Cluster Computing
https://doi.org/10.1007/s10586-024-04355-0

Design tactics for tailoring transformer architectures to cybersecurity challenges

Cigdem Avci[1] · **Bedir Tekinerdogan**[1] · **Cagatay Catal**[2]

Received: 14 August 2023 / Revised: 4 January 2024 / Accepted: 10 February 2024

Abstract
In the rapidly evolving landscape of cyber threats, effective defense strategies are crucial for safeguarding sensitive information and critical systems. Deep learning methods, notably the Transformer architecture, have shown immense potential in addressing cybersecurity challenges. However, customizing, and adapting Transformer architectures for cybersecurity applications presents a challenge, demanding the utilization of effective strategies to achieve optimal performance. This study presents a comprehensive analysis of design tactics employed in tailoring Transformer architectures specifically for cybersecurity problems. Design tactics, defined as strategic solutions to architectural challenges based on well-justified design decisions, are explored in-depth within the context of cybersecurity. By examining the modifications and adaptations made to the original Transformer architecture, this study unveils the design decisions and strategies crucial for successful implementation in diverse cybersecurity domains. The findings emphasize the significance of aligning design tactics with the unique business requirements and quality factors of each specific application domain. This study contributes valuable insights into the utilization of design tactics for customizing Transformer architectures in cybersecurity, paving the way for enhanced defense strategies against the dynamic and evolving nature of cyber threats.

Keywords Cybersecurity · Transformer architectures · Neural networks · Design tactics

1 Introduction

The realm of cybersecurity plays a pivotal role in our interconnected world, where the proliferation of connectivity has led to a parallel evolution of cyber threats and attacks. In this landscape, organizations and individuals grapple with ongoing challenges to ensure the security of software systems and the safeguarding of sensitive information against malicious activities. The aim is to attain the benchmarks of confidentiality, integrity, and availability. As cyber threats advance, it becomes imperative to enhance and adapt cybersecurity measures and strategies in tandem with emerging technologies. The goals encompassed by cybersecurity span preventive, detective, and responsive actions, encompassing the safeguarding of software systems, networks, and data. This encompasses aspects like authorization, authentication, and guarding against unauthorized access, manipulation, and data destruction. Notably, cyber-attacks can result in detrimental financial losses and reputational harm, underscoring the pressing need for robust defense strategies capable of early detection, mitigation, and neutralization of cyber threats.

In order to confront the cybersecurity challenges, machine learning, which is a subfield of artificial intelligence, is widely used. Consolidating algorithms, statistical models and data, machine learning techniques identify patterns of cyber threats and provide relevant information in real time. As an area of machine learning, deep learning techniques gained significant role and attention in cybersecurity field, being able to process with complex features from unstructured data.

✉ Bedir Tekinerdogan
bedir.tekinerdogan@wur.nl

Cigdem Avci
cigdem.avci@wur.nl

Cagatay Catal
ccatal@qu.edu.qa

1 Wageningen University and Research, Information Technology Group, Wageningen, The Netherlands

2 Department of Computer Science and Engineering, Qatar University, Doha, Qatar

In the deep learning field, the transformer architecture is one of the effective state-of-art techniques. Initially, the transformer architectures are used for natural language processing tasks, consequently it is proven that transformers are also applicable and has significant performance when used in capturing dependencies and relations in sequences. Apart from language processing, transformer architectures are applicable to computer vision, time series analysis and cybersecurity. In order to use transformer architectures, modification and adaptation is required at the architecture design and the implementation level for considering the quality factors and criteria of the application domain.

To support design decisions, the notion of design tactics have been introduced that refer to strategic approaches and solutions employed to address specific challenges and make informed decisions during the design and development of a system or architecture. These design tactics are based on well-justified design principles, often aiming to optimize certain quality attributes or meet particular requirements. Design tactics guide architects and designers in making choices related to the structure, components, interactions, and configurations of a system, ensuring that it fulfills its intended goals and functions effectively.

Several studies have discussed specific design solutions for designing transformer architectures, but no explicit and comprehensive study has been performed that identifies and synthesizes the key design tactics for tailoring transformer architectures for cybersecurity systems. To this end, the goal of this study is to explore and analyze the design tactics employed in customizing Transformer architectures to address cybersecurity challenges. By investigating the modifications and adaptations made to the original Transformer architecture, this research aims to provide comprehensive insights into the effective utilization of design tactics. Understanding and identifying these design tactics will facilitate the development of tailored and efficient cybersecurity solutions capable of detecting, analyzing, and mitigating emerging threats. Through this exploration, the study aims to contribute to the advancement of the field by providing valuable knowledge and guidance in the customization of Transformer architectures for cybersecurity applications.

The remainder of this paper is organized as follows: Sect. 2 provides an overview of cybersecurity challenges and the role of machine learning, particularly deep learning, in addressing these challenges. Section 3 presents the Transformer architecture and its applications in various domains, highlighting its potential in cybersecurity. Section 4 delves into the concept of design tactics and their importance in customizing Transformer architectures for cybersecurity. Section 5 discusses specific design tactics employed in the customization of Transformer architectures for cybersecurity applications. Section 6 presents a case study to illustrate the application of design tactics in a real-world cybersecurity scenario. Section 7 reviews related work in the literature. Finally, Sect. 8 concludes the paper, summarizing the contributions and outlining potential future research directions.

2 Background

To establish a solid foundation for subsequent discussions, this section explores the divergences from the original Transformer architecture and provides a comprehensive understanding of its implications.

2.1 Cybersecurity

Cybersecurity encompasses a broad field, focused on safeguarding the confidentiality, integrity, and availability of information. As technology advances and networks expand their reach, the significance of cybersecurity grows, protecting sensitive data and software systems from potential threats. This domain encompasses various areas, including information security, application security, network security, and internet security. Information security concentrates on shielding data, curtailing unauthorized access, and upholding confidentiality. Application security seeks to fortify software against vulnerabilities and malicious attacks. Network security aims to ensure the safety of computer networks, detecting malware and thwarting unauthorized entry. Internet security, on the other hand, safeguards online communications, transactions, and activities.

