Extraction of features from binary files and creation of detection patterns

Marek Milkovič

December 14, 2017

Outline

- Extraction of information
- Generation of detection pattern
- Using the pattern

Extraction of information

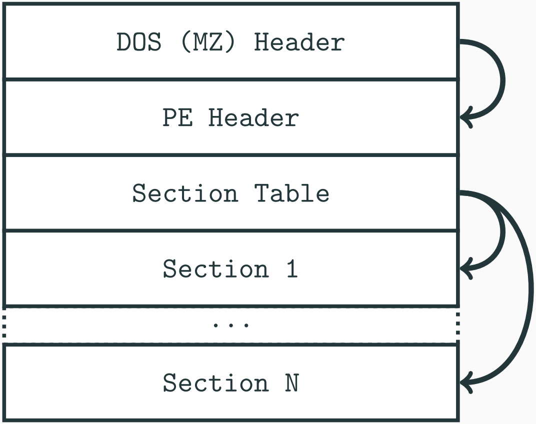
From what?

- Object file formats (OFF)
  - Executable files
  - Dynamic libraries
  - Static libraries

Object File Format

- Different formats across platforms
  - *Windows*
    - PE (**P**ortable **E**xecutable)
    - COFF (**C**ommon **O**bject **F**ile **F**ormat)
  - *Linux* – ELF (**E**xecutable and **L**inkable **F**ormat)
  - *Mac OS X* – Mach-O (**Mach O**bject)
  - *Android* – DEX (**D**alvik **Ex**ecutable)
- No single universal parser

OFF layout



$$\Sigma = \{00, ..., FF\}$$

Parsing of OFF

- Context-free grammars
- Attributed grammars
- Scattered context grammars [2]

Generation of detection pattern

Prevalent information of OFF

- **Primary features**
  - Debug information
  - Imported symbols
  - Section names, addresses, sizes
  - Bytes at entry point
- **Secondary features**
  - Wildcard pattern
  - Fuzzy hash
  - Imphash

Wildcard pattern

- $X = \{x_1, x_2, x_3, ..., x_n\}, x_i \in L \subseteq \Sigma^*$
- Sequences may vary
- $x_1 =$ `5589E5B844332211E842000000..`
- $x_2 =$ `5589E5B8FFEECCBBE856000000..`

Wildcard pattern

- Prefix tree – (V, E, b, s)
- $b : E \to \Sigma$
- $s : V \to 2^X$
- (V, E) is tree
- $\forall v \in V, \forall x = a_1 a_2 ... a_n \in X : x \in s(v) \Leftrightarrow \exists \pi : \pi = e_1 e_2 ... e_n,$ from root to $v, \forall i : 1 \leq i \leq n \wedge b(e_i) = a_i$

Wildcard pattern

- Consider prefix tree as finite automaton
- FA $= (Q, \Sigma, \delta, q_0, F)$
- $Q = V$
- $q_0 =$ root
- $F = \{v \in V | s(v) \neq \emptyset\}$
- $e = (u, v) \in E \wedge b(e) = a \Leftrightarrow v \in \delta(u, a)$
- Minimization

Wildcard pattern

- Ideally, we would replace too varying sequences with wildcards
- *An O(ND) Difference Algorithm and Its Variations* [3]
- *Multiple Longest Common Subsequence problem*
- *Rolling hash*
- NP-hard

Wildcard pattern: YARA

- YARA pattern

```
rule example {
    strings:
        $str = { 55 89 E5 B8 [4] E8 [1] 00 00 00 }
    condition:
        $str at entrypoint
}
```

Using the pattern

Aho-Corasick

- Set of patterns – $\mathcal{P} = \{p_1, p_2, ...\}$
- Match all of them against target $\mathcal{T}$
- Aho-Corasick automaton [1]

Aho-Corasick automaton

- Enhanced prefix tree
- $(Q, \Sigma, \delta, q_0, \phi, \omega)$
- $\phi : Q \to Q$ – failure function
- $\omega : Q \to 2^{\mathcal{P}}$ – output function
- $\delta(q, a)$ not defined
- $(q, aw) \vdash_{\phi} (\phi(q), aw)$ – failure transition
- How to define ϕ and ω?

Aho-Corasick automaton: Failure function

- Distance – Number of transitions from start state
- Q_n – States in distance n
- $suffix(q)$ – Return state representing suffixes of string associated with q

1. $\forall q \in Q_1 : \phi(q) \leftarrow q_0$
2. $i \leftarrow 2$
3. $\forall q \in Q_i : \phi(q) \leftarrow longest\ suffix(q)$
4. $i \leftarrow i + 1$
5. Repeat 3 until all distances are done

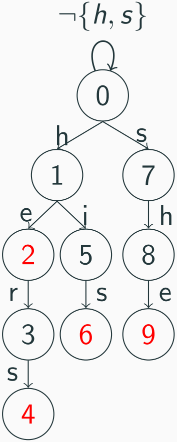
Aho-Corasick automaton: Output function

- *state*(p) – Returns state for pattern p

1. $\forall q \in Q_1 : \phi(q) \leftarrow q_0, \omega(q) \leftarrow \emptyset$
   $\forall p \in \mathcal{P} : \omega(state(p)) \leftarrow \{p\}$
2. $i \leftarrow 2$
3. $\forall q \in Q_i :$
   $\phi(q) \leftarrow longest\ suffix(q)$
   $\omega(q) \leftarrow \omega(q) \cup \omega(longest\ suffix(q))$
4. $i \leftarrow i + 1$
5. Repeat 3 until all distances are done

Aho-Corasick automaton: Example

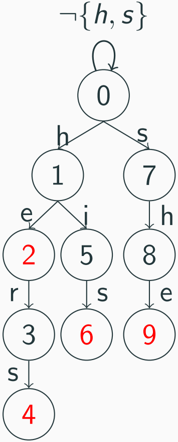
$$\mathcal{P} = \{he, she, his, hers\}$$



State	ϕ	ω
0	0	$\emptyset$
1	0	$\emptyset$
2	0	$\{he\}$
3	0	$\emptyset$
4	7	$\{hers\}$
5	0	$\emptyset$
6	7	$\{his\}$
7	0	$\emptyset$
8	1	$\emptyset$
9	2	$\{she, he\}$

Aho-Corasick automaton: Example

$$\mathcal{P} = \{he, she, his, hers\}$$



$(0, ashers, \emptyset)$
$\vdash (0, shers, \emptyset)$
$\vdash (7, hers, \emptyset)$
$\vdash (8, ers, \emptyset)$
$\vdash (9, rs, \{he, she\})$
$\vdash_\phi (2, rs, \emptyset)$
$\vdash (3, s, \emptyset)$
$\vdash (4, \varepsilon, \{hers\})$

Future Work

- Find efficient strategy and solution for wildcarding patterns
- Add dynamic detection patterns based on behavior
- Investigate what other features are prevalent in malware
- Similarity matching

References

A. V. Aho and M. J. Corasick.
Efficient string matching: An aid to bibliographic search.
Commun. ACM, 18(6):333–340, June 1975.

J. Křoustek and D. Kolář.
Context parsing (not only) of the object-file-format description language.
Computer Science and Information Systems (ComSIS), 10(4):1673–1702, 2013.

E. W. Myers.
An O(ND) difference algorithm and its variations.
Algorithmica, 1(1):251–266, Nov 1986.

Thank you for your attention!