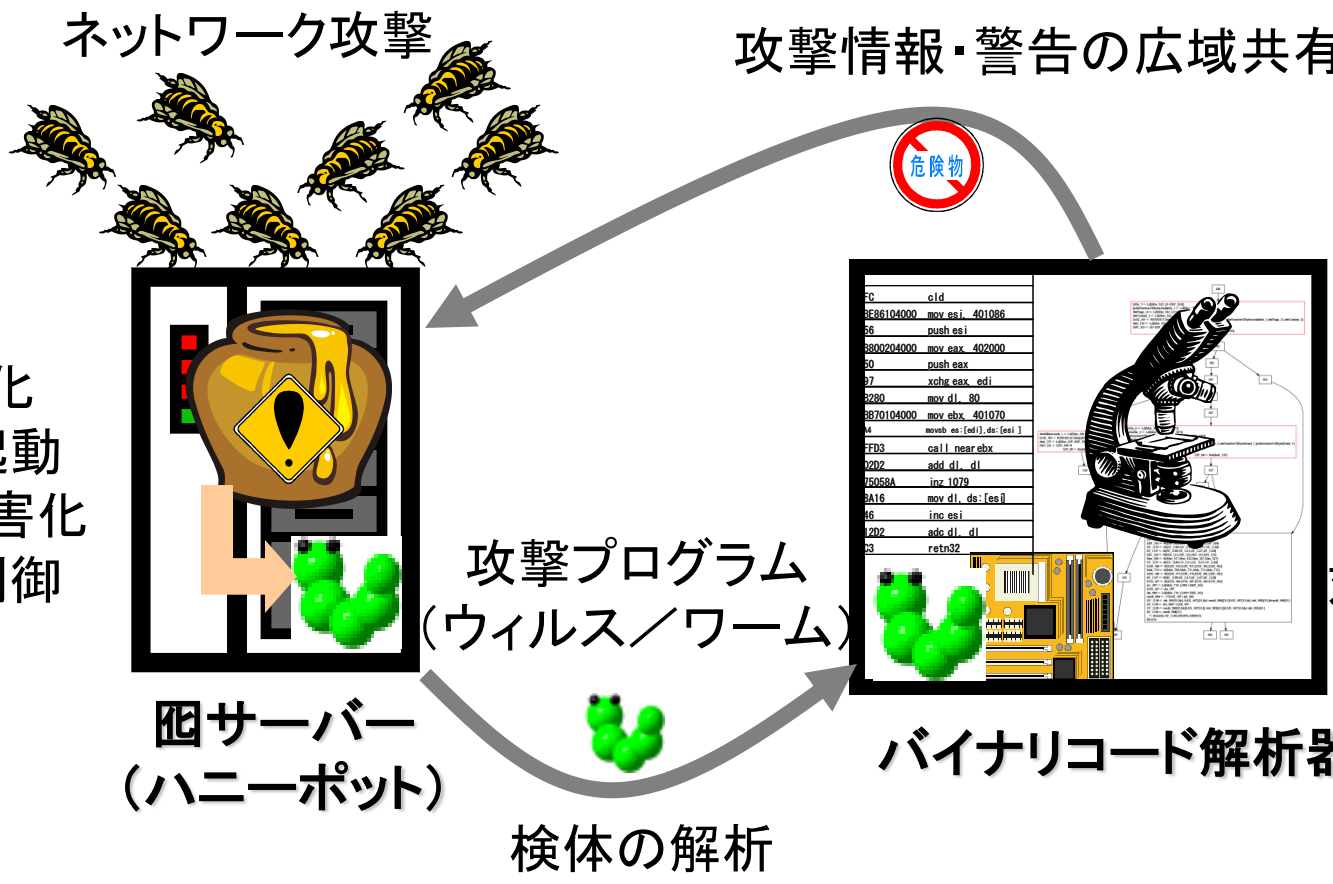
ハニーポットとバイナリコード解析の連携による ネットワーク攻撃の自動防御技術に関する研究

独立行政法人 産業技術総合研究所
国立大学法人　名古屋大学

研究の背景と目的

- ネットワーク攻撃の大規模化と先鋭化
  - 標的型攻撃：政府機関、防衛産業、原子力施設
  - ボットネット攻撃：クレジットカード情報等流出
- 既存の対処法の限界
  - 未知攻撃への対処が困難（ゼロデイ攻撃）
    - 既知攻撃パターン（シグネチャ）に依存
  - 攻撃捕捉網の不足
    - いつどのように攻撃を受けたかすら不明な場合が多い
- 未知のネットワーク攻撃を発生端緒で同定し自動的に分析を行う手法の確立

