CS7038 - Malware Analysis - Wk08.2

Analysis of Complex Data Structures

Coleman Kane
kaneca@mail.uc.edu

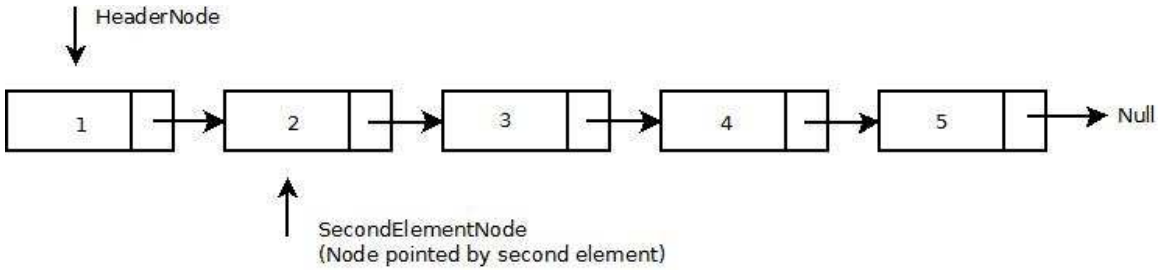
March 2, 2017

Dynamically-allocated Structures

The previous lecture discussed data organization using the C programming language. It focused on a lot of the lower level atomic datatypes, numeric encoding, and some of the primitive structures built atop that foundation.

However, most higher-level languages, and especially more powerful frameworks, make heavier use of complex multi-level data structures and dynamic allocation.

Linked Lists

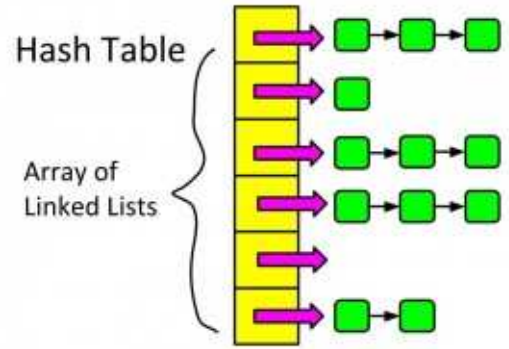


A linked list is arguably one of the core data structures used for dynamic allocation in many programming languages and frameworks.

As we observed with the 8.1 lecture, when using the common run-time allocation calls `new`/`delete` and `malloc`/`free`, the data organization in memory may not reflect the programmatic organization from the author.

In many cases, it becomes our job as reverse engineers to rebuild the author's structure, through malware analysis.

Hash Tables

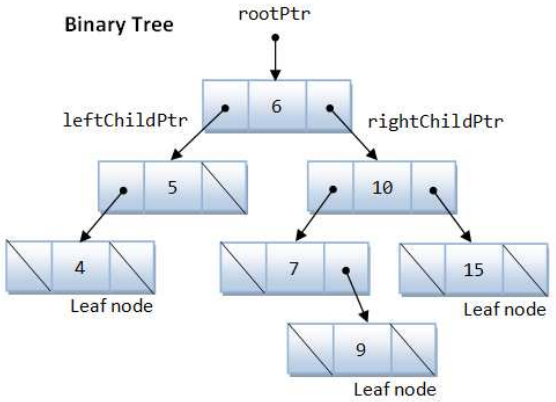


Hash tables are built from linked lists, and frequently at their core are built out of an array of pointers to the head element of a linked list. So there are really two data structures comprising the environment:

- Contiguous array
- Linked list

This provides two structures of indirection that will need to be navigated and rebuilt during analysis.

B-Tree



Binary trees are very similar to linked lists, but have two "`next`" pointers. Additionally, the data may no longer have an exact natural ordering to it, and this may complicate analysis of it.

Arguably, the biggest distinction between this and and the linked list that will make a difference for you is that there may be multiple configuarations representing the same set of data.

C++ Data Structures

In C++ it is common to use the *Standard Template Library* and its derivatives for organizing data. Under the hood, many of these elements utilize primitives similar to what we've documented in the past two classes.

- `std::vector`
- `std::list`
- `std::queue`
- `std::stack`
- `std::map`, `std::unordered_map`, `std::set`, `std::hash_map`
- `std::string`

Also, custom-rolled use-case-efficient versions of these are commonplace among seasoned C++ programmers

Other Langauges

Other langauges, such as Perl, Java, Python, PHP, C# utilize these internally as well:

- LinkedList: Java/C#
- Python lists, Perl arrays/slices
- Python dict, Perl hashes
- Java/C# Hashtable, HashMap, Map
- Java/C# Dictionary
- *Many, many, more ...*

Later we will try analyzing malware in these languages, and knowledge of this layout is imperative to understanding data storage in these systems.