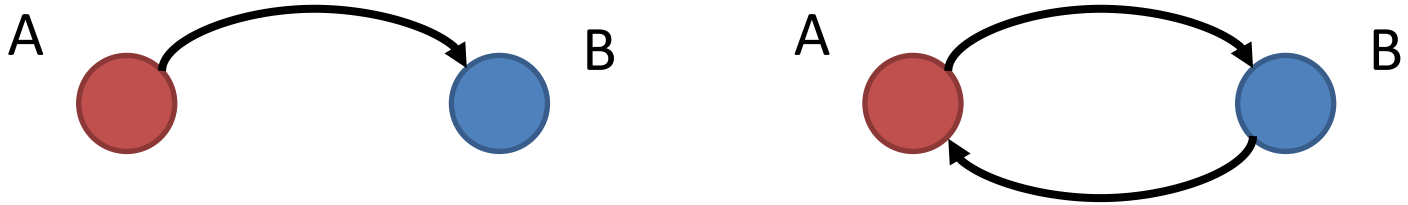
類似度可視化

◆総当たりで類似度を算出すると関係が複雑 ➡ グラフによる可視化

検体Aから検体Bへエッジ(辺)をつなぐ条件：

$$sim(A,B) = \frac{\text{Bとのブロック一致数}}{\text{マルウェアA総ブロック数}} \geq Z\ (\text{しきい値})$$