目標：ネットワーク攻撃自動防御



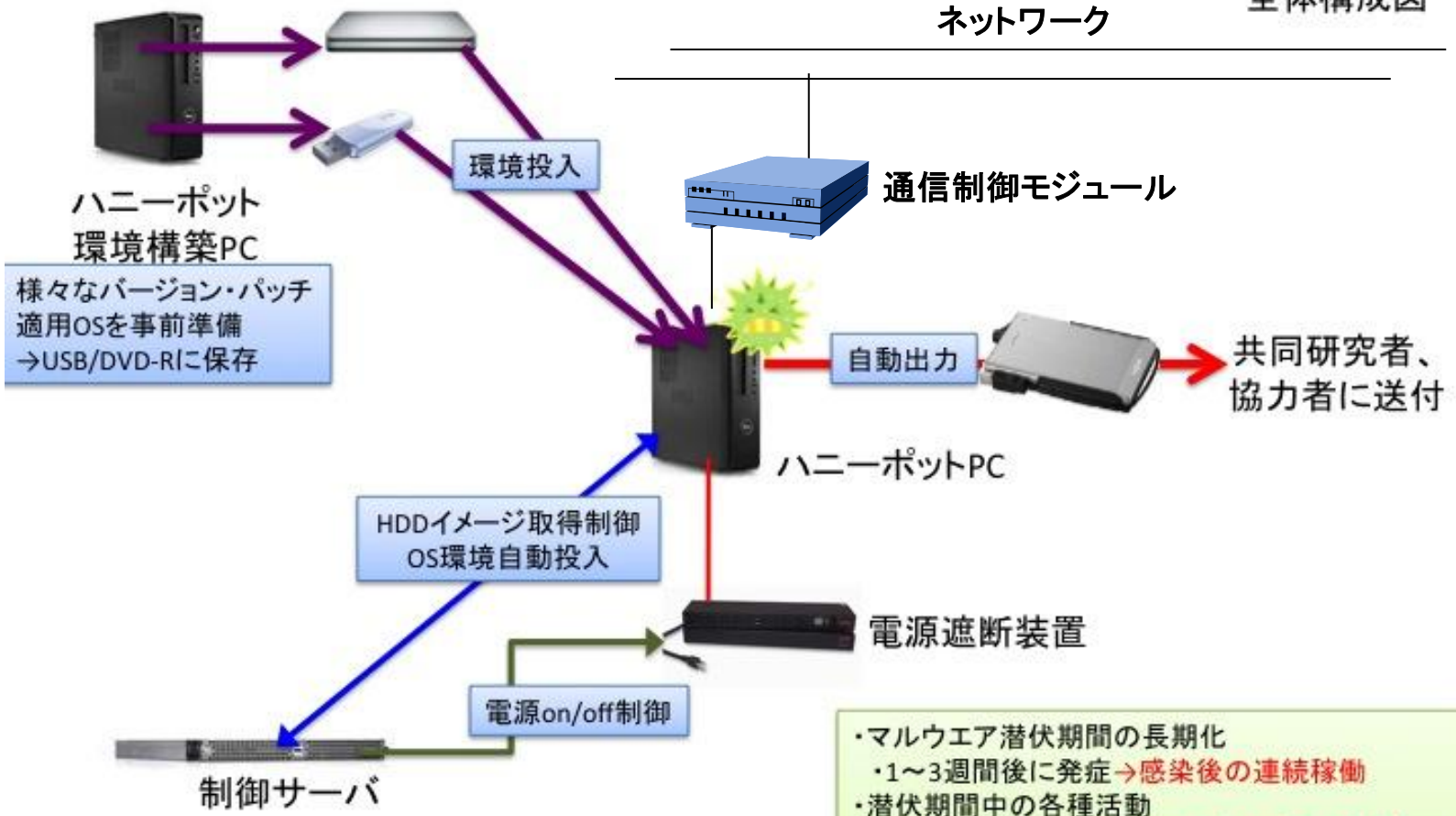
困難なネットワーク攻撃の捕捉と分析を自動化

ハニーポット技術の研究

- 高感度で運用の容易なハニーポット技術
  - 実ハードウェアに直接OSを搭載
  - 攻撃パケットを遮断し実ネットワークに接続
  - 制御サーバーによりOS起動イメージの投入や電源制御を自動化
  - 未知の攻撃パケットを同定
  - 投下された攻撃プログラム(マルウェア)を自動回収

