Since memory allocation problem is ubiquitous in computer science, this academic work to determine the efficient and secure allocation mechanism becomes more important these days. In the following we want to present an overview of related work in this area.

As aforementioned, systems with insufficient user data validation might deal with dynamic memory storage. Dewey et al. (2015) formulates the “use after free” vulnerability and conditions of such attacks. The paper has focused on C++ compiled binaries where the memory manager cannot find and update pointers to program objects when they are moved. This same vulnerability might be presented in other languages. The authors have demonstrated the example of code with attack proof of concept.

Further, Qiang Zeng et al. (2019) classifies different attack types into such systems. The paper above notices how programs might be patched to gain required defense against “use after free” type. However, the method they mentioned requires 64 bits of metadata for every memory allocation call and 4Kb of guard pages. In this work we propose the algorithm with less memory usage.

The other approach to prevent vulnerability was demonstrated and analyzed by Jonathan Ganz et al. (2017). It suggests randomizing the address space and returning random address pointers. Moreover, authors mentioned that most operating systems use current approach. With respect to our work, we also use randomization approach to obtain security.

As mentioned in [4], methods like TSLF are the most effective ones from the time complexity perspective. At the same time the additional memory usage is required due to the hash map. Moreover, Masmano et al. (2004) has introduced the TSLF algorithm that has O(1) time complexity and demonstrated the proof of such behavior. The authors also overviewed the segregated free list like method and compared it with the provided one. It is mentioned that such allocators do not use the hash map and therefore they are free of its memory. Our paper introduces the time and memory balanced protocol of allocation based on segregated free list type.

Another question which corresponds with memory allocation is fragmentation. Nikola Zlatanov (2015) has overviewed this phenomenon and suggested to define a series of partition pools with block sizes in a geometric progression. We suppose that such approach tends to use memory inefficient. Nevertheless, as mentioned in [7], the current phenomenon can be avoided. In addition, [realization](https://context.reverso.net/%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4/%D0%B0%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9-%D1%80%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9/realization)s of the best policies are already known and might be implemented on the client side.

In terms of safety and efficiency at the same time, Beichen Liu et al. (2019) has introduced so-called “SlimGuard” allocator that is designed to be secure and effective. The authors have compared memory and time usage of SlimGuard with different state-of-the art memory management algorithms. Similar to this, we introduce lightweight allocator which performance still needs to be tested and compared with others.

As can be seen in the literature review above, state-of-the art memory management allocators are lack of either time/memory efficiency or attack protection. Only experimental methods try to approach the optimal state in both directions. In our study we have collected different ideas and proposed time and memory balanced allocator.

[1] David Dewey, Bradley Reaves, P. Traynor “Uncovering Use-After-Free Conditions in Compiled Code” // International Conference on Availability, Reliability and Security, ARES, 2015

[2] Qiang Zeng, Golam Kayas, Emil Mohammed, Lannan Luo, Xiaojiang Du, Junghwan Rhee “HeapTherapy+: Efficient Handling of (Almost) All Heap Vulnerabilities Using Targeted Calling-Context Encoding” // 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2019

[3] Jonathan Ganz, Sean Peisert “ASLR: How Robust Is the Randomness?” // IEEE Cybersecurity Development (SecDev), 2017

[4] Paul R. WilsonMark S. JohnstoneMichael NeelyDavid Boles “Dynamic Storage Allocation A Survey and Critical Review” // International Workshop on Memory Management, 1995

[5] M. Masmano U, I. Ripoll, A. Crespo, J. Real “TLSF: A new dynamic memory allocator for real-time systems” // Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004

[6] Nikola Zlatanov “Dynamic Memory Allocation and Fragmentation” // ESC, 2015

[7] Mark S. Johnstone, Paul R. Wilson “The Memory Fragmentation Problem: Solved?” // ACM SIGPLAN Notices, 1998

[8] Beichen Liu, Pierre Olivier, Binoy Ravindran “SlimGuard: A Secure and Memory-Efficient Heap Allocator” // Middleware '19: Proceedings of the 20th International Middleware Conference, 2019