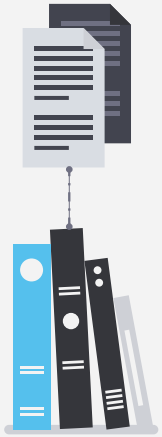
PackGenome: Automatically Generating Robust YARA Rules for Accurate Malware Packer Detection

Li, Shijia, et al. Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. 2023.

吳冠廷 112C53033

Table of contents



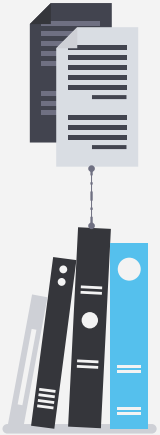
01

Introduction


殼
到處都是殼

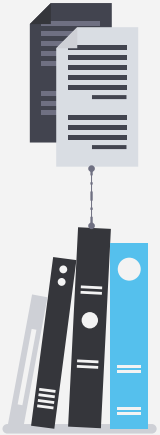
Research Motivation

基於現有檢測方法的侷限性和加殼技術的不斷演變，提出一種自動化、準確且高效的方式來偵測加殼器，減少分析人員的負擔，並提高檢測的有效性和可靠度。
通過自動生成基於加殼器基因的 YARA 規則，解決傳統手動方法的不足，並提供一個能夠快速適應和應對新威脅的檢測工具。



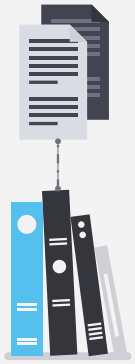
Research Objectives

1. 自動生成 YARA rule，減少人工編寫的工作量。
2. 提高檢測準確度，降低誤報和漏報。
3. 確保執行效率，不會因為大規模樣本造成資源消耗。
4. 提供通用解決方案，能應用在不同加殼技術和平台上。

