# A Survey of Unikernel Security: Insights and Trends from a Quantitative Analysis

Alex Wollman
*The Beacom College of Computer and Cyber Sciences*
*Dakota State University*
Madison, SD, USA
0009-0000-6260-2750

John Hastings
*The Beacom College of Computer and Cyber Sciences*
*Dakota State University*
Madison, SD, USA
0000-0003-0871-3622

2024 Cyber Awareness and Research Symposium (CARS) | 979-8-3503-8641-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/CARS61786.2024.10778787

***Abstract*— Unikernels, an evolution of LibOSs, are emerging as a virtualization technology to rival those currently used by cloud providers. Unikernels combine the user and kernel space into one "uni"fied memory space and omit functionality that is not necessary for its application to run, thus drastically reducing the required resources. The removed functionality is significant however, and includes components that have become common security technologies such as Address Space Layout Randomization (ASLR), Data Execution Prevention (DEP), and Non-executable bits (NX bits). This raises questions about the security of unikernels. This research presents a quantitative methodology using TF-IDF to analyze the focus of security discussions within unikernel research literature. An initial corpus of 51 unikernel-related papers spanning 2013-2023 was collected. The systematic selection process detailed in the methodology narrowed down to 33 core papers which were then analyzed for trends in security topics. Analysis found that Memory Protection Extensions (MPX) and DEP were the least frequently occurring topics, while Software Guard Extensions (SGX) was the most frequent topic. The findings quantify priorities and assumptions in unikernel security research, identifying potential risks from underexplored attack surfaces. This study represents the first application of TF-IDF analysis to quantitatively assess trends in unikernel security literature, offering novel insights into the field's development and focus areas. In addition, this approach should be broadly applicable for revealing trends and gaps in other niche security domains.**

***Index Terms*—Unikernel, Unikernel Security, TF-IDF, Mirage, Unikraft, Literature Survey**

## I. Introduction

Virtualization of compute environments began as early as the 1960s as a means to enhance mainframe performance [1]. Continuing through the late 20th and beginning of the 21st century, virtualization saw many advancements including virtual machines (VMs) and containers [2], [3]. These advancements provided both better performance and resource utilization [2], [4], [5], but for a growing cloud environment more advancements were needed [6], [7].

Unikernels are an evolution of an old idea: Library Operating Systems (LibOS). The origins of LibOSs stem from the early 1990s [8]. The principle idea behind LibOSs is that OS functionality is implemented in the form of libraries, customized to increase performance of the application by providing closer access to hardware [8]. However, the rise of VMs around the same time quickly overshadowed LibOSs [9] as a virtualization technique. VMs ability to replicate the entire OS and run applications, compared to the customization required for LibOSs [8], put LibOSs at a disadvantage. Additionally, VMs provided a way for several OSs to run on one physical device [3]. Several decades later, after the flourishing of technological advances like the Internet, LibOSs reimerged as unikernels. [10].

Unikernels are different from VMs and containers in several key ways. Unikernels, as the name implies, "uni"fy the user and kernel address spaces into one [10], [11]. This also means that when the unikernel is compiled, all necessary kernel functionality along with the application code are combined into one binary [10], [12]. Combining the application and kernel into one binary comes with some benefits. Utilizing syscalls to invoke kernel functionality and context switches into and out of kernel are no longer required, and simpler function call implementations can be used instead [8], [10], [13]. The direct invocation of kernel code in this way is where the tradition of LibOSs shines through.

Furthermore, unikernels have a drastically smaller memory requirement than other virtualization techniques [10], [12]. Unikernels achieve this through a build system specifically designed to compile in only the necessary components required to execute the unikernel's application [10], [11]. In contrast, if a Linux VM is built to host a web server, that VM is built with all the core Linux utilities and libraries in addition to any that are required by the web server. Even containers, while not comprising the entire OS, still possess a plethora of functionality above and beyond its intended function.

To help achieve the reduced size, many familiar security features have also been removed including Address Space Layout Randomization (ASLR), Non-executable Bits (NX bits), and stack canaries [10], [14], [15]. Adhering to the reductionist principle adopted by unikernels, these features,

in addition to others, have been removed to save space, simplify the code base, and eliminate extra, unnecessary features [10], [14], [15]. The majority of unikernel literature refer to this reduction of capabilities as a security benefit [15], [16], but not all agree [14]. Similar motivations, if applied to the development cycle of a prominent OS such as Windows, OSX, or Linux, combined with 20 years of advances in exploitation techniques would inevitably lead to a drastic increase in exploit development research.

To conduct a deeper investigation into the implications of removing such security components, this study presents the first application of Term Frequency-Inverse Document Frequency (TF-IDF) [17], [18] analysis to quantitatively assess trends in unikernel security literature, offering a novel perspective on the field's development and focus areas. Specifically, through this quantitative analysis, this study seeks to answer the following research questions:

- **RQ1**: What security term appears most frequently in the corpus?
- **RQ2**: What security term appears least frequently in the corpus?
- **RQ3**: What is the publication trend for unikernel related research?
- **RQ4**: Does the frequency analysis provide any insight into unikernel security?

The outcome of this study is intended to provide insight into unikernel security trends and to help inform future research efforts. **RQ2** specifically intends to provide data to demonstrate what additional research into unikernel security is needed. **RQ3** and **RQ4** intend to investigate publication trends and show if additional research is needed into unikernels.

