CS527 Final
Heepy Heap Visualization Tool

RYAN HENNESSEE, NATHAN PEERCY, ROWAN HART

MAY 3, 2020

1 Goal and Motivation

This project is aimed at making reverse engineering heap exploits easier. The core concept of this project is to visualize the changes in the heap in real time. This project is motivated by the lack of live visualization methods currently in existence. Most tools either visualize a small amount of information non-visually or only display heap activity after the program has completed execution. Neither of these methods cover the full case of displaying the heap live in a visual and more beginner-friendly method that we want to support with this tool.

2 Related Work

This project has been inspired by shellphish's malloc playground in how2heap, the heapinspect tool, and GEF's heap tools. Malloc playground is a nice place to experiment with heap exploits in a controlled environment, but it is unable to visualize a real program. The heapinspect tool is able to show the chunk attributes, but only on glibc versions: 2.19, 2.23-2.27. These are great command line tools that help visualize the heap. The main issue with these tools is that they are difficult or impossible to visualize all of the chunk attributes or unable to visualize some versions of glibc. A secondary issue is that these tools only display information in the CLI, which is more difficult for beginners and more difficult to read. Heepy is aimed to fill the gap that some of the other tools have yet to fix.

The tool villoc does visualize heap activity in a graphical format, but it does not do so in a live manner and is essentially an ltrace parser that operates on the text information ltrace outputs. Heepy provides more comprehensive visualization with an interactive display.

3 Heepy Implementation

Heepy is able to visualize all aspects of a chunk on all standard libc releases. This is accomplished by parsing through each glibc version's malloc.c (currently only mainline glibc releases from 2.10 through 2.31 are automatically obtained, but the system theoretically supports additional versions). This process uses multiple parser passes to pick up defines as well as types, then obtains the members of malloc structs and deduces each size, including arrays. After the layout of the struct is parsed, it is passed to a json file that is used as a template. This template comes into play when the program is run with heepy.

Heepy hooks the program's malloc calls with gdb and sends the information before and after the call to the backend heepy NodeJS server through a websocket. Once the information has been sent to the backend server, Heepy sorts the current state of the heap by address and forwards that information to the Heepy frontend server. The Heepy frontend server displays the heap chunks and all of its attributes. The frontend cleanly displays the organized groups based on what list the chunk is in using VisJS, a graph browser based visualization library. As an additional feature, users

can rearrange the graph in any way they feel best displays the heap information for them. The web implementation allows for an easy to read and visually pleasing display.

4 Tool Features and Limitations

Our tool features include:

1. glibc version independence.
2. Hooking malloc and free events.
3. Live heap visualization while interacting with a program normally.
4. Web interface for heap visualization.

Our tool does have a few limitations:

1. Linux support only.
2. Theoretical support for additional allocators, but they must follow a similar data structure paradigm as glibc.
3. Lingering bugs with some heap events. These can be fixed, but the initial version occasionaly misses heap objects for various reasons.

5 Use Case Examples

Our visualizer has several potential use cases. First, exploits involving the heap often require an understanding of how the program performs allocations and frees. The ability to visually inspect this process makes it much faster and easier to gain an understanding of the heap layout. The tool can also be used to show heap corruption such as tcache poisoning by quickly inspecting the addresses and contents of the malloc heap structure. This tool can also be used in debugging during development by, as before, displaying the heap structures and contents in an easy to quickly analyze manner. Theoretically there is no limit to potential use cases as this tool could visualize any program that uses the heap, which makes Heepy a very powerful tool.

6 Source Code

Source code for the project can be found at https://github.com/novafacing/heepy.