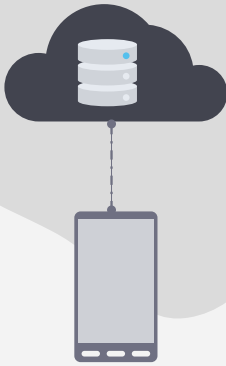


02

Background



Malware Packing Techniques



Packer

目的：壓縮
常見工具：UPX

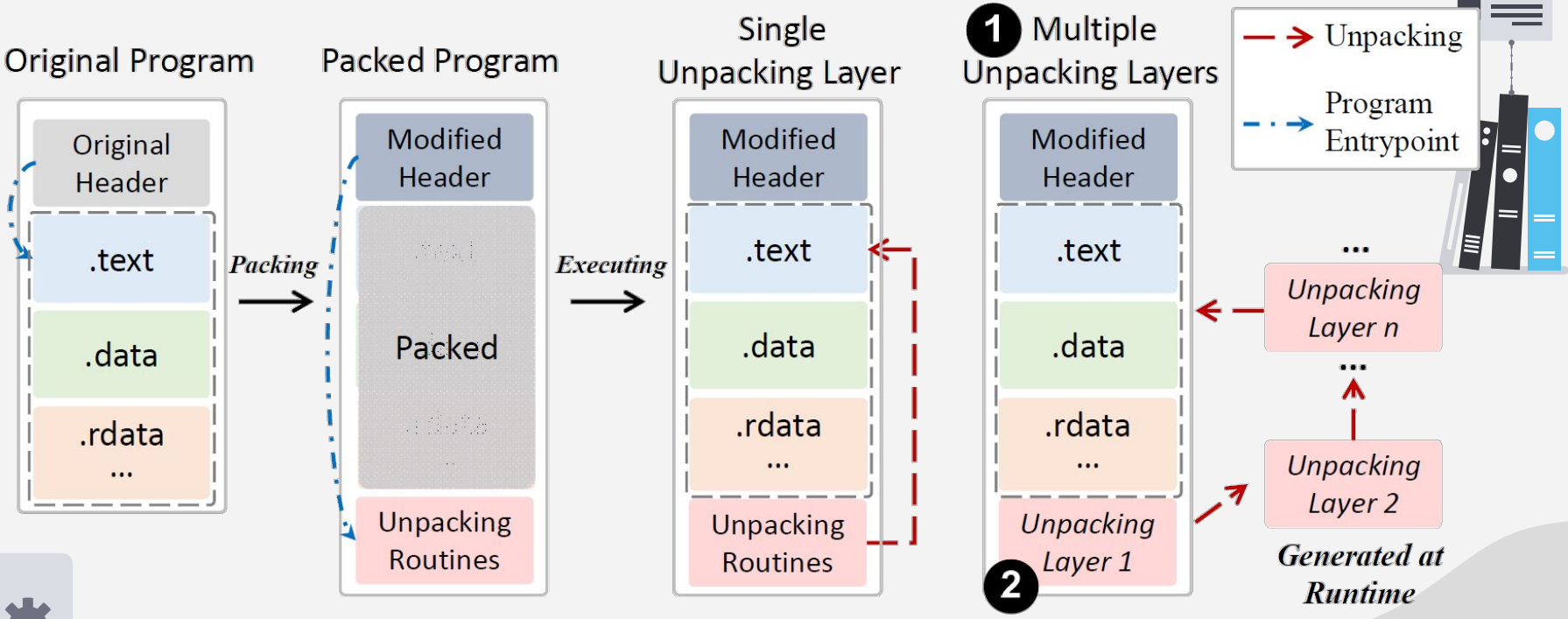
Crypter

目的：加密/混淆
常見工具：ZProtect

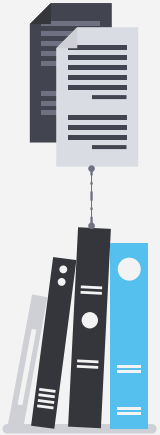
Protector

目的：防止破解或修改
常見工具
：VMProtect、
WProtect

Malware Packing Techniques



Signature-based Packer Detection



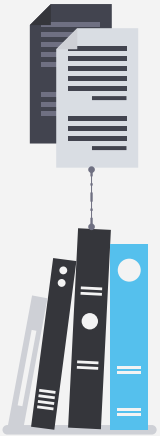
YARA 是 VirusTotal 的開源專案，旨在幫助惡意軟體研究人員識別和分類惡意軟體樣本。根據文字或二進位 pattern 建立惡意軟體的描述（Rule）。

```
rule silent_banker : banker
{
   meta:
      description = "This is just an example"
      threat_level = 3
      in_the_wild = true

   strings:
      $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
      $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
      $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

   condition:
      $a or $b or $c
}
```

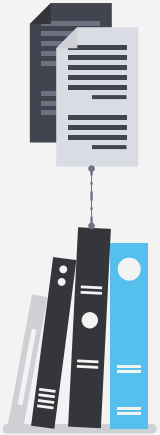
Signature-based Packer Detection



1. text strings（$a）
2. text strings with regular expression
3. hexadecimal strings（$b）
4. hexadecimal string with special constructions（wildcards（?? in $c）,jumps（[4] in $d）,alternatives（57|87） in $d）

```
1 rule UPX {
2 strings:
3   $a = "UPX"
4   $b = {60 E8 00 00 00 00 58 83 E8 3D}
5   $c = {EB ?? ?? ?? ?? ?? 8A 06 46 88 07 47 01 DB 75 07 8B
          1E 83 EE FC 11 DB}
6   $d = {60 E8 [4] 58 83 E8 3D 50 8D B8 [4] (57|87) 8D B0}
```