The rest of this paper is organized as follows. Section II addresses the literature review, its methodology, and the quantitative analysis methodology. Section III shares the findings of the research. Section IV offers a discussion and insight into the findings of the research. Section V discusses related work, and Section VI discusses future research avenues. Finally, Section VII presents the conclusion reached and through this study.

## II. Study Design

### A. Literature Review Methodology

To answer the research questions specified in Section I, the following research goals are established:

- **RG1**: Collect a diverse corpus of research literature related to unikernels.
- **RG2**: Calculate the frequency of security terms and unikernel names in literature.
- **RG3**: Determine the significance of the terms in relation to the corpus of literature.

To satisfy **RG1**, a literature review methodology was developed to ensure only unikernel related literature would be included in this study. The methodology follows three primary steps detailed in the following sections: II-B Literature Search, II-C Literature Selection, and II-D Data Extraction.

### B. Literature Search

In the literature search a tailored, but generic, list of search terms was used to generate an initial set of literature. In order to conduct a thorough investigation into unikernel security, a corpus of literature was collected utilizing different databases.

*1) Database Selection:* This study primarily utilized three databases: (1) IEEE Xplore, (2) The ACM Digital Library, and (3) Google Scholar. IEEE Explore and The ACM Digital Library were selected based upon size, popularity, relevance to the subject, and reputation. Google Scholar was selected to broaden the search scope and utilize alternative data sources, such as Springer.

*2) Search Method:* Search terms such as "unikernels", "unikernel security", and "unikernel debugging" were supplied to the databases listed in II-B1 to generate initial results. These results were read by the researcher and used to create additional, more specific, search terms such as "MirageOS vulnerabilities", "SGX in unikernels", "memory corruption in unikernels", and "security overview of unikernels".

### C. Literature Selection

The literature selection phase utilized exclusion and inclusion criteria to generate a relevant corpus of literature. From the search results, each research paper was evaluated using a simple check list:

- Is the paper about unikernels?
- Does the paper involve unikernels in a substantial manner?

The simplicity of this list is meant to gather a variety of topics in order to better evaluate the field as a whole. The literature was neither screened in favor of, nor against, particular topics in order to avoid artificially impacting the TF-IDF calculation described in II-D. For instance, only selecting security related research papers could skew the TF-IDF value showing a greater than expected frequency of security terms. Similarly, selecting only development focused research papers could have the opposite effect. It was also important to ensure that research papers had more than a passing reference to unikernels. Referencing a specific unikernel once in a research paper could give the impression it is exceedingly rare.

### D. Data Extraction and Frequency Calculation

To address **RG2** and **RG3**, data is extracted from the research papers to calculate how frequently different security terms are used in relation to unikernels. During the data extraction phase, literature is scanned for specific security terms and unikernel names. A security term list was used as the basis to scan the literature and extract the data used

to calculate the TF-IDF values. This list also provided a standard way to extract and eventually compare the collected literature.

The security terms were selected using two methods. The first method comprised the manual extraction of terms from the corpus of literature. Each research paper was read and every unique security term added to a list, including common acronyms and abbreviations. Accounting for these acronyms is important because the full term is often only used once, or only at the beginning of research papers or sections, and its acronym used elsewhere. The second method utilized the researchers' experience, including terms that may not appear in the corpus but are still relevant to the security field. A sample of security acronyms is shown in Table I.

TABLE I
SECURITY ACRONYMS

| Sample Security Acronyms |
|---|
| ASLR, MPK, NX-Bit, CPI |

The search term list also included unikernel names gathered using the same methods described for security terms. To account for variations in term appearance, a term is included in the list multiple times utilizing regular expressions (regex.) The most common variation utilized a single dash ("-") between words: software-defined security versus software defined security. The full list of both security and unikernel terms are provided Tables II and III, respectively.

TABLE II
SECURITY SEARCH TERM LIST

| | |
|---|---|
| ASLR | code-pointer\s?integrity |
| cfi | address\s?space\s?layout\s?randomization |
| cpi | control\s?flow\s?integrity |
| dep | control-flow\s?integrity |
| mpk | data\s?execution\s?prevention |
| mpx | memory\s?protection\s?keys |
| NX-bit | memory\s?protection\s?extensions |
| NX\s?bit | return\s?oriented\s?programming |
| rop | non-executable\s?bit |
| sgx | software-fault\s?isolation |
| sfi | software\s?fault\s?isolation |
| sdsec | software\s?guard\s?extensions |
| no-executable\s?bit | software-defined\s?security |
| stack\s?canaries | software\s?defined\s?security |

To determine the significance of security terms, and address **RG2**, the Term Frequency - Inverse Document Frequency (TF-IDF) algorithm is used. Term Frequency (TF) is a mathematical equation used to determine, per document, the frequency of a term.

TF is defined as the total number of times the specific term $t$ in document $d$ occurs divided by the summation of all terms $t\prime$ in the same document $d$ [18]. The TF value can be useful for comparison of terms within a document, but is not as useful for comparing between documents due to variations in document length. For this research a normalized method was utilized to help account for variations in document length, resulting in an addition and multiplication of 0.5 as shown in the mathematical formula below.