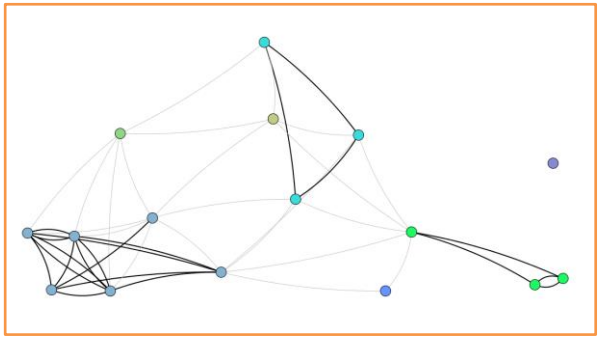
※BからAに対しても条件が成り立てば双方向からエッジをつなぐ



◆類似度可視化に用いた手法

Fruchterman-Reingold(FR)法：
エッジが密なほど検体間の距離が近い

Newmanアルゴリズム：
関連度の低いエッジから削除してクラスタリング



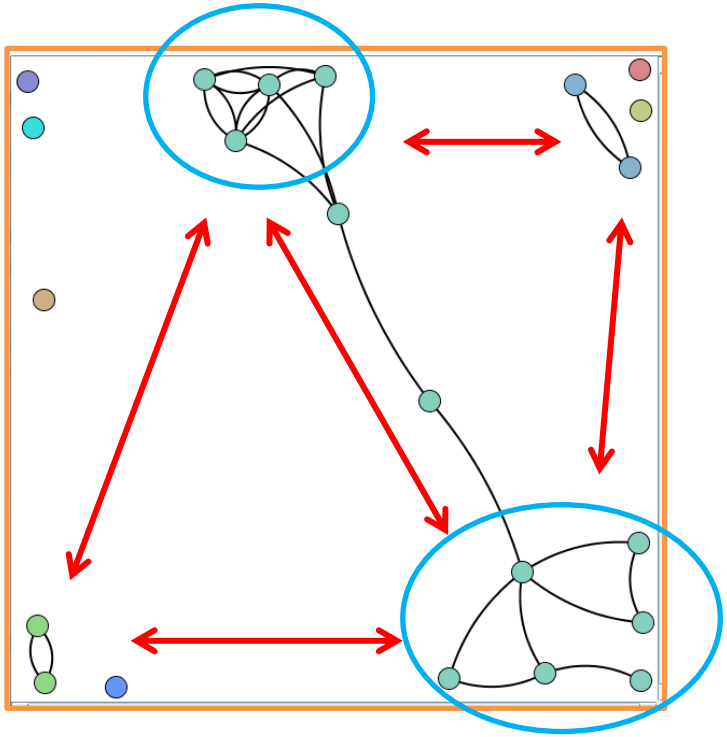
FruchtermanReingold（FR）法

➢FR法(概要)：

✓リンクの強さを電磁力に見立てて，距離で類似度を示したマップ．

✓リンクを持つもの同士には引力が，リンクを持たないもの同士には弱い斥力が働く．

✓リンクの強さは距離の２乗に反比例する．



Newman Algorithmによるクラスタリング

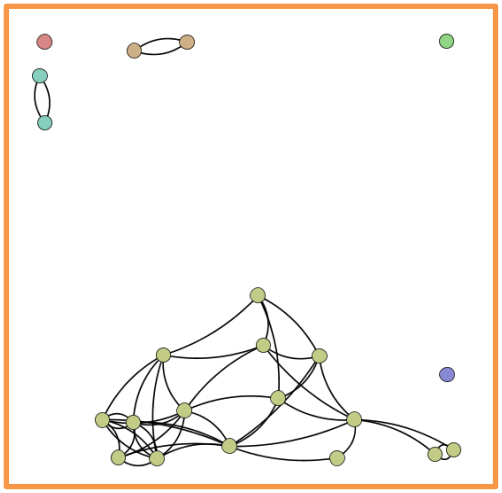
➢Newman Algorithm(概要):

✓トップダウンアプローチでのクラスタリング手法.

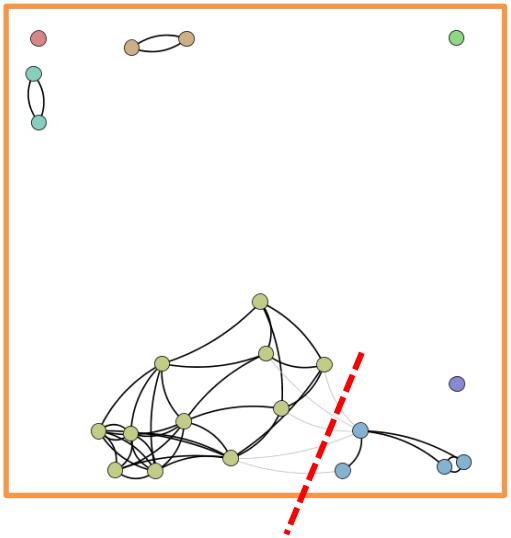
✓コミュニティ間を跨ぐ存在になる可能性の高いエッジから順に切り離す.