Signature-based Packer Detection



1. Address-based

   只在特定位址搜尋，如 $b 和 $d，只在 PE 檔的進入點進行搜尋。
   超過 90% 的加殼器偵測規則只在特定位址搜尋，但可以透過更改進入點的指令來繞過。

2. Full-binary matching

   為了增加穩健性，可以搜尋整個二進位檔，如 $a 和 $c，但偵測器只配對位元組的格式而不是 instruction encoding，因此有問題的整體二進制規則會導致高誤判。

```
7  condition:
8    $a and ($b at pe.entry_point or $d at pe.entry_point)
           and $c
9  }
```

Challenges Of Generating Packer Detection Rules

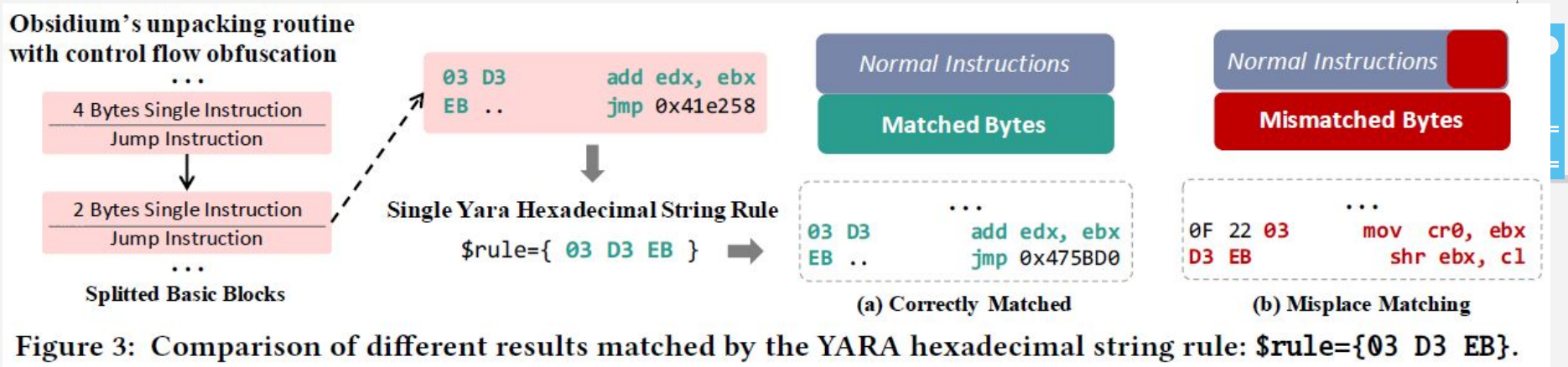
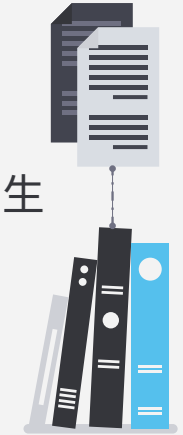


Figure 3: Comparison of different results matched by the YARA hexadecimal string rule: `$rule={03 D3 EB}`.

Challenges Of Generating Packer Detection Rules



緩解這些問題的一個方向是自動生成加殼程式的 signature rule，但現有的自動規則生成器主要關注的是為惡意軟體的 payload 自動生成 signature，而不是為加殼器生成。