開発ハニーポットシステム

全体構成図



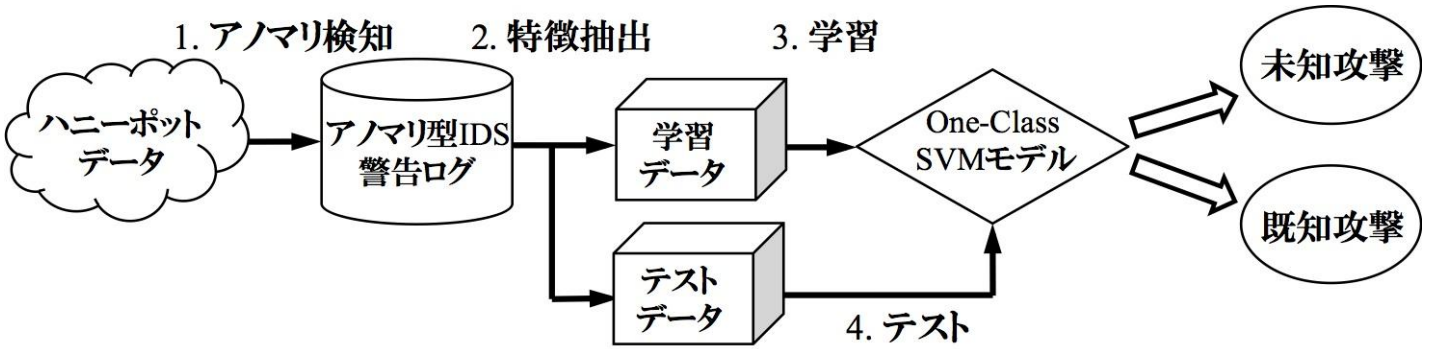
・マルウエア潜伏期間の長期化
　・1～3週間後に発症→感染後の連続稼働
・潜伏期間中の各種活動
　・MS社等への到達性確認→実環境への接続
　・攻撃との識別必要→制御サーバ構築
・マルウェア自己消滅対策
　・シャットダウン検知→一定周期で電源断
　(問題)HDDおよび電源ユニットに過大の負荷

その他開発ハニーポットシステム

- 迷惑メールハニーポット
  - 迷惑メール本文中のリンクを選択的に巡回
  - 毎分80件のメールを処理可能
  - 30分平均で毎分10 件の迷惑メールを受信し、合計1,968,593個のURLを抽出し20,153 件のマルウェアファイル(疑われるものも含む)を収集
- Linuxハニーポット
  - Linux OSを標的としたネットワーク攻撃を捕捉
- IPv6ハニーポット
  - statelessアドレスへのスキャンや攻撃を検知・分析し、別サブネットのハニーポットが攻撃されるよう制御

攻撃データ特徴抽出による未知攻撃同定

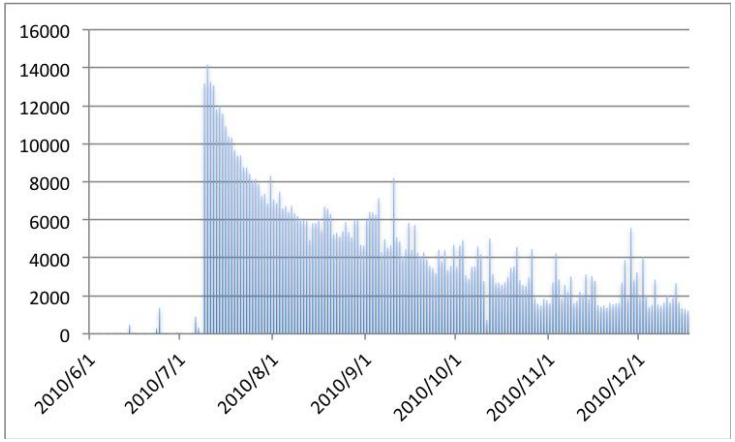
- ハニーポット観測データよりIDSやアンチウィルス、攻撃コード検知システムの情報を抽出する。
- 過去の攻撃データ学習データとし、調査対象の攻撃データをテストデータに分類する.
- 学習データを用いてOne-Class SVMのモデル生成



- 未知判定の正解率の向上とfalse positiveの増加

未知ネットワーク攻撃捕捉例

- 平成21年度
  - 韓国および米国に対するDDoS攻撃（7月4日～7月10）
  - Oracleに対する攻撃（7月8日）
  - Windows Vistaに対するゼロデイ攻撃（10月10日）
  - Windows XPに対するゼロデイ攻撃（11月10日）
- 平成22年度
  - SIPサーバ攻撃（7月9日）
- 平成23年度
  - Windowsポート{135,445,1433}/tcp攻撃（9月9日）
  - JBossの脆弱性攻撃（10月19日）
  - 韓国から23/tcpログインを試みる攻撃（12月5日～）
  - Windows脆弱性PoC（3月16日公表）ベースの攻撃（3月17日）



マルウェアバイナリコード解析の研究

- マルウェアバイナリコードの自動分析
  - 動的エミュレーション（暗号化、難読化）と静的コード解析（マルチパス解析）を連動
  - 静的単一代入（SSA）形式による間接ジャンプ命令の飛び先アドレス計算
  - プログラム全体（サブルーチン等含む）解析
  - ハードウェア仮想化を利用した高精度な解析
  - API関数呼び出しを含む振舞いの網羅解析