任意の2ノード間の最短パスに最も多く含まれるリンク

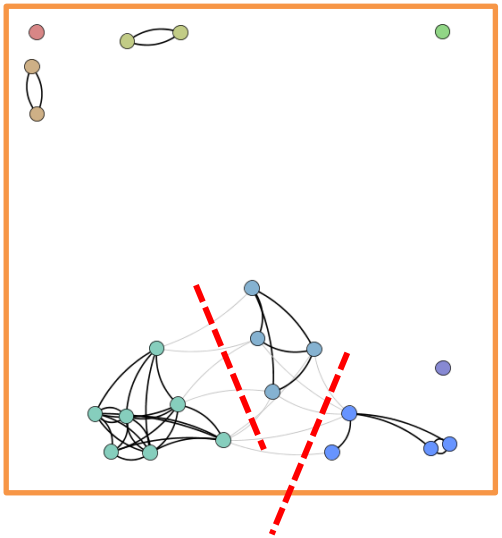
ステップ1



ステップ2



ステップ3



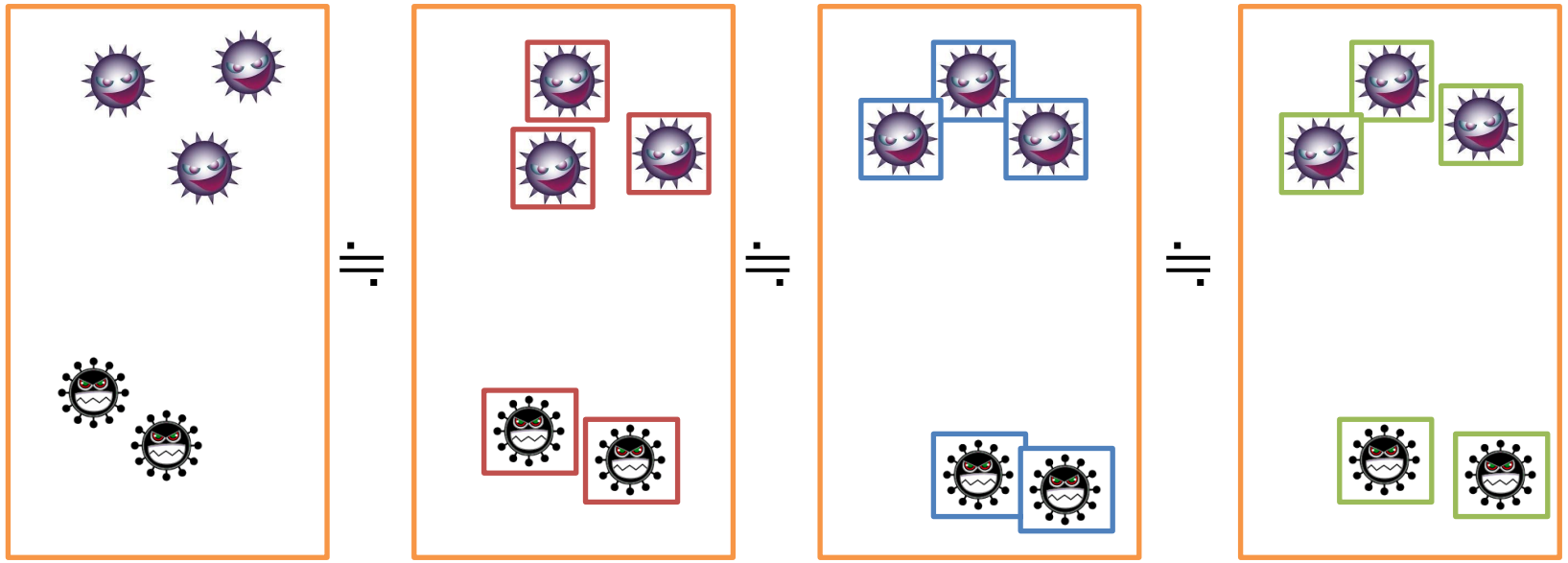
実験2：類似度判定・可視化

◆目的

- コードの本質的な部分が取得できているかどうかを評価.
- 多段パックの種類によらず，同じような類似度判定が行えるかどうかを判定.

◆使用検体は，実験1で用いた168検体(パック無し+3種の多段パッカー)

(1) パック無し　(2) tElock　(3) PESpin　(4) yoda's Protect

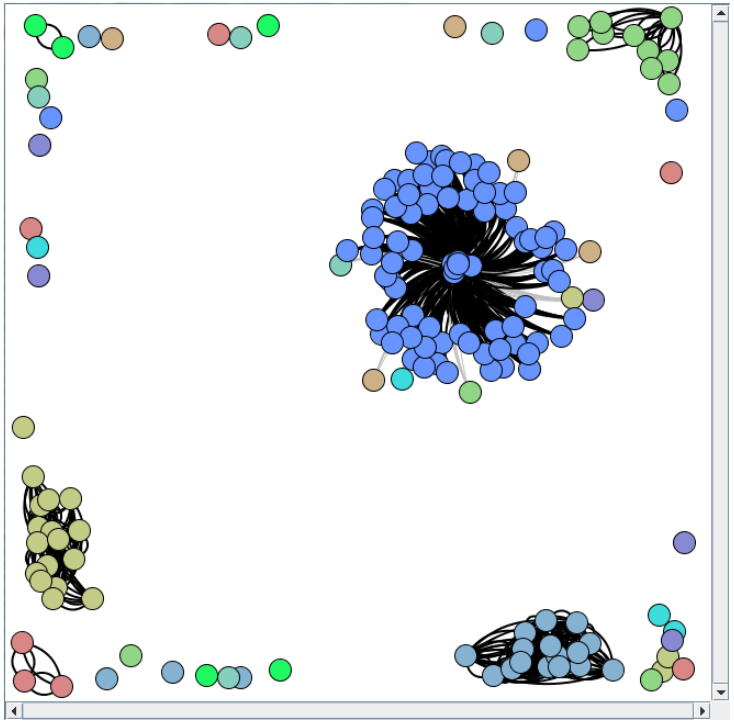


実験結果2：類似度判定

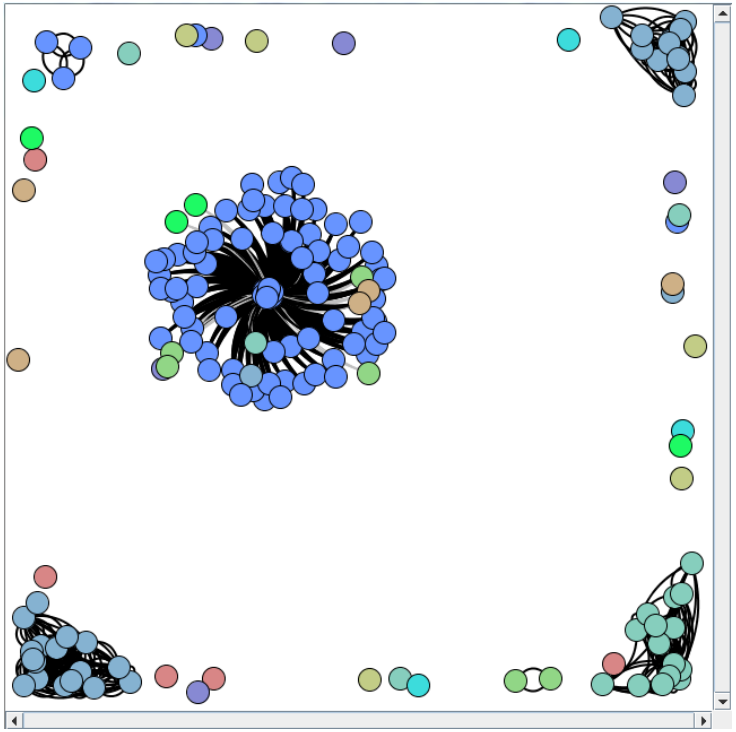
◆168検体に対して総当たりで類似度判定を行った.