Cyber threats are observed across society, affecting individuals and organizations alike. A spectrum of issues, from malware and social engineering to ransomware and phishing, underscore the multifaceted challenges in cybersecurity. The consequences of these attacks span reputational harm, financial losses, disruption of vital services, and even legal repercussions. Keeping pace with evolving technological trends and the widespread impact of cyber threats demands cybersecurity systems that are continuously updated and enhanced. To establish effective defenses, routine security assessments are essential. Employing expert threat detection systems and promoting user awareness through education are vital measures in this pursuit. Achieving the desired level of cybersecurity impact necessitates collaboration among stakeholders, including private organizations, government bodies, individuals, and professionals. Establishing cooperation and information sharing, guided by international standards and best practices, becomes a cornerstone for developing successful cybersecurity policies within an organization.

In recent times, artificial intelligence (AI) and machine learning (ML) have emerged as promising assets within cybersecurity systems. AI/ML-powered solutions enable automated threat detection, anomaly recognition, and intelligent response mechanisms that identify and mitigate cyberattacks. This technology-driven approach bolsters human capabilities by efficiently processing vast amounts of data. However, it's essential to note that even with these advancements, resilient systems and advanced defense strategies remain essential, as attackers continually innovate new techniques.

2.2 Transformer Architectures

Transformer architectures form revolutionary models in deep learning for processing sequential data. Being applied initially for natural language processing solutions, transformers have a wide range of application domains one of which is cybersecurity. Transformer architectures' self-attention mechanism separates it from traditional sequential models such as recurrent neural networks (RNNs).

Self-attention mechanism enables the model to weight the importance of the elements in a sequence while processing the sequence simultaneously. The transformer architecture effectively identifies long range dependencies by capturing the global dependencies and relationships. This leads to improved task performance for tasks that require to have a context and relationship understanding. Self-attention layers are the core of transformer architectures. They are the enablers of the model so that each element in the sequence can be assigned weights based on its relevance to other elements. Self-attention enables model to dynamically focus on important features. It weights the features and enhances the model ability such that the context and dependencies are captured.

A transformer architecture consists of an encoder and a decoder component. Processing the input sequence, the encoder generates a representation which encodes the sequence's contextual information. Subsequently, the new representation which is called as the hidden state is used by the decoder while generating an output sequence. Both the encoder and the decoder can be formed of multiple layers, that perform self-attention and feed forward network operations. Positional encoding is also used in transformer architecture for capturing the sequential order of the input elements. With positional encoding, the model can differentiate the elements considering their positions within the sequence. Therefore, the explicit recurrence or convolution operations are not required for transformer architectures. To preserve the temporal characteristics of the data and understand its sequential nature, positional information is vital. An important advantage of the transformer architecture is that it can be functioning in parallel such that all elements in the sequence can be processed simultaneously, which is a difference of transformer architecture from RNNs. Functioning in parallel, the implementation performance is improved during training and prediction and improves efficiency for large-scale applications. The transformer architecture is proven to have a state-of-art performance in tasks such as classification and prediction. Not only in natural language processing where it is initially applied for tasks such as machine translation, sentiment analysis, but also for tasks in image classification, object detection and image generation it has shown to have successful outcomes. For time series analysis, transformer architectures are applied for forecasting, anomaly detection and signal processing.

Transformer architectures have a significant potential for improving cyber defense strategies in the field of cybersecurity. For detecting malicious patterns in network traffic, analyzing system logs for anomaly detection, and identifying patterns in cybersecurity data, transformers can be used. With their contextual understanding ability and adaptability, transformer architectures can be utilized for more effective and accurate threat detection and response systems.

Following a sequence-to-sequence paradigm, the vanilla transformer architecture consists of a stack of identical encoders and decoders (Fig. 1). In this architecture, the encoder blocks are composed of a multi-head self-attention module and a position-wise feed-forward network (FFN). The augmentation with a residual connection enables the formation of deeper transformer models. This way the information flows directly through the module. Afterwards, a layer normalization module ensures improvement of training stability and enables faster convergence [1].

The encoder layer of the transformer architecture converts the input sequence to a continuous representation while it preserves the learned information. The modules that the encoder layer consists of are the multi-headed attention module and the fully connected network. Since the encoder can be composed multiple stacked blocks, it can learn different attention representations with distinct parameters in each block.

The vectors sequentially progress through the encoder stack of the encoder layer. As the input vector arrives a multi-head attention layer with the self-attention mechanism, it flows towards residual connection and layer normalization. A combination and normalized form of the outputs of the multi-head attention is formed. Afterwards a feed-forward network is applied with a residual or skip connection. Eventually, an addition and normalization step is implemented. BERT (Bidirectional Encoder Representations from Transformers) [3] is an example of an encoder-only transformer. Using BERT, the capability of the transformer encoder architecture is presented for capturing bidirectional context in natural language processing tasks with advanced language understanding and generation capacities.

Decoder layer also has importance for transformer architectures from the aspect of generating output sequences. It possesses auto-regressiveness, since as input,