機械命令を代入文の列へ変換

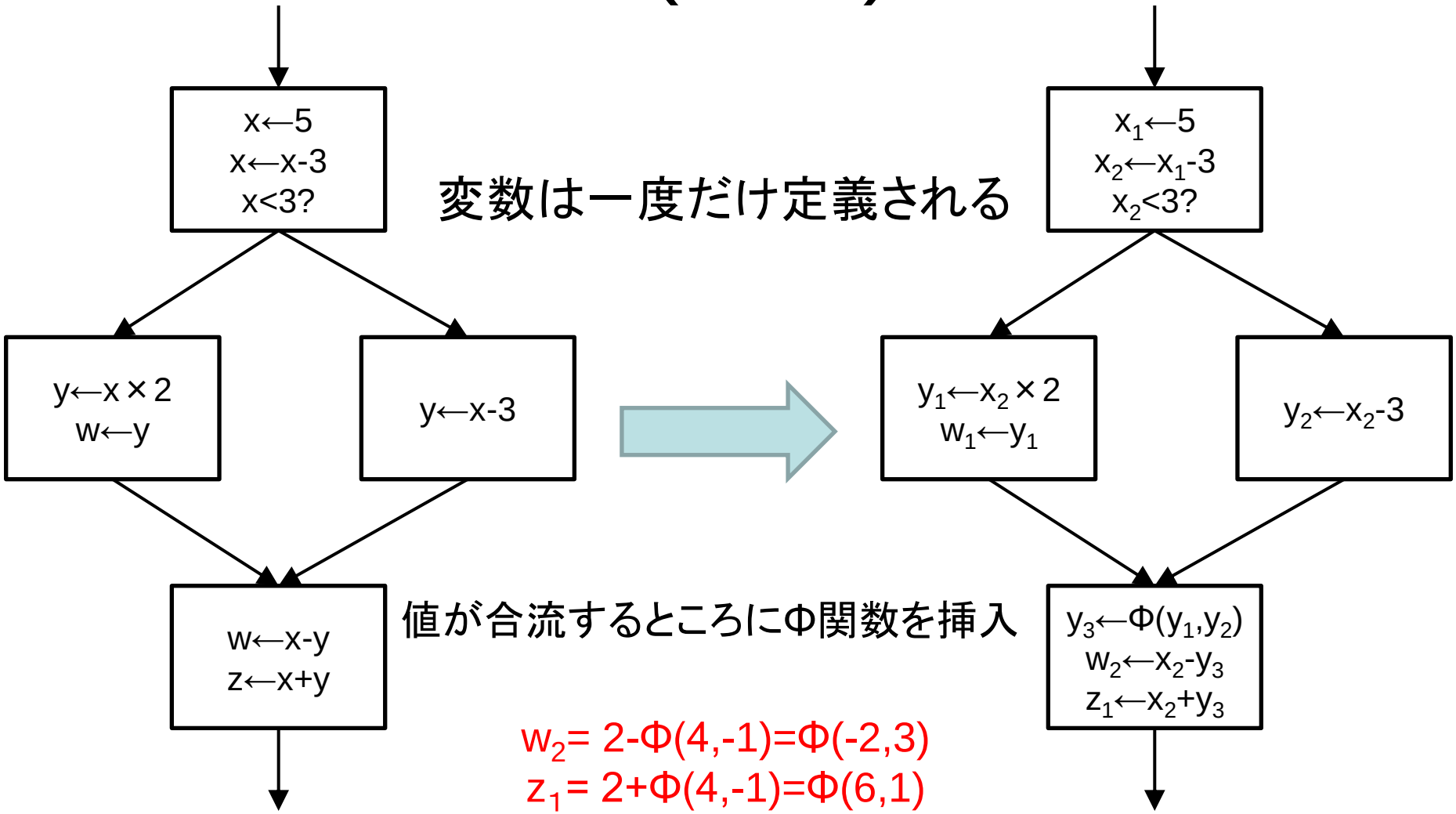
機械命令	代入文への変換
ADD EAX, EBX	$result_{32}$←EAX+ EBX, CF←EAX > 0xFFFFFFFF - EBX, SF←$result_{32}$[31], ZF←$result_{32}$==$0x0_{32}$, EAX←$result_{32}$
JZ 0x4000	Branch(ZF_1, 0x4000, cur + ilen)
CALL EBX	dest←EBX ESP←ESP−4 M←St_{32}(M,ESP, next) ←Call(dest)
RET	dest←Ld_{32}(M, ESP) ESP←ESP+4 ←Ret(dest)