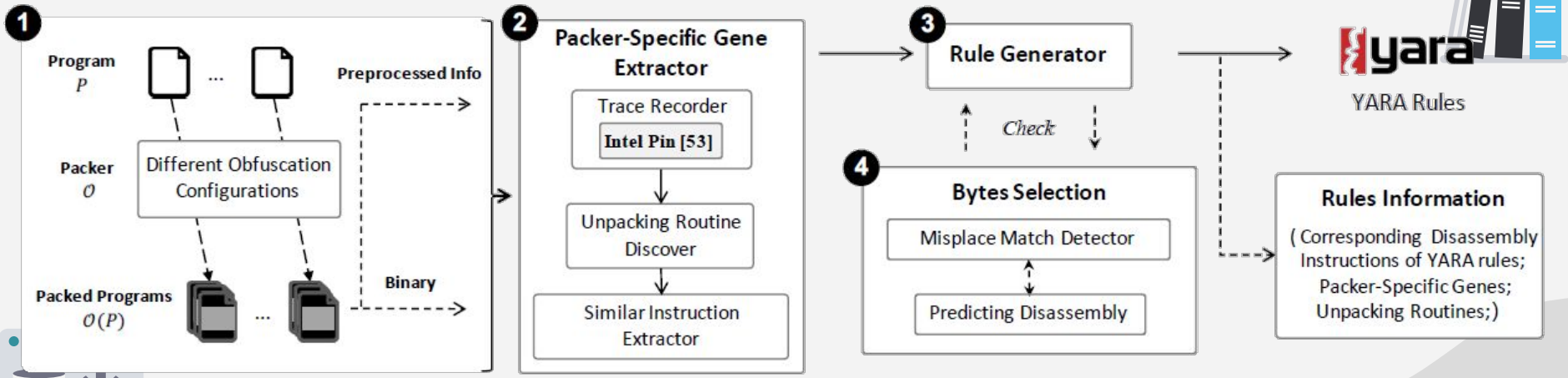
- YaraGenerator
  基於惡意軟體家族之間共用的常見文本特徵（如 string）來生成規則。
- yabin
  使用 function prologues 的固定長度位元組來生成。
- yarGen
  從預先建立的白名單資料庫中過濾出突出文字和十六進位字串來生成規則。
- AutoYara
  結合 heuristic 和 biclustering 演算法，從有限的樣本中找到頻繁出現的 large N-gram 位元組來生成高品質規則。

03

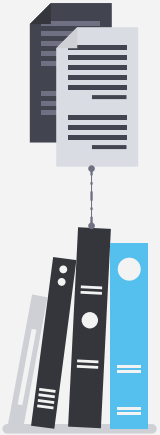
Method

Overview

1. Packed Program Preprocessing
2. Packer-Specific Gene Extraction
3. YARA Rule Generation
4. Byte Selection



Packed Program Preprocessing

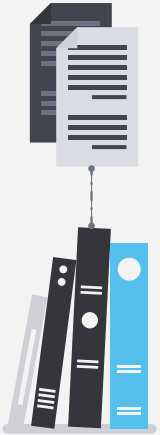


準備加殼程式並從中收集必要資訊，來幫助提取 packer-specific gene。

1. 製作大量的多樣化樣本庫，遍例加殼器的所有配置組合來涵蓋不同的脫殼例程。
2. 收集加殼程式的 section 資訊（如 name 和 address），以幫助在執行時發現靜態可見的脫殼例程指令。
3. 透過收集的資料，監控加殼程式中被寫入並被靜態可見的脫殼例程指令執行的區域，並為這些指令分配標籤。

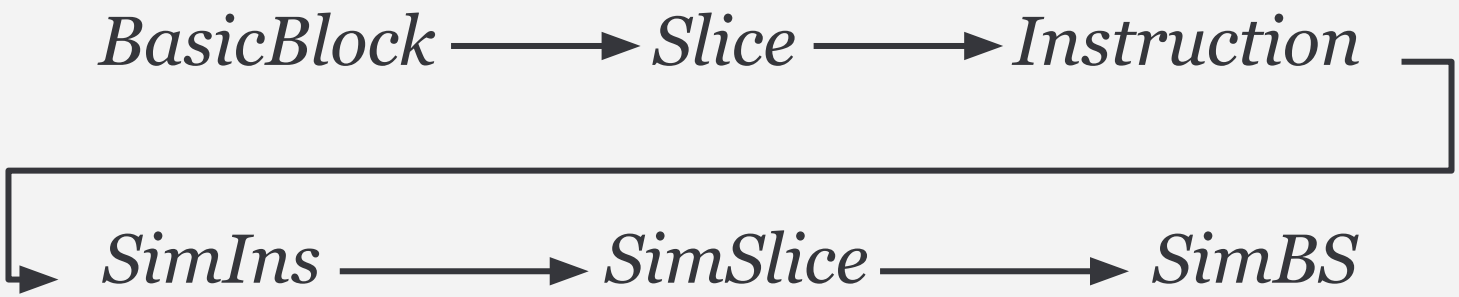
脫殼例程的明顯特徵是必須將脫殼後的原始內容放回原始的虛擬位址上，如執行時脫殼指令必須放在＂.text＂ section 的虛擬位址。