◆しきい値 Z= 90(%)

(1) パック無し168検体



(2) tElockでパックした168検体



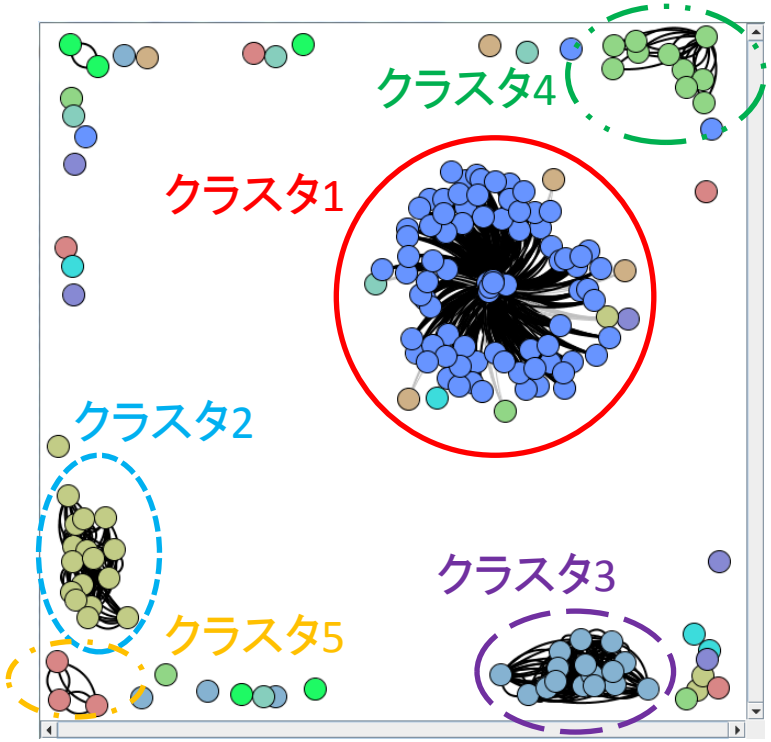
PESpin，yoda’s Protectの類似度判定結果

実験結果2：類似度判定

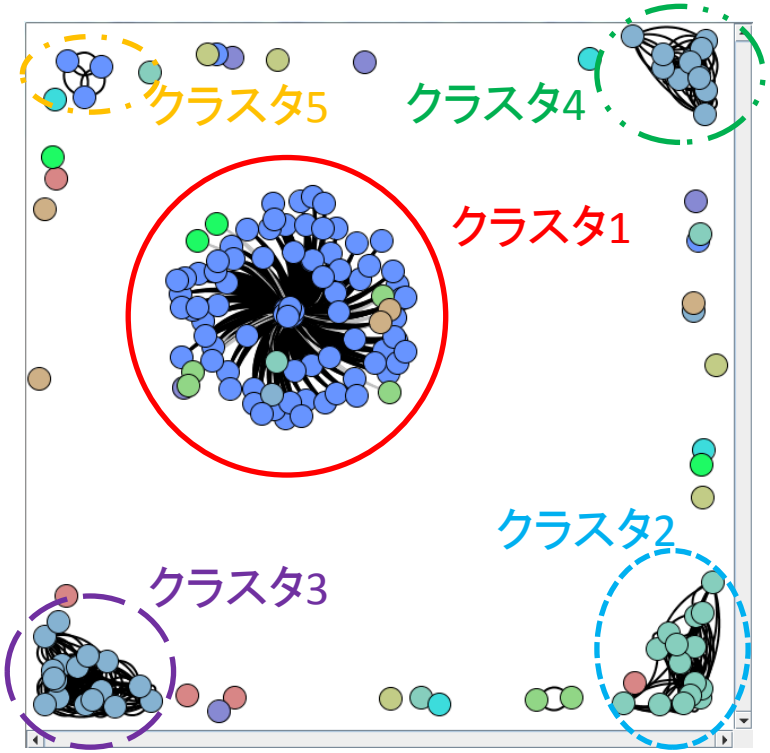
◆168検体に対して総当たりで類似度判定を行った.