制御フラグ更新

メモリ読み出し（配列要素）

メモリ書き込み（配列の更新）

静的単一代入(SSA) 形式へ変換



$w_2 = 2-\Phi(4,-1)=\Phi(-2,3)$

$z_1 = 2+\Phi(4,-1)=\Phi(6,1)$

$w_2+z_1 = \Phi(-2,3)+\Phi(6,1)=\Phi(-2+6,3+1)=\Phi(4,4)=4$

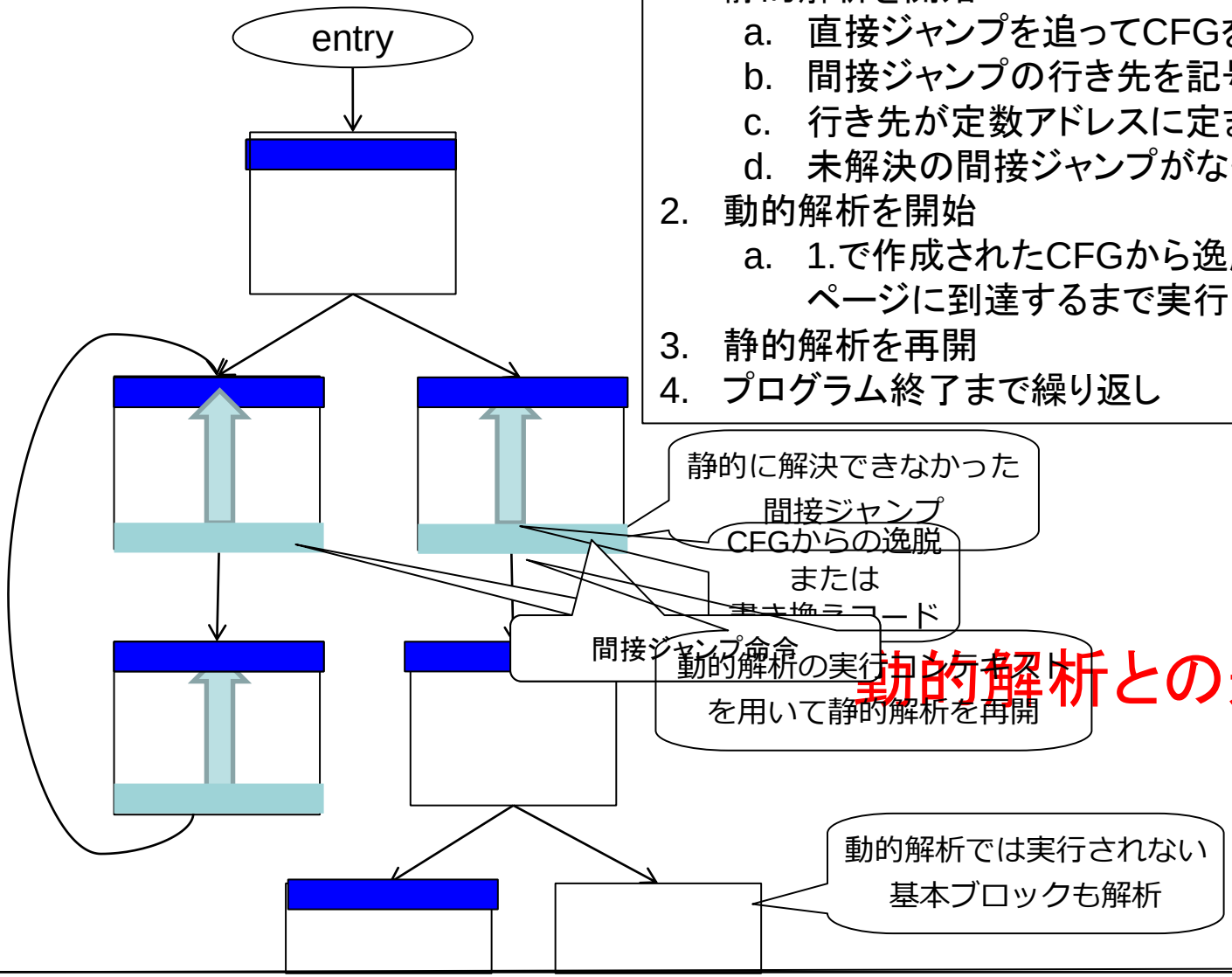
http://en.wikipedia.org/wiki/Static_single_assignment_form

間接ジャンプ命令の飛び先を記号計算

- 変数のuse-def鎖を辿って定数値へ帰着
  - 要求駆動型定数伝播
- メモリは巨大な配列として扱う（メモリ変数）
  - Ld(St(M,A,D),A) = A
  - Ld(St(M,A,D),A') = Ld(M,addr') if A ≠ A'
- Φ関数の代数則（分配則など）を利用
- 全プログラム解析
  - サブプロシジャのような文脈依存コードの解析
    - CALL/RET命令は信用できない
  - Φ関数で表現された飛び先アドレス
- 動的解析との連携