Packer-specific Gene Extraction

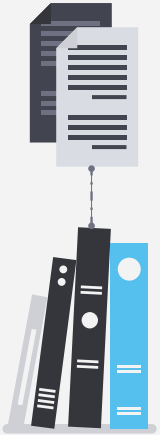


從加殼程式中提取獨特的指令序列（Packer-specific Gene），用於識別加殼技術。

從脫殼例程中重複使用的相似指令中提取出 packer-specific genes。



YARA Rule Generation



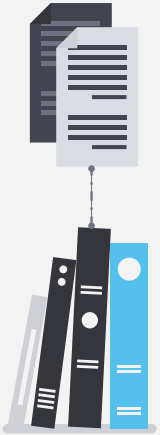
基於相似性資訊從每個 pakcer-specific genes 的 basic block 中生成十六進位字串規則（hexadecimal string rule,HSR）。

若 packer-specific gene 的位元組完全相同，直接將這些位元組轉換成 HSR。
否則從部分相同的位元組中找出相異部分，用詳細的特殊構造來取代並建立 HSR。

HSR 最小長度為 2。

Byte Selection

根據計算 YARA rule 的 misplace match 機率來選擇 YARA rule。

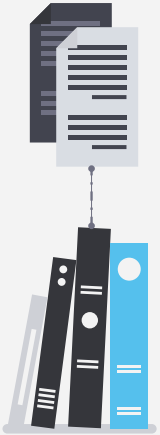


```
1  rule Packer_v1 {
2  strings:
3    $a = {a4 eb} //P_a = 0.7
4    $b = {21 41 3c e8 74} //P_b = 0.5
5    $c = {8b 96 8c 00 00 00 8b c8 c1 e9 10 33 db 8a 1c 11
           8b d3 eb} //P_c = 0
6  condition:
7    $a and $b and $c
8  }
9  rule Packer_v2 {
10 strings:
11   $a = {a4 eb} //P_a = 0.7
12   $b = {21 41 3c e8 74} //P_b = 0.5
13 condition:
14   $a and $b
15 }
```

04

Evaluation

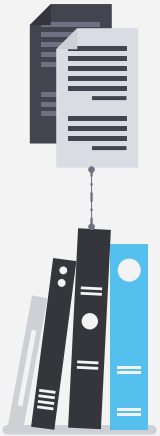
Evaluation



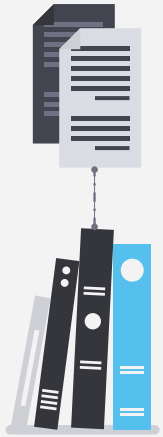
1. PackGenome 是否能有效的生成不同類型的加殼器的偵測規則？
2. 與人工編寫的規則和其他自動化工具相比，PackGenome 的規則準確率和效率如何？
3. PackGenome 生成的規則在偵測加殼樣本的可擴展性如何？
4. PackGenome 生成的規則在偵測現實程式時的性能如何？