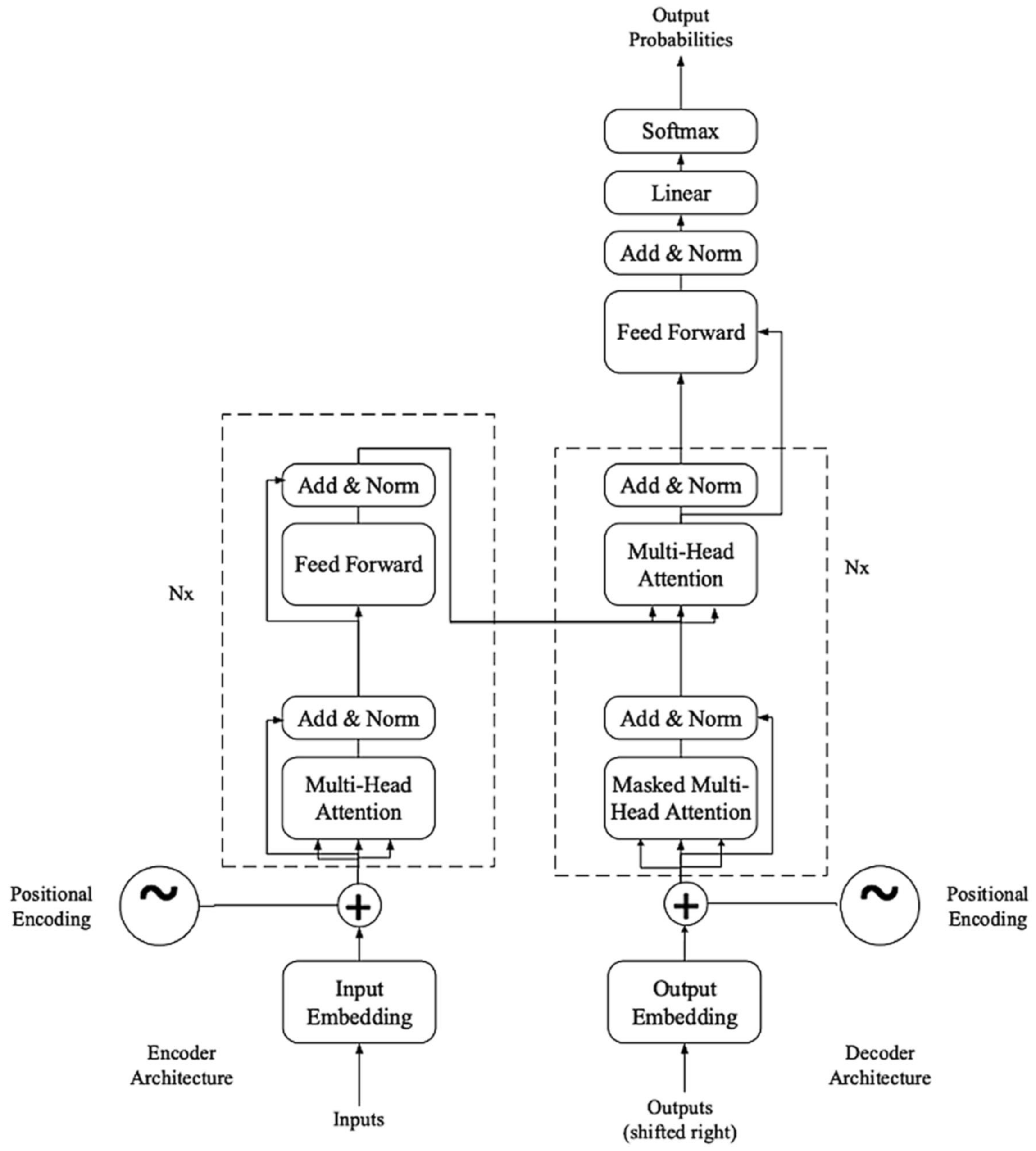


Fig. 1 Vanilla transformer architecture [2]

it utilizes both the previous outputs and the output of the encoder layer. Decoder layer forwards the input sequences to the positional embedding layer to have the positional embedding vectors as an output that includes the sequential information. Afterwards these output vectors are directed to the multi-head attention layer to calculate the attention scores. Decoder layer can have multiple multi-headed attention layers and for each sub-layer residual connections, feed forward layers and layer normalizations are presented in the architecture. In this way, a decoder stack having multiple decoder blocks with different parameters can be formed. The shape of the input and output vectors is sustained through the decoder stack.

While the multi-headed attention layers in decoder are similar to the ones in encoder, they can function on different tasks depending on the requirements of the application. Linear layer can be utilized for classification tasks and the softmax function can be used to calculate the classification probabilities. A look-ahead matrix is completed with an attention score matrix such that the decoder layer does not condition on future input tokens. As a result, the attention of the decoder is on the available information at each time step. GPT-2, GPT-3 and GOPHER are decoder-based architectures and examples of transformer models which demonstrate decoder capabilities including but not limited to language modelling, natural language understanding, and text generation. Furthermore, an example of a multimodal decoder-only transformer is GATO (Generative Adversarial Transformer for Objects), which facilitates visual information for tasks using images and text [3]. Decoder-based transformer models also have significant performance indicating the adaptability and effectiveness of transformer architectures.

2.3 Design tactics

The concept of design tactics that is applied in this study for transformer architectures has been adapted from software design methods. They are high-level design decisions to reach the identified criteria for the quality factors and target performance [41]. The tactics are used as options for design and cover lower-level design decisions for the architects to meet the functionality and quality requirements of the system [5]. In order to address the identified design challenges and guide the design process for achieving the target outcomes, the design tactics are diligently formed.

Design tactics are also helpful during software integration while selecting among the existing software components [41]. Using design tactics, a systematic approach is applied for design decision making and this way the architects can select suitable strategies to meet the quality requirements [6]. The well-documented design tactics are presented as best practices and generic solutions for common design problems and constructing design patters guided by the design tactics for reuse [6].

To apply the design tactics, the trade-offs between different design choices, the desired quality attributes and the constraint about the target problem shall be considered. Implementation approaches, architectural modifications, system behaviors adapted for optimization of performance and effectiveness are guided by the design tactics for the design solution.

Design tactics emerge as a key methodology in tailoring transformer architectures catered to cybersecurity challenges. By seamlessly aligning the transformer architecture with the requisites of cybersecurity solutions, these tactics serve as guiding principles throughout the customization and adaptation process. They intricately steer the modifications and refinements of transformer architectures, meticulously steering them towards the desired cybersecurity outcomes and the criteria mandated by the identified quality factors [4]. Elevating the significance, the utilization of design tactics earmarked for transformer architectures tailored to cybersecurity solutions holds tremendous promise. Researchers and practitioners, leveraging these tactics, can effectively construct bespoke applications adept at not only detecting, but also scrutinizing and defusing cyber threats. As an approach to software design, design tactics proffer a structured and methodical route, intricately navigating the complexities inherent in adapting transformer architectures for cybersecurity purposes.