TABLE III
UNIKERNEL SEARCH TERM LIST

| | |
|---|---|
| hermitux | unik |
| minios | clive |
| halvm | graphene-sgx |
| clickos | azalea |
| mirageos | drawbridge |
| includeos | ling |
| nanos | guk |
| osv | runtime.js |
| occlum | unikraft |
| rumprun | torokernel |

$$TFreq(t,d) = 0.5 + 0.5(\frac{f_{t,d}}{\sum_{t\prime \in d}})$$

Inverse Document Frequency (IDF) assigns a weight to each word in a document, inversely proportional to how often the term occurs in a corpus of documents: the more frequent the term the lower the weight [17], [18]. IDF is defined as the logarithm[1] of the total number of documents $N$ divided by the number of documents $d$ within the overall corpus $D$ in which the term $t$ appears [18]. To avoid a possible divide by zero error if a term does not appear, 1.0 is added to the numerator and denominator, as shown in the mathematical formula below.

$$IDFreq(t,d) = \log(\frac{N + 1.0}{(d \in D : t \in d) + 1.0})$$

The TF-IDF is then calculated by multiplying the TF and IDF values. The more documents in which the term appears brings the TF-IDF value closer to 0.0 [17], [18]. Therefore, TF-IDF values that are closer to 1.0 indicate terms that appear less frequently in the corpus. While the formula is straight forward, it is provided below for completeness.

$$f(t,d) = TFreq(t,d) * IDFreq(t,d)$$

This research utilized Python3 version 3.10.12 to write the artifact which scans the corpus of literature and generates the TF, IDF, and TF-IDF values utilizing the term list as shown in Table II. The Python package PyPDF2 [19] reads the PDF documents, and the built-in Python3 regular expression package "re" searches for the terms. Each line of the PDF is read and scanned, searching for each term in the list. An internal structure is used to track the number of occurrences, per research paper, and the sentence containing the term. When all documents are processed, the TF, IDF, and TF-IDF values are calculated.

[1]When researching this algorithm there was ambiguity in the sources for which logarithm to use: base 10 or natural log. For this research the base 10 logarithm was used.

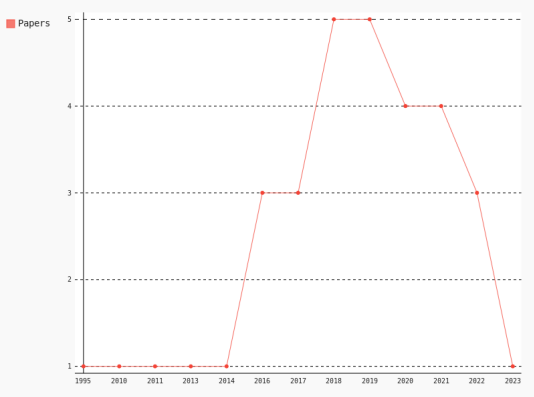


Fig. 1. Corpus Published Dates

## III. Results

### A. Overview

A total of 51 papers were initially selected to comprise the corpus of unikernel-related literature. Following the application of the inclusion and exclusion criteria detailed in Section II-C, the corpus was systematically pruned to 33 papers for in-depth analysis. The literature corpus was published between March 2013 and May 2023, with one outlier from 1995. Table IV shows the number and sources of the pruned corpus, while Table V lists the papers. The majority of papers were found through both ACM (14) and IEEE (12), with the remainder through Google Scholar (7). The individual entities comprising the Google Scholar papers were Defense Technical Information Center, NCCGroup, Usenix, Security in Computing and Communications, Network and Distributed System Security, and the University of the Netherlands. An observation while conducting the search, as described in II-B, is that the current state of unikernel research primarily focuses on development, with some research applying processor specific techniques such as Software Guard Extensions (SGX) [20] and Memory Protection Keys [13].

TABLE IV
Paper Sources and Counts

| Paper Source | Paper Count |
|---|---|
| ACM | 14 |
| Google Scholar | 7 |
| IEEE | 12 |

To address **RQ3**, Figure 1 shows how many papers in the corpus were published, per year, between 1995 and 2023. Beginning in 2014 the number of papers steadily increased to five a year in both 2018 and 2019. An interesting observation is the dip in papers beginning in 2019 and continuing through 2023, declining back to one. The literature review methodology as discussed in Section II-A does not restrict publication dates, which indicates this dip is inherent in the data. A possible explanation for this dip could be the COVID-19 pandemic, however an in-depth investigation is outside the scope of this paper.