◆しきい値 Z= 90(%)

(1) パック無し168検体



(2) tElockでパックした168検体

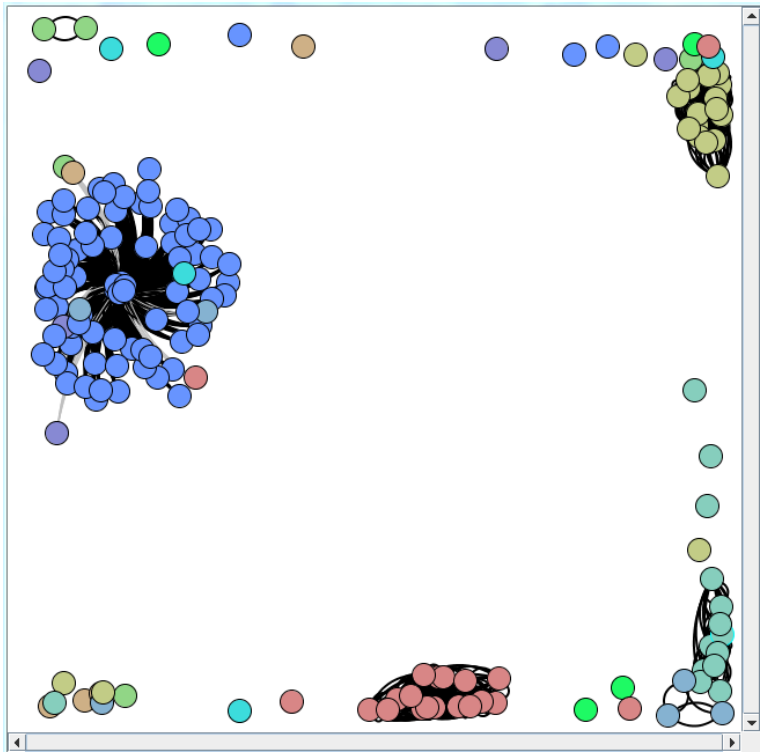


PESpin，yoda’s Protectの類似度判定結果

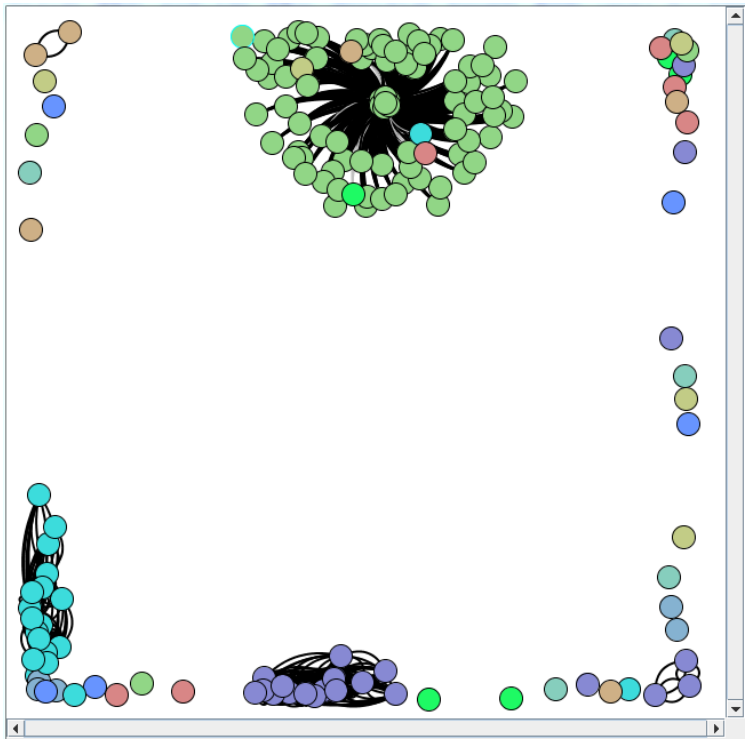
実験結果2：類似度判定

- PESpin，yoda’s Protectでパックした検体もパック前と同様の類似度判定を行うことができた.

(3) PESpinでパックした168検体



(4) yoda’s Protectでパックした168検体



まとめ

◆多段パックされたマルウェアからIATを再構築していない状態のオリジナルコードを取得する手法を提案した.

◆168検体を3種類の多段パッカーでパッキングを行い，ほぼすべての組み合わせでオリジナルコードを取得することができた.

◆取得したオリジナルコードのオペコード部分をマルウェア間で比較することで類似度判定を行うことができた.

◆可視化して，マルウェア間の類似度関係をより分かりやすくした.