/\* TO DO: Paper Title \*/

/\* TO DO \*/

line 1: 1st Given Name Surname   
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

/\* TO DO \*/

line 1: 4th Given Name Surname  
line 2: *dept. name of organization*  
*(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

*Abstract* — /\* TO DO \*/

Keywords — /\* TO DO \*/

# Introduction

/\* TO DO: area of research, relevance of research, problem statement \*/

/\* TO DO?: Transfer Related work section\*/

Since memory allocation problem is ubiquitous in computer science, this aca-demic 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++ com-piled binaries where the memory manager cannot find and update pointers to pro-gram 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 re-turning 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 re-quired 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 men-tioned in [7], the current phenomenon can be avoided. In addition, realizations 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 intro-duce 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.

/\* TO DO sections (plan of the paper)\*/

# Proposed Algorithm

/\* TO DO:

## Talk about OS memory management like mmap

## Linked list (General) and free blocks list

## Talk about big objects

## Splitting blocks if possible to save memory

\*/

///////////////////////////

# Ease of Use

## Selecting a Template

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the A4 paper size. If you are using US letter-sized paper, please close this file and download the Microsoft Word, Letter file.

## Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

# Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

## Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

## Units

* Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as “3.5-inch disk drive”.
* Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
* Do not mix complete spellings and abbreviations of units: “Wb/m2” or “webers per square meter”, not “webers/m2”. Spell out units when they appear in text: “. . . a few henries”, not “. . . a few H”.
* Use a zero before decimal points: “0.25”, not “.25”. Use “cm3”, not “cc”. (*bullet list*)

## Equations

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

Number equations consecutively. Equation numbers, within parentheses, are to position flush right, as in (1), using a right tab stop. To make your equations more compact, you may use the solidus ( / ), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

*a**b* 

Note that the equation is centered using a center tab stop. Be sure that the symbols in your equation have been defined before or immediately following the equation. Use “(1)”, not “Eq. (1)” or “equation (1)”, except at the beginning of a sentence: “Equation (1) is . . .”

## Some Common Mistakes

* The word “data” is plural, not singular.
* The subscript for the permeability of vacuum **0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter “o”.
* In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
* A graph within a graph is an “inset”, not an “insert”. The word alternatively is preferred to the word “alternately” (unless you really mean something that alternates).
* Do not use the word “essentially” to mean “approximately” or “effectively”.
* In your paper title, if the words “that uses” can accurately replace the word “using”, capitalize the “u”; if not, keep using lower-cased.
* Be aware of the different meanings of the homophones “affect” and “effect”, “complement” and “compliment”, “discreet” and “discrete”, “principal” and “principle”.
* Do not confuse “imply” and “infer”.
* The prefix “non” is not a word; it should be joined to the word it modifies, usually without a hyphen.
* There is no period after the “et” in the Latin abbreviation “et al.”.
* The abbreviation “i.e.” means “that is”, and the abbreviation “e.g.” means “for example”.

An excellent style manual for science writers is [7].

# Using the Template

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

## Authors and Affiliations

**The template is designed for, but not limited to, six authors.** A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

### For papers with more than six authors: Add author names horizontally, moving to a third row if needed for more than 8 authors.

### For papers with less than six authors: To change the default, adjust the template as follows.

#### Selection: Highlight all author and affiliation lines.

#### Change number of columns: Select the Columns icon from the MS Word Standard toolbar and then select the correct number of columns from the selection palette.

#### Deletion: Delete the author and affiliation lines for the extra authors.

## Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is “Heading 5”. Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract”, will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

## Figures and Tables

#### Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

///////////////

# Conclusion

Different applications that implement custom allocation mechanism might lack enough secure or time/memory efficiency. In this paper we have presented common allocation approaches and implemented new memory management algorithm that is balanced in both requirements. This can protect programs vulnerable to attacks like “use after free”.

In the future we could develop a virtual laboratory for testing memory allocation mechanisms. This could help to measure the security and efficiency level of the current, existing, and future algorithms and compare them. Last but not least pro-posed algorithm could be reconstructed from multithreaded perspective in order to support a wider spectrum of applications.

##### References

1. David Dewey, Bradley Reaves, P. Traynor “Uncovering Use-After-Free Conditions in Compiled Code” // International Conference on Availability, Reliabil-ity and Security, ARES, 2015
2. Qiang Zeng, Golam Kayas, Emil Mohammed, Lannan Luo, Xiaojiang Du, Junghwan Rhee “HeapTherapy+: Efficient Handling of (Almost) All Heap Vulner-abilities Using Targeted Calling-Context Encoding” // 49th Annual IEEE/IFIP In-ternational 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 Confer-ence 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 In-ternational Middleware Conference, 2019

/\* TO DO?: reference format \*/