3 Problem statement

In the literature, various transformer architectures have been proposed, some of which are described in [1] and [7]. The choice of a specific transformer architecture depends on the problem at hand. Based on the identified features and the surveys conducted in [1] and [7], transformers can be categorized according to their implementation area and structure. In [1], a taxonomy for transformers is introduced, considering three perspectives: architectural modification, pre-training, and applications. [1] highlights the improvements in transformers with respect to model efficiency, model generalization, and model adaptation. One of the addressed challenges in [1] is the use of sparse attention variants, which reduce computational complexity and address overfitting by introducing structural priors on the input data. Transformers can be used in three ways: as encoder-decoder models, encoder-only models, and decoder-only models. Furthermore, [1] discusses the taxonomy of transformers, exploring module-level, architecture-level, and application-level variations, as well as pre-trained models. At the module level, variations include activation functions, capacity enlargement, dropping of the feed-forward network module, module placement, substitution, normalization-free models, sparse or linearized attention, query prototyping, memory compression, low-rank self-attention, attention with prior knowledge, improved multi-head attention, absolute position attention, relative position attention, other representations, and implicit representations. Architecture-level variations include cross-lock connectivity, adaptive computation time, recurrency and hierarchy, and alternative architectures [1]. Pre-trained encoder, decoder, and encoder-decoder models are prepared, and their applications can be observed in text, vision, audio, and multi-modal data systems. In [7], the taxonomy of the self-attention design space is presented, focusing on self-attention in vision models. Existing approaches based on self-attention encompass both single-head and multi-head self-attention. Single-head self-attention includes techniques such as non-local attention, crisscross attention, local relation nets, and attention augmented CNN within convolutional neural networks (CNNs). It also includes self-attention as a stand-alone primitive, such as stand-alone self-attention and vector attention. For multi-head self-attention, various design approaches for vision tasks are listed, including uniform scale vision transformers (e.g., vision transformer, data-efficient transformer, token-to-token transformer, transformer-in-transformer, cross-covariance image transformers), multi-scale vision transformers (e.g., pyramid vision transformer, segformer, swin transformer, crossformer, focal transformer), hybrid vision transformers with convolutions (e.g., convolutional vision transformer, compact convolutional transformer, local vision transformer, LeVit, ResT, NesT), and self-supervised vision transformers (e.g., DINO, MoCo v3, EsViT) [7]. These variations demonstrate the diverse design possibilities and applications of self-attention in vision models.

The systematic analysis of the reasons behind selecting a specific transformer architecture is crucial, considering the numerous applications available that align with the research focus of this study. Therefore, in this study, we aim to provide an in-depth exploration of the design tactics used in designing transformer architectures specifically for cybersecurity problems. As previously mentioned, design tactics are applied to address identified quality factors. In the context of cybersecurity, we introduce design tactics specifically tailored for transformer architectures.

4 Research method

Our research methodology follows a domain-oriented approach with distinct domain and application phases [8–10]. In the domain phase, we gathered cybersecurity domain problem categories and related quality factors and consequences while identifying variations within the transformer architecture design and forming a set of design tactics. Once the domain phase reached maturity, we moved into the application phase, where for the selected problem, quality factors were introduced to select the relevant design tactics and tailor the application implementation to create a desired research application. The implementation process of the resulting application is presented with a case study where the tailoring approach for the design of the transformer architecture is discussed, and the experimental results are presented.

To accomplish our research objectives, we followed the outlined steps:

1. We collected studies on transformer architectures that incorporated architectural modifications or enhancements.
2. We identified design tactics that were specific to transformer architectures, considering their application in the cybersecurity domain.
3. We grouped the identified tactics according to the categories of transformer architectures.
4. For each design tactic, we described the design decisions made and provided the rationale behind those decisions.
5. We defined the specific problems addressed by each design tactic and described the corresponding solutions.
6. We mapped each design tactic and its corresponding adaptation/modification to the vanilla transformer architecture, establishing a clear connection between the original architecture and the customized variants.

By following this systematic approach, we aim to contribute to the understanding of design tactics and their application in transformer architectures for cybersecurity problems. This research provides insights into the adaptation and modification of transformer architectures, offering valuable knowledge for researchers and practitioners working in the field of cybersecurity and deep learning.

Table 1 lists the selected papers together with the design tactic*s*. In the following section, we will describe the design tactics listed in Table 1 per selected study.

For the case study section, the research method we have followed for data collection, model training, and validation processes are as follows:

- Data Collection: The selection criteria for the datasets is that the dataset shall be used successfully in a research implementation and alpha-numerical. Datasets mentioned and made public by two studies have been used [26, 27].
- Model Training: The models are run up to 10 epochs. "adam" (a stochastic gradient descent method) is used as the optimizer, and "binary_crossentropy (cross-entropy loss between true labels and predicted labels)" is used for the loss function.
- Validation: The datasets are split into train (75%) and test (25%) data and the test data is used for validation purposes. The metrics for the experiments are precision, recall, and f1_score.

5 Design tactics for transformer architectures

In this section we will first provide a template for describing the design tactics for tailoring transformer architectures for cybersecurity. This is followed by a description of a feature model that defines the common and variant features of the transformer architecture solutions. Finally, we will present each of the design tactics using the tactic template.

Table 1 Selected papers and tactics

#	Title	Tactics
1	Scalable Detection of Promotional Website Defacements in Black Hat SEO Campaigns [11]	Tactic 1: Tag-aware bidirectional encoder (T-BERT)
2	Lightweight URL-based phishing detection using natural language processing transformers for mobile devices [12]	Tactic 2: Pre-trained transformer models available for immediate use
3	A Spam Transformer Model for SMS Spam Detection [13]	Tactic 3: Memory
4	Generating Fake Cyber Threat Intelligence Using Transformer-Based Models [14]	Tactic 4: Pre-trained model GPT-2 with fine tuning
5	URLTran: Improving Phishing URL Detection Using Transformers [15]	Tactic 5: Byte pair encoding (BPE) tokenizers
6	Training Transformers for Information Security Tasks: A Case Study on Malicious URL Prediction [16]	Tactic 6: An auxiliary auto-regressive loss function, balanced mixed objective training
7	Cascaded Multi-Class Network Intrusion Detection with Decision Tree and Self-attentive Model [17]	Tactic 7: Integrating decision tree and feature tokenizer (FT)-transformer
8	MalBERT: Using transformers for cybersecurity and malicious software detection [18]	Tactic 8: Source code as a set of features
9	Self-supervised and interpretable anomaly detection using network transformers [19]	Tactic 9: Network transformer (NeT)
10	An Accuracy-Maximization Approach for Claims Classifiers in Document Content Analytics for Cybersecurity [20]	Tactic 10: ClaimsBert
11	Towards the evolutionary assessment of neural transformers trained on source code [21]. Learning and evaluating contextual embedding of source code	Tactic 11: Code Understanding BERT (CuBERT)
12	Attack Tactic Identification by Transfer Learning of Language Model [22]	Tactic 12: Packet embedding method-based language model (PELAT)
13	Network Intrusion Detection via Flow-to-Image Conversion and Vision Transformer Classification [23]	Tactic 13: Vision transformer (ViT)
14	Detection of false data injection attacks in smart grid: A secure federated deep learning approach [24]. XTM: A Novel Transformer and LSTM-Based Model for Detection and Localization of Formally Verified FDI Attack in Smart Grid [25]	Tactic 14: SecFed-Transformer (secure federated learning)

5.1 Tactic template

The selected studies serve as a basis for defining each tactic as derived from the architecture described in the corresponding research. For each design tactic, a template is used to provide the details of the tactic in a structured way. The template is shown in Table 2.