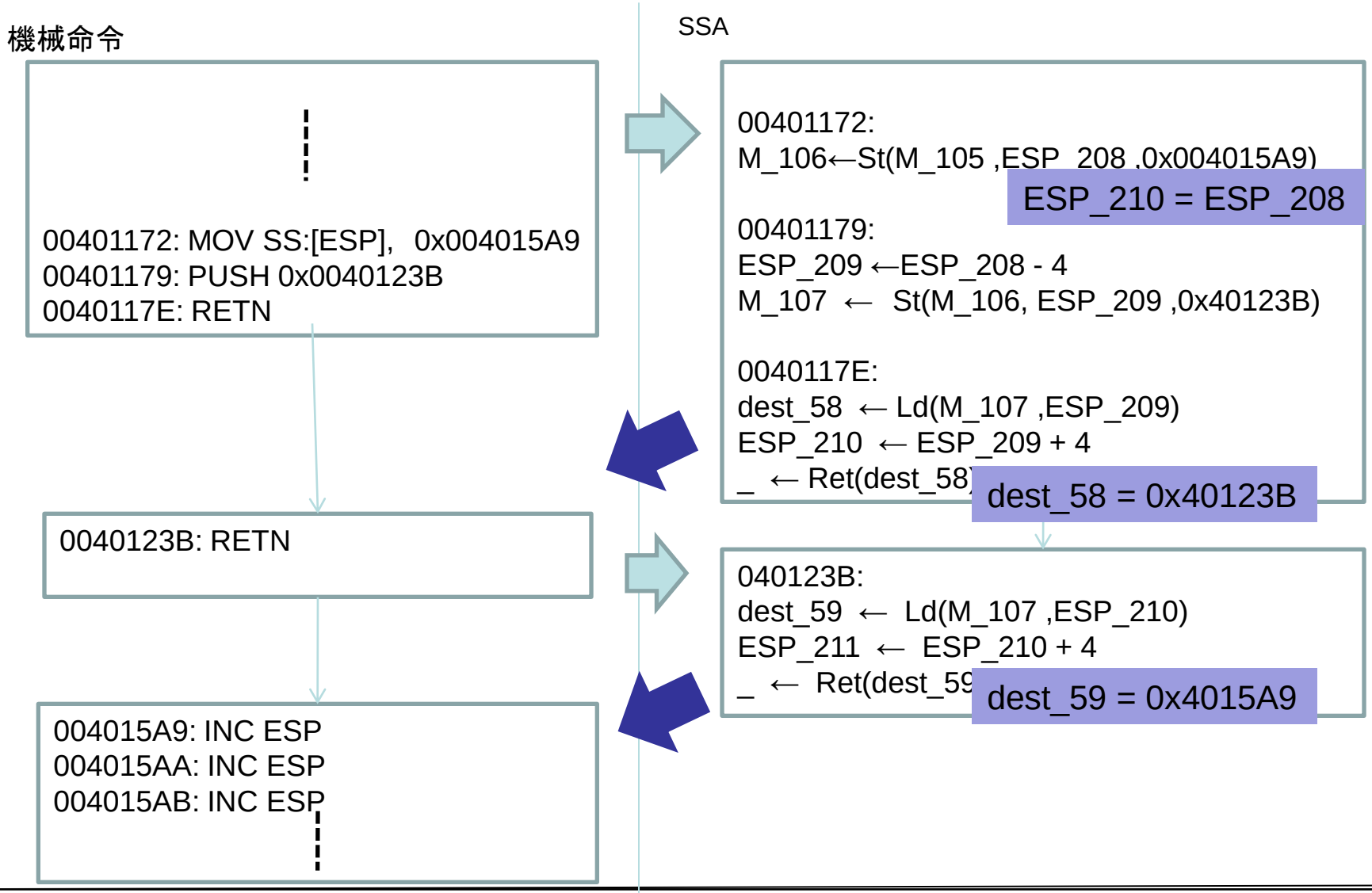
展開型静的解析

1. 静的解析を開始
   a. 直接ジャンプを追ってCFGを構築(再帰探索)
   b. 間接ジャンプの行き先を記号評価を用いて解決
   c. 行き先が定数アドレスに定まった場合グラフを拡張
   d. 未解決の間接ジャンプがなくなるまで繰り返す
2. 動的解析を開始
   a. 1.で作成されたCFGから逸脱するか書き換えられたページに到達するまで実行
3. 静的解析を再開
4. プログラム終了まで繰り返し