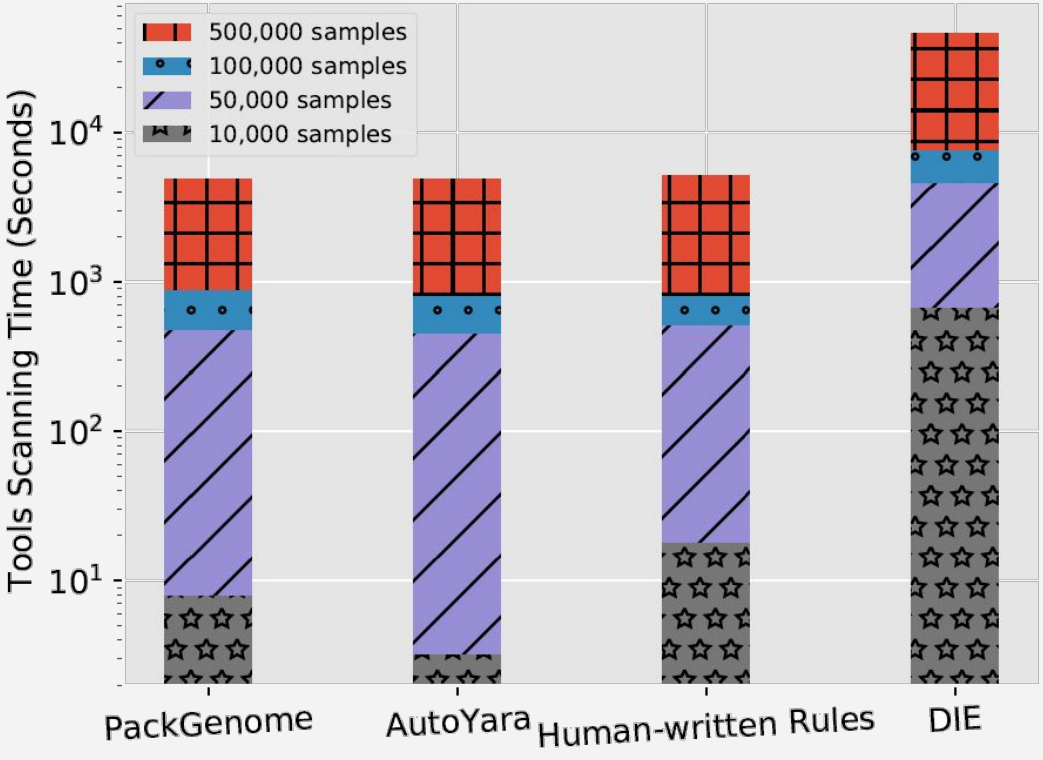
PackGenome是否能有效的生成不同類型的加殼器的偵測規則？

提取 packer-specific genes 並為 20 個現成加殼器生成 70 條規則，Byte selection 可以幫助 PackGenome 生成較低 misplace match 的規則。