Each design tactic in the transformer has a unique identifier to distinguish it from others. These tactics are specifically designed to address various cybersecurity problems, including:

- Web Defacement: An attack that modifies the visual appearance of a website or web page.
- SMS Spam: Unwanted messages, particularly advertising, targeted towards text messaging or other mobile communications services.
- Malware Detection: Identifying and mitigating harmful software designed to damage computers, servers, clients, or computer networks. Malware can leak

Table 2 Tactic description template

Title	Description
Transformer Design Tactic Name	Tactic name
Cybersecurity problem	The known area of application where the tactic is implemented
Quality factor	Quality factors that lead the design strategy to different design tactics
Aim of Transformer Modification/Adaptation	The goal of the modification/adaptation
Design decision	The design decisions related to the design of the transformer architecture
Implementation	Implementation issues related to the design decision
Design diagram	Description of the applied tactic to the Transformer Architecture
Consequences	Consequences resulting from applying the tactic

confidential data, grant unauthorized access, block data access, or compromise user security and privacy.

- Data Poisoning Attack: Intentionally misclassifying harmful samples as desired classifications, such as marking spam emails as safe, to manipulate the prediction behavior of a model.
- Phishing Detection: Detecting and preventing social engineering attacks where perpetrators deceive victims into divulging personal information or downloading malicious software.
- Software Vulnerability: Identifying weaknesses or flaws in software code that could be exploited by attackers to compromise security.
- Network Attack: Interference, obstruction, compromise, or destruction of data and computer systems through the exploitation of computer networks.
- Network Intrusion: Unauthorized access to a computer within a business or a permitted domain.
- Network Anomaly: Unusual behaviors or characteristics in a network that are often associated with malicious activity.
- False Data Injection Attack: Attacks that manipulate or modify sensor measurements to affect the computing capabilities of a control center.

To address each of the aforementioned cybersecurity problems, specific quality factors are considered during the design of the transformer. These quality factors can include performance, security, scalability, maintainability, and more. Each design tactic aims to improve one or more of these quality factors in order to effectively tackle the identified problem.

The “Aim of Transformer Modification/Adaptation” field outlines the specific goals or objectives of modifying or adapting the transformer. It highlights the desired outcomes or improvements that the design tactic aims to achieve. For example, the aim might be to enhance the security measures, improve performance efficiency, or ensure better scalability of the transformer.

The “Implementation” provides the issues related to the implementation of the identified design decisions.

The “Design Decision” field encompasses the decisions made during the design process of the transformer architecture, specifically concerning the identified problem and the relevant quality concerns. These decisions guide the selection of suitable design approaches and techniques that align with the design tactic’s objective. The design decisions shape the overall architecture and implementation of the transformer.

The”Solution” section of each tactic describes the specific design of the transformer architecture that addresses the identified problem and aligns with the chosen design decisions. It provides a detailed explanation of the approach taken, including software design, implementation strategies, and behavioral considerations. The solution section highlights how the design tactic is applied to modify or adapt the transformer effectively.

Finally, the “Consequences” field outlines the outcomes that arise from applying the design tactic. It includes both positive and negative impacts that may result from the modification or adaptation of the transformer. Examples of consequences could include improved security measures, increased performance efficiency, added complexity, or potential trade-offs between different quality factors. These consequences help evaluate the overall impact of applying the design tactic.

5.2 Selecting design tactics

When designing transformers for cybersecurity systems, selecting the appropriate tactics is essential. To facilitate the selection process, a feature model is utilized, which provides a comprehensive representation of the mandatory and optional features in a tree or graph format. In this study, the feature model is prepared based on collected and analyzed information specific to transformers in cybersecurity. Figure 3 showcases the feature model for the cybersecurity system transformer, highlighting the essential elements. It serves as a visual representation of the various features and their relationships within the transformer design. In order to align with existing taxonomies, the transformer variants discussed in this study are mapped to the variants listed in [1], which categorizes transformers based on their application area in cybersecurity, primarily focusing on text-based systems. This mapping aids in understanding the specific transformer variants relevant to cybersecurity problems. A feature model is a generic model representation for a problem that lists the mandatory and optional features with a tree or graph form. The feature model presented below, that is prepared using the collected and analyzed information within this study, shows essential elements for transformers in cybersecurity problems. The design tactics can be formed starting with the appropriate tailoring of the feature model.

According to the taxonomy of transformers in [1], the transformer application area in cybersecurity is mainly text based systems. Below is the Table 3 that maps the transformer variants discussed in this study to the variants that are listed in [1]:

In [28], another taxonomy is presented for efficient transformers. Unlike the previous taxonomies, the categorization covers recurrence, memory / down sampling, learnable parameters, low rank / kernels, and fixed / factorized / random patterns. For cybersecurity systems, we have observed that memory and recurrence can be a variation point in transformer architecture. Furthermore,

Table 3 Mapping of transformer variants in [1] to the features in Fig. 2

#	Transformer variant in [1]	Transformer variant for cybersecurity systems
1	Module level–attention–sparse	Embedding–token
2	Module level–position encoding	Embedding–positional
3	Module level–attention–sparse	Embedding–weighted sentence vector
4	Module level–attention–prior attention	Architecture–residual connection
5	Architecture level–alternative architecture	Architecture–stacking
6	N/A	Input–token processing
7	N/A	Output–integration with hybrid architecture