TABLE V
Corpus of Papers

| Paper Names |
|---|
| SCONE: Secure Linux Containers with Intel SGX[21] |
| OS Noise Analysis on Azalea-Unikernel[22] |
| Demo: On-The-Fly Generation of Unikernels for Software-Defined Security in Cloud Infrastructures[23] |
| A TOSCA-Oriented Software-Defined Security Approach for Unikernel-Based Protection Clouds[24] |
| Unikernel-based Approach for Software-Defined Security in Cloud Infrastructures[11] |
| A Survey on Security Isolation of Virtualization, Containers, and Unikernels[5] |
| Cloud Cyber Security: Finding an Effective Approach with Unikernels[16] |
| Exokernel: An Operating System Architecture for Application-Level Resource Management[25] |
| USETL: Unikernels for the Serverless Extract Transform and Load; Why Should You Settle for Less?[26] |
| Want More Unikernels? Inflate Them![7] |
| Understanding and Hardening Linux Containers[27] |
| MirageOS Unikernel with Network Acceleration for IoT Cloud Environments[28] |
| Azalea Unikernel IO Offload Acceleration[29] |
| Unikernel Network Functions: A Journey Beyond the Containers[30] |
| Unikraft and the Coming Age of Unikernels[31] |
| fASLR: Function-Based ASLR Via TrustZone-M and MPU for Resource-Constrained IoT Systems[32] |
| Unikernels: Library Operating Systems for the Cloud[10] |
| NCC Group Assessing Unikernel Security[14] |
| A Syscall-Level Binary-Compatible Unikernel[33] |
| Rethinking the Library OS from the Top Down[8] |
| Unikernel Linux (UKL)[34] |
| Virtualization: A Survey on Concepts, Taxonomy And Associated Security Issues[1] |
| Virtualization and Containerization of Application Infrastructure: A Comparison[2] |
| Uniguard Protecting Unikernels using Intel SGX[20] |
| Occlum: Secure and Efficient Multitasking Inside a Single Enclave of Intel SGX[35] |
| Panoply: Low-TCB Linux Applications with SGX Enclaves[36] |
| A Design and Verification Methodology for Secure Isolated Regions[37] |
| Container Security: Issues, Challenges, and the Road Ahead[38] |
| Intra-unikernel Isolation with Intel Memory Protection Keys[13] |
| A Security Perspective on Unikernels[15] |
| A Fresh Look at the Architecture and Performance of Contemporary Isolation Platforms[12] |
| Unikernels as Processes[39] |
| Accelerating Disaggregated Data Centers Using Unikernel[40] |

While collecting papers for the research corpus, only [14] was found to provide a comprehensive analysis of unikernel security. Its findings showcase vulnerabilities that existing security techniques are designed to fix [14], and their removal in unikernels could indicate these vulnerabilities may pose a significant problem once again.

The development and construction of unikernels is the primary focus of the unikernel literature, with some focusing on processor specific security implementations [13], [20] and others discussing security more broadly [15], [16]. Some research papers do not refer to the lack of security features as an issue, in part due to removed features such as “bash” or

"sh" which on traditional systems enabled further exploitation [14], [15]. Instead, literature in the corpus relies on the nature of unikernels and their lack of a diversified code base [14] to provide security, using phrases such as:

- "...unikernel images that integrate protection mechanisms...allow significant reduction of the attack surface."[11]
- "Numerous Unikernels do not implement a shell natively, making most types of payloads, that typically rely on bash, ineffective." [15]
- "Pivot attacks that depend on shell-access are thwarted." [6]

### B. TF-IDF Results

The resulting security term TF-IDF values can be seen in Figure 2. The security terms that appear least frequently with a TF-IDF value of 0.5 or greater are MPK (Memory Protection Keys), Memory Protection Extensions (MPX), Data Execution Prevention (DEP), and Control-Flow Integrity (CFI). These values can be seen in Table VI and are denoted by a single star (*).

Addressing **RQ2**, the least frequently occurring term is tied amongst Memory Protection Extensions, Data Execution Prevention, and Control-Flow Integrity with a value of 0.615. Using their acronyms to break the tie, only CFI is eliminated. Careful observation of the results reveals that a few terms are conspicuously absent: Return Oriented Programming and Software Defined Security. Whereas these terms' acronyms occurred in the corpus, neither of these terms did themselves.

TABLE VI
SECURITY TF-IDF VALUES

| Term | TF-IDF Value |
|---|---|
| ASLR** | 0.288 |
| Address Space Layout Randomization | 0.376 |
| CFI | 0.464 |
| Control-Flow Integrity* | 0.615 |
| DEP* | 0.527 |
| Data Execution Prevention | 0.615 |
| MPK* | 0.527 |
| Memory Protection Keys | 0.527 |
| MPX | 0.527 |
| Memory Protection Extensions* | 0.615 |
| ROP | 0.416 |
| Return Oriented Programming | 0.000 |
| SGX** | 0.245 |
| Software Guard Extensions | 0.288 |
| Stack Canaries | 0.464 |
| SFI | 0.527 |
| Software Fault Isolation | 0.464 |
| SDSec | 0.464 |

To address **RQ1**, the security terms that appear the most frequently with a TF-IDF value of 0.3 or smaller are ASLR and SGX as seen in Figure 2, and are denoted by two stars (**) in Table VI.

The unikernel term TF-IDF values can be seen in Figure 3 and Table VII. The term in the left column of the table

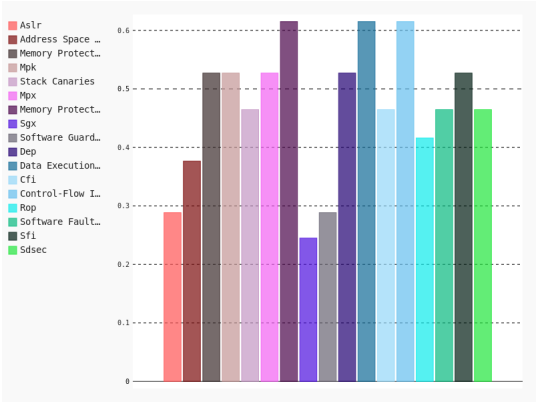


Fig. 2. Security TF-IDF Values

is the unikernel name, with its associated TF-IDF value in the right column. The TF-IDF values for unikernel terms are best interpreted to identify the most frequently discussed unikernels. These unikernels with a TF-IDF value of 0.4 or less are, IncludeOS, MirageOS, OSv, Rumprun, Graphene-SGX, and Ling and are denoted by a single star (*) in Table VII. Missing unikernel terms are Nanos, Unik, Guk, Runtime.js, and Torokernel.

