卒論ミーティング (10/21)

31010 佐藤秀明

概要

- Tamperproofing
  - あらまし
  - 卒論の題材 (候補)
- これからの予定

Tamperproofing とその周辺技術

- ソフトウェアが改変されたことを、そのソフトウェア自身が検知するシステム
  - 改変を監視する対象は一般にプログラムのコード部
  - ソフトウェアに埋め込まれた透かし情報の改変をチェックする場合もある (→Software watermarking)
  - 検知システムは攻撃を受けにくいものでなければならない (→Obfuscation)
    - その存在を攻撃者に気づかれにくい
    - 気づかれたとしてもそのシステムのみを安全に除去することが困難

既存技術の例 (1)

- コードの checksum やハッシュ値を比較 [1]
  - あらかじめ計算しておいた値と異なる場合、データが破壊される

```
start:
   [codes to be checked]
end:
   ...
   add %sp, -checksum
   mov start, %r1
for:
   cmp %r1, end
   bg next
   ld [%r1] , %r2
   add %sp, %r2, %sp
   add %r1, 4, %r1
   jmp for
next:
   ...
```

既存技術の例 (2)

- jump 先のアドレスをハッシュ関数に計算させる [2] [3]
  - 関数 f は $\{a_i \rightarrow i\}$ のハッシュ関数を経由して $\{i \rightarrow b_i\}$ のテーブルを引き、a_i に return せず直接 b_i に jump する
  - テーブルを修正せずにコードだけ改変すると、誤ったアドレスへ jump する

$$a_i: \quad \text{jmp } b_i$$
$$\downarrow$$
$$a_i: \quad \text{call f}$$

既存技術の問題

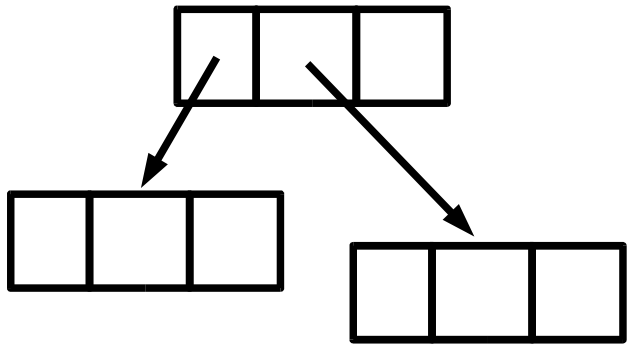
- データや制御を壊してしまうだけの機能
  - 壊れたことをチェックして異常終了させるかどうかを instruction が明示的に判定→攻撃者に気づかれる危険性
- 改変されたら勝手に終了して欲しい !!

やりたいと思っていること

- 監視するコードの内容を基にデータ構造を参照
- 改変されたコードを基にした操作はポインタの不正な参照 (NULL 参照など) を引き起こす

operate_structure(structure, codes);

↓　各ノード参照



利点

- 改変を検知すると速やかにSegmentation faultで落ちることが期待できる
  - 誤った動作を続けることが少ない
- 実行時のデータ構造を利用する
  - 攻撃者は動的な解析をしなければならない

解決すべき課題

- データ構造の占めるメモリの量を抑える
- valid なコードのみを受理するデータ構造を構成する方法
- データ構造と元プログラムとの密着性
  - プログラム中の定数の値をデータ構造に依存させることによって実現する予定 [4]

これからの予定

- 実装する環境を決定する
  - おそらく、Sparc のアセンブリに対する操作を想定
- 適切なデータ構造を設定する
- 逆コンパイルの手法を学ぶ

References(1)

- [1] H. Chang and M. Atallah. Protecting software code by guards. In Proc. 1st ACM Workshop on Digital Rights Management (DRM 2001), pages 160-175. Springer LNCS 2320, 2002.
- [2] C. Linn and S. Debray. Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), pages 290-299, October 2003.

References(2)

- [3] Christian Collberg, Edward Carter, Saumya Debray, Andrew Huntwork, Cullen Linn, and Mike Stepp. Dynamic path-based software watermarking. In SIGPLAN '04 Conference on Programming Language Design and Implementation, june 2004.
- [4] Yong He. Tamperproofing a software watermark by encoding constants. Master's thesis, Comp. Sci. Dept., Univ. of Auckland, 2002.