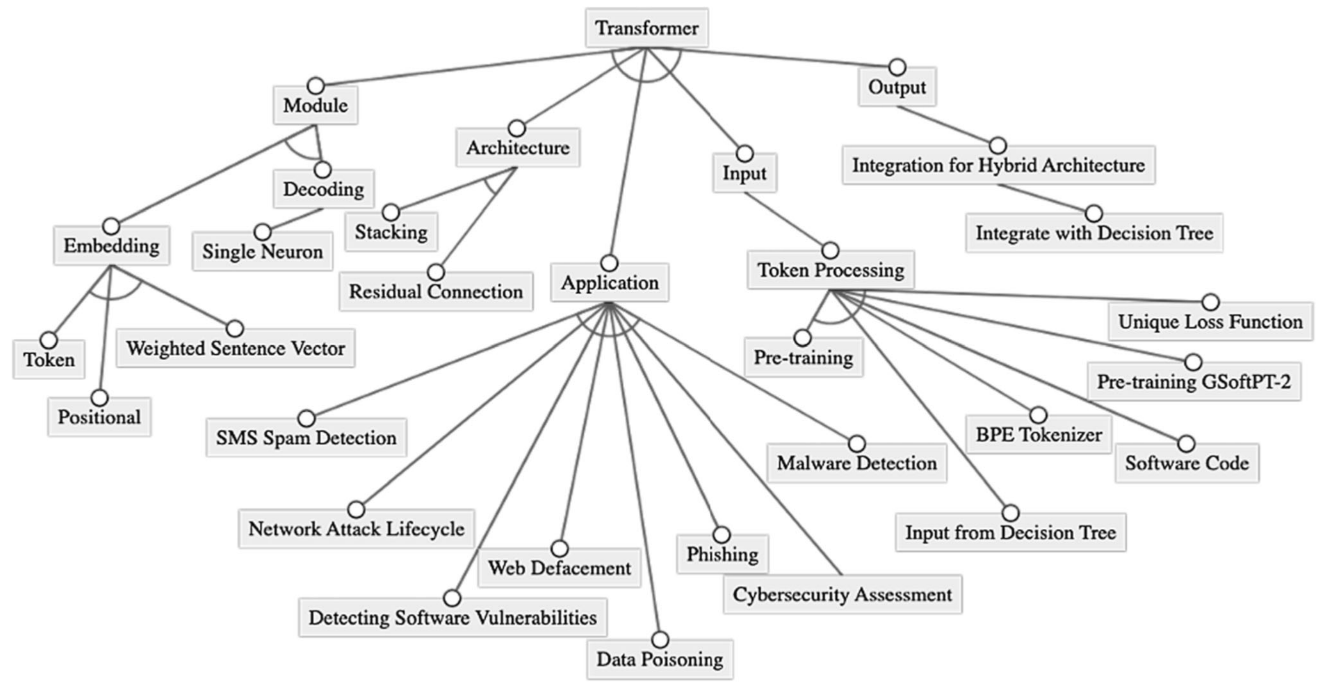


Fig. 2 Transformer for cybersecurity system feature model

according to [1] memory and recurrence can be discussed under alternative architectures. Besides, it should be noted that hybrid architectures can be formed by integrating transformers with other architectures at the input or output.

5.3 List of tactics

In this section, the design tactics for transformers are listed and discussed in detail. There are 14 identified transformer architecture design tactics in this study which are listed in forms using the tactic description template in Table 1 in the following sections, named as T-BERT, Pre-trained Transformer Model, Memory–list of trainable parameters, Pre-trained model GPT-2 with fine tuning, BPE tokenizers, An auxiliary auto-regressive loss function & balanced mixed objective training, Integrating decision tree and FT-transformer, Source code as a set of features, Network transformer (NeT), ClaimsBert, CuBERT (Code Understanding BERT), PELAT (Packet Embedding Method Based Language Model), Vision Transformer (ViT) and SecFed-Transformer (secure federated learning). While tactics such as BPE tokenizers and an auxiliary auto-regressive loss function & balanced mixed objective training are applied for phishing detection, there are other tactics in the known literature, one of which is Integrating decision tree and FT-transformer, that are applicable to the area of network attacks. The tactics are not only categorized according to the cybersecurity problem but also the quality factor that the tactic is implemented for is listed in a tactic description. The relevant quality factors to the subject are resilience, performance, completeness, interpretability, scalability, and accuracy.

5.3.1 T-BERT

See (Table 4).

Table 4 Tactic 1–T-BERT

Title	Description
Transformer Design Tactic Name	T-BERT
Cybersecurity problem	Web defacement detection
Quality factor	Resilience
Aim of Transformer Modification/ Adaptation	Improvement of the detection performance on a large scale
Design decision	To address the issue of illicit content modifying essential tags to gain a high ranking, a design decision is made to utilize tag-embedding and search engine indexing algorithms. The chosen approach focuses on the text and tags within content areas (such as web pages) that are not easily visible but still impact indexing. For this purpose, a multi-layer bidirectional transformer structure, specifically BERT (Bidirectional Encoder Representations from Transformers), is employed [29]. BERT is known for delivering state-of-the-art results in various natural language processing (NLP) tasks
Implementation	The implementation involves a tag embedding method, potentially based on HTML tags. An open-source BERT implementation with an encoder-based architecture is utilized. BERT employs token embedding for each word, along with segment and position embedding, to enhance the information in the sentence representation. By combining these three types of embedding, the sentence representation is created. For the tag-sentence pair, the tag embedding is concatenated with the sentence vector to generate the embedding. To represent the entire web page, an attention layer is applied
Design diagram	See Fig. 3. Tag Aware Hierarchical Network
Consequences	Implementing this tactic can result in improved search engine ranking and visibility, enhanced user experience, increased protection against content manipulation, more efficient indexing, potential computational overhead, and a dependency on the chosen BERT implementation. These consequences should be considered when implementing the tactic to evaluate its impact and trade-offs

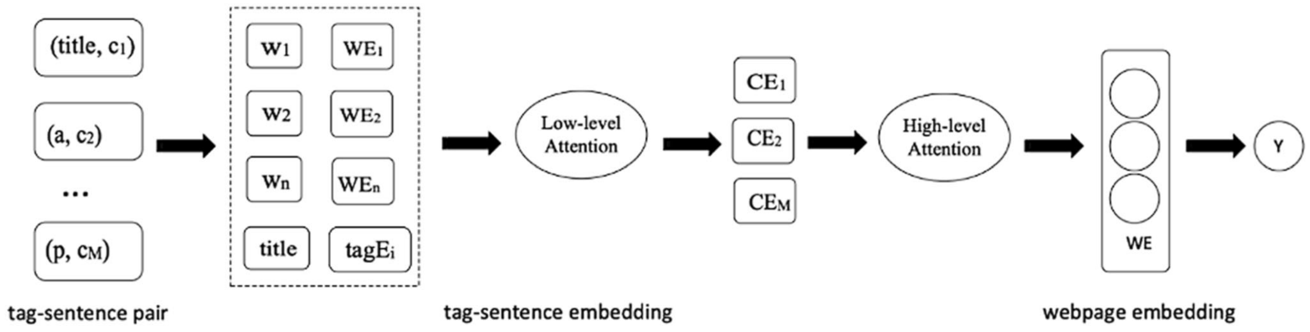


Fig. 3 Tag Aware Hierarchical Network

5.3.2 Pre-trained transformer model

See (Table 5).