TABLE VII
UNIKERNEL TF-IDF VALUES

| Term | TF-IDF Value |
|---|---|
| Azalea | 0.527 |
| Clickos | 0.265 |
| Clive | 0.464 |
| Drawbridge | 0.376 |
| Graphene-Sgx* | 0.343 |
| Halvm | 0.416 |
| Hermitux | 0.416 |
| Includeos* | 0.208 |
| Ling* | 0.288 |
| Minios | 0.464 |
| Mirageos* | 0.192 |
| Occlum | 0.527 |
| Osv* | 0.192 |
| Rumprun* | 0.314 |
| Unikraft | 0.376 |

## IV. DISCUSSION

The TF-IDF values for the terms ASLR (0.288) and SGX (0.245) indicate that they are common topics in the corpus, revealing key areas of emphasis in unikernel security research. Manual inspection of the occurrences for ASLR reveals a variety of uses. In [13] the term ASLR appears in the context of bypassing ASLR with a vulnerability in order to retrieve sensitive kernel data. However, the encasing paragraph is discussing how the proposed security technique (in this paper MPK) provides protection *despite* the vulnerability. In contrast, [15] directly addresses the lack of ASLR. This manual process reveals that while a TF-IDF value will

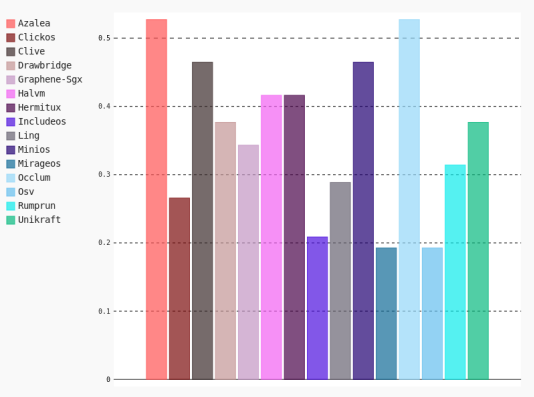


Fig. 3. Unikernel TF-IDF Values

indicate term frequency, it does not provide any context for the frequency.

The need for contextual analysis presents an exciting opportunity for future research. Addressing these variations through context utilization could provide even more nuanced insights into unikernel security considerations with the current approach serving as a stepping stone for developing more sophisticated analytical methods.

It is also more difficult to account for the context of security terms in regard to specific unikernels. While some literature focused on creating a new unikernel [20], other literature investigate multiple unikernels [12], and still other literature choose one unikernel to make generalizations about [14], [15]. In the first and last case, it is possible to identify specific unikernel(s) and then correlate their related security terms appropriately. However, the second case would be difficult to fully automate and guarantee accurate results. This challenge presents an opportunity to develop more advanced techniques for correlating security terms with specific unikernels.

A side effect of accounting for different variations of terms is reflected in the results: acronyms are counted separate from their expanded form. While the term "Control Flow Integrity" doesn't appear in many papers, its acronym "CFI" does, resulting in a variable value for, effectively, the same term. This is discussed further in Section VI.

These points all relate to answering **RQ4**. Where quantitative results provide answers to **RQ1**, **RQ2**, and **RQ3**, and can be used to provide some insight into unikernel security, relying solely on those values does not provide a complete picture. As the discussion in this section indicates, while the methods in this paper provide an initial, general answer to **RQ4**, more specialized quantitative methods are needed to provide a more accurate answer. This presents an exciting opportunity to combine quantitative and qualitative approaches in future research as discussed further in Section VI.

While creating the artifact and evaluating preliminary results, several challenges were discovered. Many papers include a "keyword" section which will affect the term frequency calculation, either by including or omitting words. The impact this has on the value is inappropriate because the term is not used in any context related to the paper, but rather metadata to assist external entities (researchers, search engines, etc.) While manual inspection and removal of these, and other, sentences from the artifact is possible, it would be infeasible to do for a large corpus. Additionally, not all security considerations can be clearly identified through acronyms or well known terms such as ASLR or DEP. For example, privilege escalation can be described as "elevating to root", "gaining root permissions", "getting administrative access", or "getting admin credentials"; generating an exhaustive list of all such phrases would not be feasible.

The Python3 package PyPDF2 also revealed some challenges in the parsing of PDF documents. Variations in the way PDF documents are created result in inconsistencies in PyPDF2's output, one example being spacing between words. In examining this problem, the majority of sentences did not contain any spaces, but seemingly without rational a few sentences would. This complicates the construction of the regex and sentence validation, as the arrangement of acronyms cannot be guaranteed to be unique across word boundaries. In this research only one document[14] was discovered where this occurred, however as additional papers are added they should be monitored for this behavior. These technical challenges are typical hurdles in pioneering research methodologies and addressing them can lead to more robust approaches in future studies.