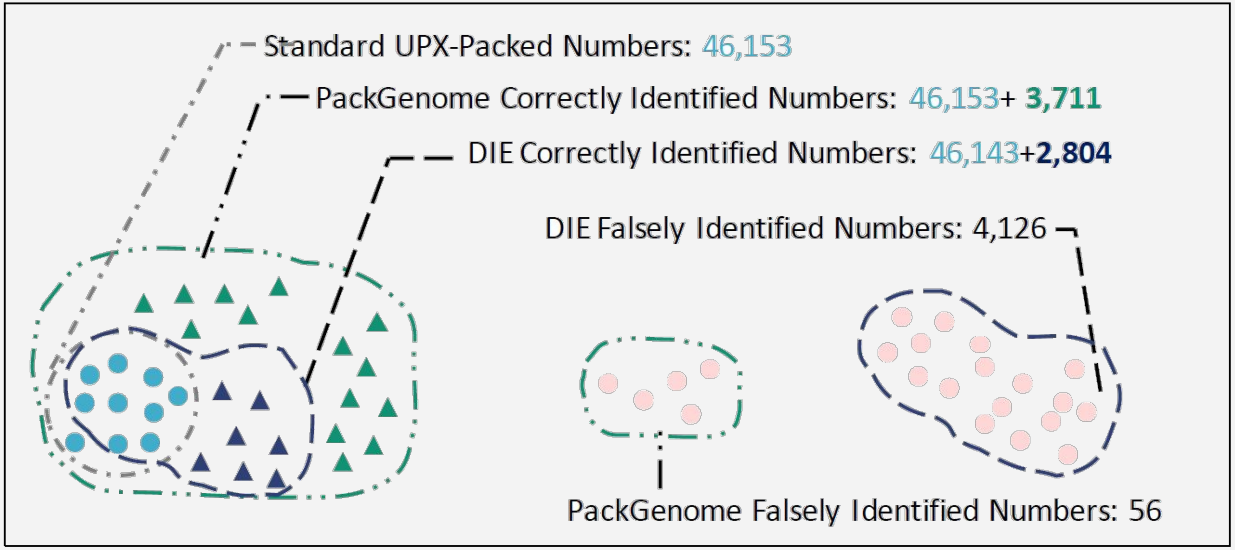


與人工編寫的規則和其他自動化工具相比，PackGenome的規則準確率和效率如 何？



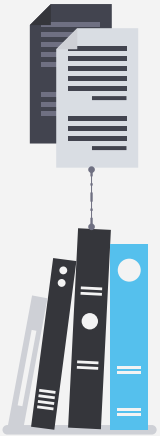


與人工編寫的規則和其他自動化工具相比，PackGenome的規則準確率和效率如 何？



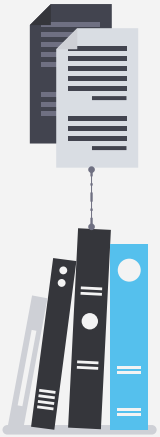
PackGenome生成的規則在偵測加殼樣本的可擴展性如何？

生成的規則適用於不同版本的加殼器，根據 packer specific gene 建立的規則可以直接偵測重複使用相同脫殼例程的客製化加殼器。



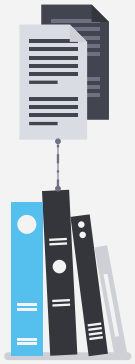
PackGenome生成的規則在偵測現實程式時的性能如何？

可以偵測到客制化加殼器並具有低誤判率。

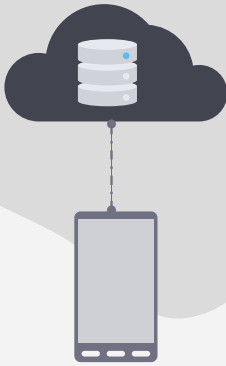


05

Discussion



Discussion



Missing brand-new packers

若能取得新的加殼器，還是能生成規則。或是收集相同加殼器的程式來生成。

Unavoidable byte mismatch

透過 byte selection 來降低 mismatch 機率，在 byte mismatch 和性能之間取得平衡。

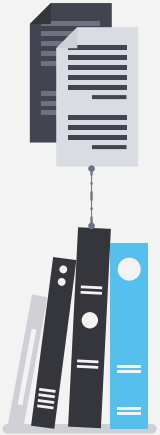
Heavyweight obfuscation

Signature-based 的限制是天生無法處理重度混淆，PackGenome 使用特殊結構來克服輕度混淆。

06

Conclusion

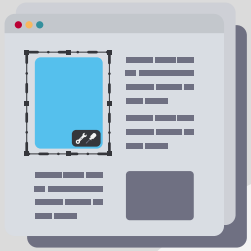
Conclusion



流通的加殼惡意軟體太多，分析人員依靠 signature-based 的偵測來確定加殼器，以便進行脫殼。
但現有的加殼器 signature 生成嚴重依賴分析人員的經驗，這使得編寫和維護規則繁瑣且容易出錯。

PackGenome 是用於加殼器偵測的自動 YARA rule 生成框架，從被相同加殼器的程式重複使用的脫殼例程收集加殼器偵測規則。並提出第一個系統性評估位元組規則 mismatch 機率的模型。

實驗表明，PackGenome 在零漏判、低誤判和微小的掃描開銷方面優於現有的人工規則和工具。



Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**