Table 5 Tactic 2–Pre-trained Transformer Model

Title	Description
Transformer Design Tactic Name	Pre-trained transformer model
Cybersecurity problem	Malware or Phishing Detection
Quality factor	Performance
Aim of Transformer Modification/ Adaptation	Extending the architectures for more specialized applications
Design decision	There are many benefits of using pre-trained transformers to detect phishing websites by looking at their URLs. Software updates are simpler than feature-based systems since pre-processing of URLs is avoided, and they are safer to employ because phishing websites can be predicted without physically visiting the harmful sites. The model is deployable for real-time detection and requires little training time. It can also be utilized to run on portable devices
Implementation	The tactic offers the application of pre-trained transformer models available for immediate use. BERT and ELECTRA, having encoder-based architectures, are suitable for the implementation of this tactic
Design diagram	N/A
Consequences	Implementing this tactic can result in overfitting, increased protection against malware / phishing, decreased development and implementation duration and a dependency on the chosen model implementation. These consequences should be considered when implementing the tactic to evaluate its impact and trade-offs

5.3.3 Memory, list of trainable parameters

See (Tables 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 and 17).

Table 6 Tactic 3–Memory–list of trainable parameters

Title	Description
Transformer Design Tactic Name	Memory, list of trainable parameters [13]
Cybersecurity problem	SMS spam detection
Quality factor	Completeness
Aim of Transformer Modification/ Adaptation	Reducing the difficulty of feature extraction and data representation
Design decision	Instead of using the output sequence embedding, a list of trainable parameters named as memory is used. Since spam detection is a binary classification problem, the linear layer in the modified transformer model for SMS spam detection has a single neuron in the final layer rather than converting the output of the decoder stack to a vector
Implementation	A list of trainable parameters named as memory is used in the solution. Output embedding layer is excluded. Instead of a decoder stack, a single neuron is used for the final layer. Encoder-decoder transformer architecture is applied
Design diagram	Figure 4 A Spam Transformer Model for SMS Spam Detection
Consequences	Implementing this tactic can result in keeping relevant important information in memory such that the contribution to the attention is complete. The length of the memory is configurable which can have an effect on performance

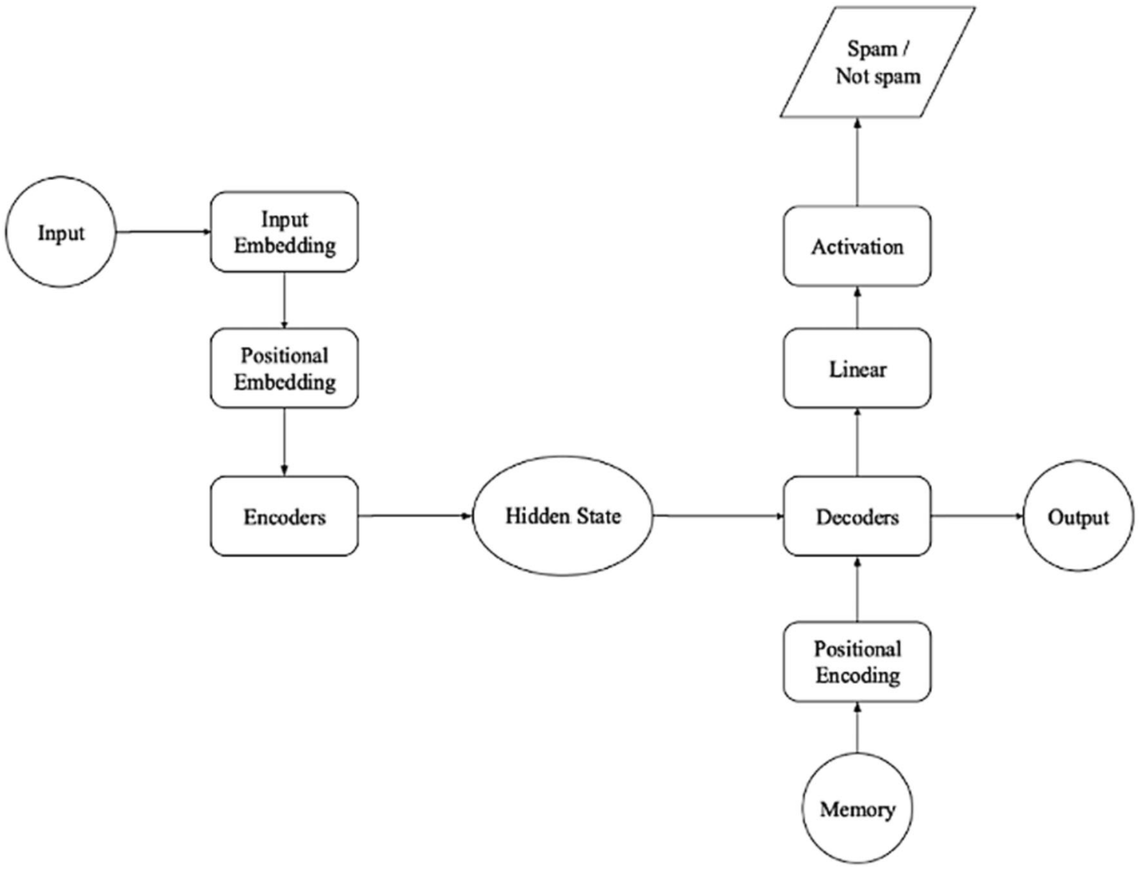


Fig. 4 A Spam Transformer Model for SMS Spam Detection

Table 7 Tactic 4–Pre-trained model GPT-2 with fine tuning

Title	Description
Transformer Design Tactic Name	Pre-trained model GPT-2 with fine tuning [14, 30]
Cybersecurity problem	Data poisoning attack
Quality factor	Resilience
Aim of Transformer Modification/ Adaptation	Text generation
Design decision	Pre-trained transformer-based language model GPT-2 for a general task using unlabeled data has an advantage of adaptability to new domains. Extending the architectures for more specialized applications may require fine-tuning the general pre-trained models
Implementation	The tactic proposes a model that has already been trained with pre-trained parameters rather than starting from scratch and initializing with random weights. GPT-2, having transformer decoder blocks, can serve as a base model for the solution
Design diagram	Figure 5. GPT-2 Architecture
Consequences	Implementing this tactic can result in overfitting, increased protection against data poisoning attack, decreased development and implementation duration and a dependency on the chosen model implementation. These consequences should be considered when implementing the tactic to evaluate its impact and trade-offs