The TF-IDF values indicate that some security terms are more prevalent in unikernel papers than expected. To provide better context the artifact could be improved to correlate unikernels and their related security terms. This and other possibilities are discussed in Section VI.

While this study focuses specifically on unikernels, the methodology developed here can be broadly applied to other niche security domains, demonstrating its wider value to the research community.

## V. Related Work

TF-IDF has been used as a comparison metric for other term frequency calculation algorithms [41], [42].

In order to get the most accurate calculation, TF-IDF is best suited to parsing documents containing a single topic. [41] proposed a different method which utilized a weighted log-odds ratio with an informative prior to calculate term frequency given a time range. The research used this method to calculate frequency with "noisy" text extracted from cybercrime forums.

[42] calculated the significance of a term "locally" then used that value to help determine the terms "global" significance, utilizing TF-IDF as the catalyst.

> "This novel perspective is a new avenue to develop more novel retrieval models, and it extends the original TF-IDF term weights to model microscopic phenomena at the document-location level, rather than macroscopic phenomena at the document level."

While there is no directly relatable work calculating term frequency for unikernels, TF-IDF is strongly tied to machine learning, natural language processing, and document typing [17], [18] which have all become very popular.

## VI. Future Work

There are several expansions to the artifact that could improve the results. All variations of a security term and its acronym could be combined into one value, giving better insight into its frequency. This could be accomplished by adding a numerical field to the term list. After the individual values are determined, those that share a common numerical value would be summed, and then the new sum used in the TF calculation.

To evaluate the frequency of security terms in relation to specific unikernels, a two-fold approach could be used. One simple case exists if a paper is discussing a single unikernel. Any security term(s) discussed would be done in the context of that unikernel. A more complex case involves discussion of multiple unikernels throughout the research paper. The sentences discussing the security term(s), and possibly those sentences surrounding it, would need closer inspection to identify the context. This could be done by expanding the regex search for unikernel terms to include one or two sentences before or after a security term. If even more context were needed, such as determining a positive or negative connotation to the context, it could transform into a natural language processing (NLP) problem.

The corpus of papers vary greatly in size and therefore word count (ranging from only twelve pages [10] to over one hundred pages [14]). An alternative algorithm might be implemented to account for these variations.

A future objective is to correlate the occurrence of security terms to their corresponding unikernel. This component could provide better insights into the state of unikernel security. The research presented here only shows the frequency of terms, not how they relate. By adding another structure to track security terms as they appear in context to unikernels, a better understanding of unikernel security topics could be evaluated. This could be done in a manner similar to that described above for frequency calculation of security terms to specific unikernels.

Artifact improvements are not the only potential research avenue. A more dynamic classification guide for terms and phrases could expand security insights. A particular problem that arises is the creation of new acronyms based off existing ones. For instance MMDSFI, which stands for MPX-Based (Memory Protection Extensions), Multi-Domain SFI (Software Fault Isolation)[35] would trigger MPX and SFI, but not the actual acronym used. Not only would the count be inaccurate for MPX and SFI, but the current artifact would never detect nor count MMDSFI. These specialized terms would likely only appear in the originating document, but could create bias due to their unique nature. A more dynamic approach could possibly detect these terms and expand the initial list. This would once again, likely transform the work into an NLP problem.

## VII. Conclusion

This research utilized a literature review methodology to collect a corpus of 51 unikernel research papers, which was systematically pruned to a set of 33 for evaluation by TF-IDF. Analysis of the papers revealed a downward trend in unikernel research beginning in 2019 and an overall theme of unikernel development instead of security development. TF-IDF frequency analysis of security terms in the corpus reveals gaps that indicate potential vulnerabilities may be reintroduced with unikernels' reduced capabilities. Terms for foundational security mitigations like Data Execution Prevention (DEP) and Control-Flow Integrity (CFI) were among the least frequently occurring despite their adoption for hardening systems against common attacks.

The implications of the findings are significant for future unikernel research and development. While development simplicity and minimalism are touted as unikernel benefits, the lack of research discussing impacts of forfeited defenses hints at an underestimation of, and the potential for, risks. While unikernels show promise for improving cloud infrastructure security by shrinking the attack surface, more research is needed to explore trade-offs and develop multilayered protections tailored for these special-purpose virtual machines.

This research introduced the first application of TF-IDF analysis to unikernel security literature, providing a quantitative approach to identifying trends in this domain as well as potential gaps. The gaps highlighted in this study can help guide future research efforts toward securing unikernels against attacks without sacrificing their performance advantages. Furthermore, the research demonstrates a literature review and TF-IDF methodology which should be widely applicable to identifying trends and providing insights into other niche security domains.

## References


[1] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," in *2010 Second International Conference on Computer and Network Technology*, 2010, pp. 222–226. DOI: 10.1109/ICCNT.2010.49.

[2] M. J. Scheepers, "Virtualization and containerization of application infrastructure: A comparison," in *21st twente student conference on IT*, vol. 21, 2014, pp. 1–7. [Online]. Available: https://thijs.ai/papers/scheepers-virtualization-containerization.pdf.

[3] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, security threats, and solutions," *ACM Computing Surveys*, vol. 45, no. 2, Mar. 2013. DOI: 10.1145/2431211.2431216.

[4] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1206–1243, 2018. DOI: 10.1109/COMST.2018.2794881.