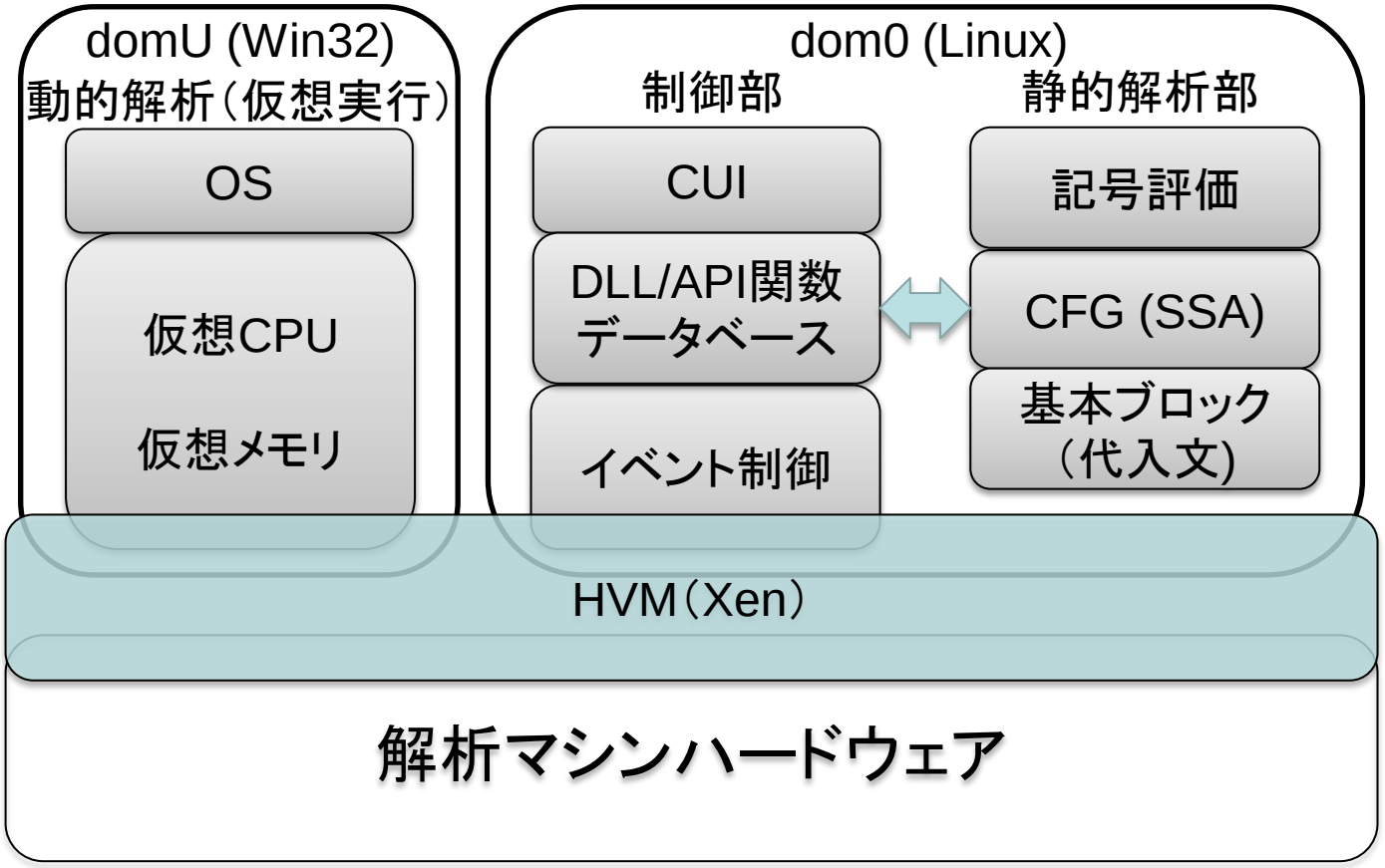


動的解析との連携を実現

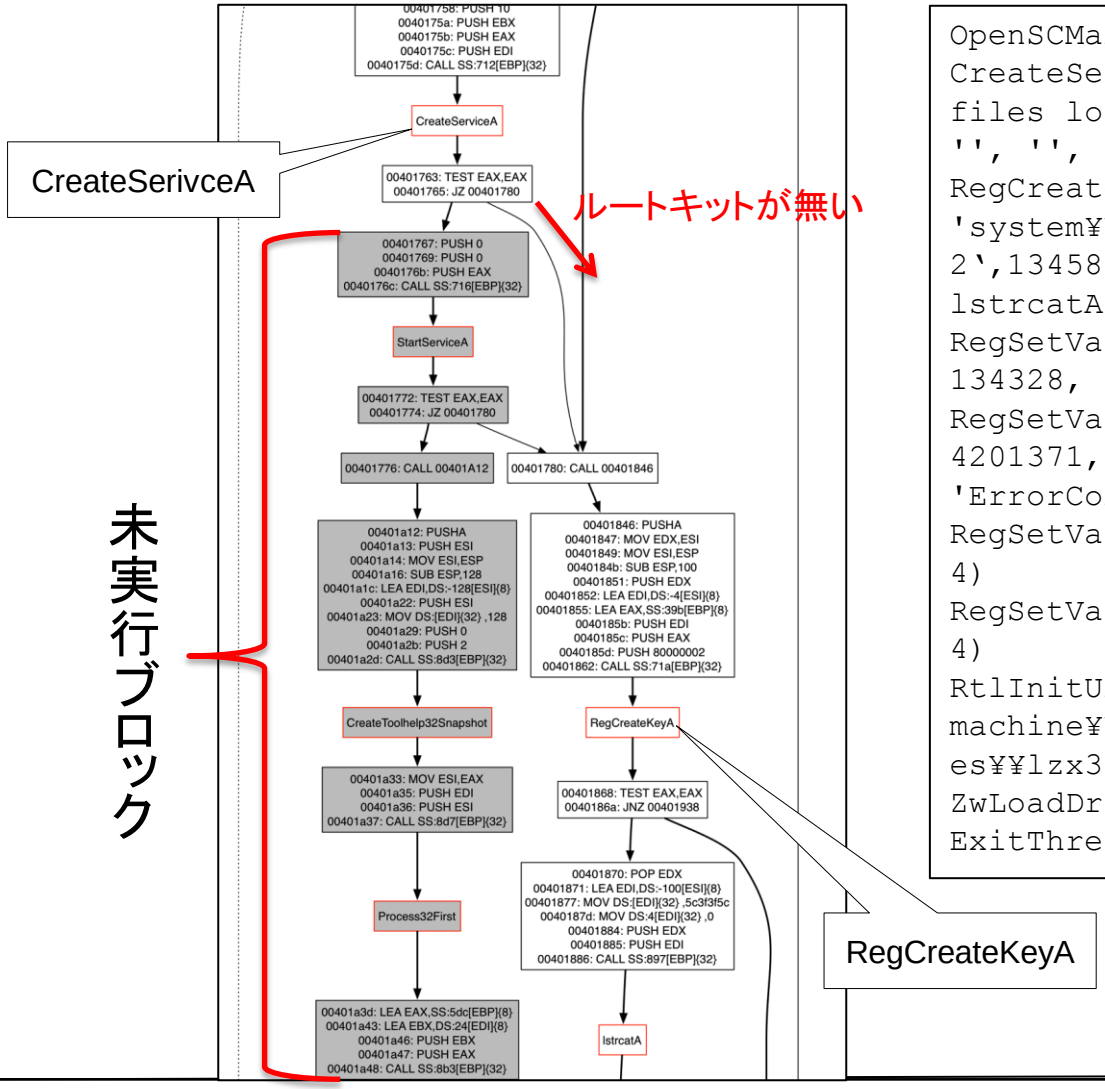
難読化解析の例: Rustok.B



Xenハイパーバイザー上での実装



ボットネットワームのドロッパー解析例 (Win32/Rustock)



動的解析のトレース

```
OpenSCManagerA('', '', 983103)
CreateServiceA(0, 'pe386', 'Win23 lzx
files loader', 16, 1, 1, 0, '`','Base',  0,
'', '', '`)
RegCreateKeyA(2147483650L,
'system¥¥CurrentControlSet¥¥Services¥¥lzx3
2`,134580)
lstrcatA('¥¥??¥¥', '`)
RegSetValueExA(16, 'ImagePath', 0, 1,
134328, 4)
RegSetValueExA(16, 'DisplayName', 0, 1,
4201371, 21) RegSetValueExA(16,
'ErrorControl', 0, 4, 4200764, 4)
RegSetValueExA(16, 'Start', 0, 4, 4200768,
4)
RegSetValueExA(16, 'Type', 0, 4, 4200772,
4)
RtlInitUnicodeString(134328,u'¥¥registry¥¥
machine¥¥system¥¥CurrentControlSet¥¥Servic
es¥¥lzx32`)
ZwLoadDriver(134328,)
ExitThread(0,)
```

現状と今後の課題

- 基幹ネットワークでのハニーポット自動運用により、ネットワーク攻撃の捕捉性能が向上
- ハードウェア仮想化エミュレーションと静的バイナリーコード解析の連携により、マルウェアコード解析の性能が向上
  - 実際に蔓延した高度なボットネットワームやルートキット(ドロッパー)を自動的に解析可能
- これら技術の連携の有効性を確認
  - 亜種の同定、接続ホストや分岐情報等の二次攻撃情報の提供、など
- バイナリーコード解析の処理効率改善が必要(数十分から数十時間)
    - インクリメンタル再計算、並列処理、記号計算ツール