[5] M. J. De Lucia, "A survey on security isolation of virtualization, containers, and unikernels," Defense Technical Information Center, Tech. Rep., May 2017. [Online]. Available: https://apps.dtic.mil/sti/citations/AD1035194.

[6] A. Happe., B. Duncan., and A. Bratterud., "Unikernels for cloud architectures: How single responsibility can reduce complexity, thus improving enterprise cloud security," in *Proceedings of the 2nd International Conference on Complexity, Future Information Systems and Risk - COMPLEXIS*, INSTICC, SciTePress, 2017, pp. 30–41. DOI: 10.5220/0006282800300041.

[7] G. Gain, C. Soldani, F. Huici, and L. Mathy, "Want more unikernels? inflate them!" In *Proceedings of the 13th Symposium on Cloud Computing*, ser. SoCC '22, San Francisco, California: Association for Computing Machinery, 2022, pp. 510–525. DOI: 10.1145/3542929.3563473.

[8] D. E. Porter, S. Boyd-Wickizer, J. Howell, R. Olinsky, and G. C. Hunt, "Rethinking the library os from the top down," in *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XVI, Newport Beach, California, USA: Association for Computing Machinery, Mar. 2011, pp. 291–304. DOI: 10.1145/1950365.1950399.

[9] E. Bugnion, S. Devine, K. Govil, and M. Rosenblum, "Disco: Running commodity operating systems on scalable multiprocessors," *ACM Transactions on Computer Systems*, vol. 15, no. 4, pp. 412–447, Nov. 1997. DOI: 10.1145/265924.265930.

[10] A. Madhavapeddy, R. Mortier, C. Rotsos, *et al.*, "Unikernels: Library operating systems for the cloud," *SIGARCH Comput. Archit. News*, vol. 41, no. 1, pp. 461–472, Mar. 2013. DOI: 10.1145/2490301.2451167.

[11] M. Compastié, R. Badonnel, O. Festor, R. He, and M. Kassi-Lahlou, "Unikernel-based approach for software-defined security in cloud infrastructures," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–7. DOI: 10.1109/NOMS.2018.8406155.

[12] V. van Rijn and J. S. Rellermeyer, "A fresh look at the architecture and performance of contemporary isolation platforms," in *Proceedings of the 22nd International Middleware Conference*, ser. Middleware '21, Québec city, Canada: Association for Computing Machinery, Dec. 2021, pp. 323–335. DOI: 10.1145/3464298.3493404.

[13] M. Sung, P. Olivier, S. Lankes, and B. Ravindran, "Intra-unikernel isolation with intel memory protection keys," in *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '20, Lausanne, Switzerland: Association for Computing Machinery, Mar. 2020, pp. 143–156. DOI: 10.1145/3381052.3381326.

[14] S. Michaels and J. Dileo, "Assessing unikernel security," NCC Group, Tech. Rep., 2019. [Online]. Available: https://people.cs.pitt.edu/~babay/courses/cs3551/projects/SCADA_Unikernel/NCC_Group-Assessing_Unikernel_Security.pdf.

[15] J. Talbot, P. Pikula, C. Sweetmore, *et al.*, "A security perspective on unikernels," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2020, pp. 1–7. DOI: 10.1109/CyberSecurity49315.2020.9138883.

[16] B. Duncan, A. Happe, and A. Bratterud, "Cloud cyber security: Finding an effective approach with unikernels," in *Advances in Security in Computing and Communications*, J. Sen, Ed., Rijeka: IntechOpen, 2017, ch. 2. DOI: 10.5772/67801.

[17] P. Bafna, D. Pramod, and A. Vaidya, "Document clustering: Tf-idf approach," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, pp. 61–66. DOI: 10.1109/ICEEOT.2016.7754750.

[18] S. Qaiser and R. Ali, "Text mining: Use of TF-IDF to examine the relevance of words to documents," *International Journal of Computer Applications*, vol. 181, no. 1, pp. 25–29, Jul. 2018. DOI: 10.5120/ijca2018917395.

[19] M. Fenniak, *Pypdf2*, https://pypi.org/project/PyPDF2/ Accessed 11/8/2023.

[20] I. Sfyrakis and T. Gross, "Uniguard: Protecting unikernels using Intel SGX," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, 2018, pp. 99–105. DOI: 10.1109/IC2E.2018.00032.

[21] S. Arnautov, B. Trach, F. Gregor, *et al.*, "Scone: Secure linux containers with intel sgx," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16, Savannah, GA, USA: USENIX Association, 2016, pp. 689–703. DOI: 10.5555/3026877.3026930.

[22] S.-J. Cha, S. H. Jeon, Y. J. Jeong, J. M. Kim, and S. Jung, "Os noise analysis on azalea-unikernel," in *2022 24th International Conference on Advanced Communication Technology (ICACT)*, 2022, pp. 81–84. DOI: 10.23919/ICACT53585.2022.9728776.

[23] M. Compastié, R. Badonnel, O. Festor, and R. He, "Demo: On-the-fly generation of unikernels for software-defined security in cloud infrastructures," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–2. DOI: 10.1109/NOMS.2018.8406131.

[24] M. Compastié, R. Badonnel, O. Festor, and R. He, "A tosca-oriented software-defined security approach for unikernel-based protected clouds," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 151–159. DOI: 10.1109/NETSOFT.2019.8806623.

[25] D. R. Engler, M. F. Kaashoek, and J. O'Toole, "Exokernel: An operating system architecture for application-level resource management," *SIGOPS Oper. Syst. Rev.*, vol. 29, no. 5, pp. 251–266, Dec. 1995. DOI: 10.1145/224057.224076.

[26] H. Fingler, A. Akshintala, and C. J. Rossbach, "Usetl: Unikernels for serverless extract transform and load why should you settle for less?" In *Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems*, ser. APSys '19, Hangzhou, China: Association for Computing Machinery, 2019, pp. 23–30. DOI: 10.1145/3343737.3343750.

[27] A. Grattafiori, "Understanding and hardening linux containers," *Whitepaper, NCC Group*, vol. 123, 2016. [Online]. Available: https://www.research.nccgroup.com/wp-content/uploads/2020/07/ncc_group_understanding_hardening_linux_containers-1-1.pdf.

[28] T. Imada, "Mirageos unikernel with network acceleration for iot cloud environments," in *Proceedings of the 2018 2nd International Conference on Cloud and Big Data Computing*, ser. ICCBDC '18, Barcelona, Spain: Association for Computing Machinery, 2018, pp. 1–5. DOI: 10.1145/3264560.3264561.

[29] Y. Jeong, J. Kim, S. Jeon, *et al.*, "Azalea unikernel io offload acceleration," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020, pp. 1377–1380. DOI: 10.1109/ICTC49870.2020.9289322.

[30] T. Kurek, "Unikernel network functions: A journey beyond the containers," *IEEE Communications Magazine*, vol. 57, no. 12, pp. 15–19, 2019. DOI: 10.1109/MCOM.001.1900138.

[31] H. Lefeuvre, G. Gain, D. Dinca, *et al.*, "Unikraft and the coming of age of unikernels," *login; The Usenix Magazine*, 2021. [Online]. Available: https://www.usenix.org/publications/loginonline/unikraft-and-coming-age-unikernels#:~:text=Unikraft%5C%20is%5C%20a%5C%20novel%5C%20micro,developers%5C%20to%5C%20obtain%5C%20high%5C%20performance..

[32] L. Luo, X. Shao, Z. Ling, H. Yan, Y. Wei, and X. Fu, "Faslr: Function-based aslr via trustzone-m and mpu for resource-constrained iot systems," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 120–17 135, 2022. DOI: 10.1109/JIOT.2022.3190374.

[33] P. Olivier, H. Lefeuvre, D. Chiba, S. Lankes, C. Min, and B. Ravindran, "A syscall-level binary-compatible unikernel," *IEEE Transactions on Computers*, vol. 71, no. 9, pp. 2116–2127, 2022. DOI: 10.1109/TC.2021.3122896.

[34] A. Raza, T. Unger, M. Boyd, *et al.*, "Unikernel linux (ukl)," in *Proceedings of the Eighteenth European Conference on Computer Systems*, ser. EuroSys '23, Rome, Italy: Association for Computing Machinery, 2023, pp. 590–605. DOI: 10.1145/3552326.3587458.

[35] Y. Shen, H. Tian, Y. Chen, *et al.*, "Occlum: Secure and efficient multitasking inside a single enclave of Intel SGX," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20, Lausanne, Switzerland: Association for Computing Machinery, Mar. 2020, pp. 955–970. DOI: 10.1145/3373376.3378469.

[36] S. Shinde, D. Le Tien, S. Tople, and P. Saxena, "Panoply: Low-tcb linux applications with sgx enclaves.," in *NDSS*, 2017. DOI: 10.14722/ndss.2017.23500.

[37] R. Sinha, M. Costa, A. Lal, *et al.*, "A design and verification methodology for secure isolated regions," in *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '16, Santa Barbara, CA, USA: Association for Computing Machinery, 2016, pp. 665–681. DOI: 10.1145/2908080.2908113.

[38] S. Sultan, I. Ahmad, and T. Dimitriou, "Container security: Issues, challenges, and the road ahead," *IEEE Access*, vol. 7, pp. 52 976–52 996, 2019. DOI: 10.1109/ACCESS.2019.2911732.

[39] D. Williams, R. Koller, M. Lucina, and N. Prakash, "Unikernels as processes," in *Proceedings of the ACM Symposium on Cloud Computing*, ser. SoCC '18, Carlsbad, CA, USA: Association for Computing Machinery, 2018, pp. 199–211. DOI: 10.1145/3267809.3267845.

[40] W. Yoon, J. Oh, S. Moon, and Y. Kwon, "Accelerating disaggregated data centers using unikernel," in *Proceedings of the SIGCOMM '20 Poster and Demo Sessions*, ser. SIGCOMM '20, Virtual event: Association for Computing Machinery, 2021, pp. 73–75. DOI: 10.1145/3405837.3411397.

[41] J. Hughes, S. Aycock, A. Caines, P. Buttery, and A. Hutchings, "Detecting trending terms in cybersecurity forum discussions," in *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, W. Xu, A. Ritter, T. Baldwin, and A. Rahimi, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 107–115. DOI: 10.18653/v1/2020.wnut-1.15.

[42] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, "Interpreting tf-idf term weights as making relevance decisions," *ACM Trans. Inf. Syst.*, vol. 26, no. 3, Jun. 2008. DOI: 10.1145/1361684.1361686.