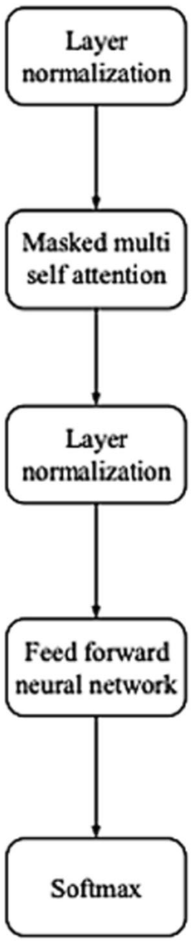


Fig. 5 GPT-2 Architecture

5.4 Integration of design tactics with broader theoretical frameworks

In [31] architectural tactics for big data analytics cybersecurity systems are introduced. Among the discussed architectural tactics, ML algorithm optimization applies to our case. While the selected quality factor is performance for the design tactic ML algorithm optimization, further mapping of the design tactics identified in our study with the architectural tactics in [31] on other quality factors can be accomplished.

A framework for the identification of the malicious edge device is presented in [32]. Intrusion detection is a core component of the system architecture. For the design of the intrusion detection components, Tactic 7 and Tactic 13 can be considered.

6 Case study

In this section, we have applied the proposed design method based on the design tactics for the transformer architectures in the previous sections, for a selected cybersecurity problem as a case study. In Sect. 5.1 we explain the application of the identified tactics for a case study on phishing detection. In Sect. 5.2 we discuss the implementation of the case study.

Table 8 Tactic 5–BPE tokenizers [15]

Title	Description
Transformer Design Tactic Name	BPE tokenizers [15]
Cybersecurity problem	Phishing Detection
Quality factor	Performance
Aim of Transformer Modification/ Adaptation	Phishing URL attacks can take place on short-lived domains and URLs that differ slightly from already-existing, trustworthy domains
Design decision	Performance of the model is improved when fewer layers and a domain specific vocabulary for short lived domains are used. Custom character level and byte-level BPE vocabularies are created
Implementation	Byte Pair Encoding (BPE) tokenizers are employed instead of extracting lexical features or CNNs kernels. Using the training URL data, custom character-level, and byte-level BPE vocabularies are developed, resulting in two different vocabulary sizes, 1K and 10K, for URLTran_CustVoc. The BPE models aim to strike a balance between the use of entire words and character subsequences. Encoder only BERT is used
Design diagram	N/A
Consequences	Implementing this tactic can result in overfitting, having a domain specific vocabulary decreases the flexibility of a system and causes a dependency on the created vocabulary. These consequences should be considered when implementing the tactic to evaluate its impact and trade-offs

Table 9 Tactic 6–An auxiliary auto-regressive loss function & balanced mixed objective training [16]

Title	Description
Transformer Design Tactic Name	An auxiliary auto-regressive loss function & balanced mixed objective training [16]
Cybersecurity problem	Phishing Detection
Quality factor	Resilience
Aim of Transformer Modification/ Adaptation	Improving training stability
Design decision	A classifier performs better and has more stable convergence characteristics when numerous correlated tasks are optimized simultaneously. A unique loss function is utilized for dynamically rebalancing gradients of auxiliary losses with the main task loss at each training step, aiming to increase training stability
Implementation	The tactic proposes to apply joint optimization across dataset X with labels Y for both next character prediction and malicious/beneficial classification. Regardless of the loss value, no one loss term dominates. Encoder only BERT is used
Design diagram	N/A
Consequences	Pre-training with an auto regressive loss does not improve performance. Performance is improved with auto regressive next character prediction loss

Table 10 Tactic 7–Integrating decision tree and FT-transformer [17]

Title	Description
Transformer Design Tactic Name	Integrating decision tree and FT-transformer [17]
Cybersecurity problem	Network attack lifecycle
Quality factor	Performance
Aim of Transformer Modification/ Adaptation	To lessen the negative effects of having only a little amount of high-quality labeled data on accuracy, less inaccurate predictions on the imbalanced intrusion datasets are needed
Design decision	The FT-Transformer, which generates embedding vectors for categorical and continuous features, is chosen. A common representation space is learned using a stack of transformers
Implementation	Initially, the binary classification of legitimate (normal) and malicious data is performed using the decision tree algorithm. The malicious data is then subjected to multi-category classification using FT-transformer to determine the nature of the attack. Encoder only BERT is used
Design diagram	N/A
Consequences	Due to the use of decision trees, disadvantages such as overfitting and instability can be observed when the design tactic is implemented

6.1 Case study description: transformer architectures for phishing detection

Phishing is a cyberattack technique in cyberspace where false actors or institutions imitate ones with websites, emails, hyperlinks, or other media aiming to deceive users to gather their information, such as usernames and passwords. Phishing detection is applied with software methods and techniques based on but not limited to, machine learning, deep learning, information retrieval, human users, and profile matching. This study demonstrates the applicability of transformer-based machine-learning model architectures proposed and evaluated for phishing website detection using URL properties as features. The choice of the phishing detection approach is influenced by quality characteristics like efficiency and accuracy as well as the software system architecture used for implementation. The case study provides insights for both practitioners and researchers for further research on transformer models and phishing detection. Phishing detection and mitigation techniques are widely studied, and new applications are implemented further in pace with the advances in machine learning, deep learning, information retrieval, human users, profile matching, and other relevant disciplines.

Phishing is a semantic attack targeting uneducated and naïve internet users rather than software applications and their bugs. The response rate of employees vulnerable to phishing attacks is approximately 20%. Therefore, considering the impact of phishing attacks and the vitality of having the appropriate system response, the study aims to

Table 11 Tactic 8–Source Code as a Set of Features [18]

Title	Description
Transformer Design Tactic Name	Source Code as a Set of Features [18]
Cybersecurity problem	Malware Detection
Quality factor	Performance
Aim of Transformer Modification/ Adaptation	Resources and time are saved because the malware doesn't need to be activated by running the code
Design decision	A novel feature representation can be achieved by examining the software applications' source code that could be considered as a collection of features. Text classification is conducted on feature set including information that comprises activities, intents, and permissions
Implementation	To characterize malware and classify it into several representative malware categories, a BERT (Bidirectional Encoder Representations from Transformers) model is proposed. It performs a static analysis on the source code of Android applications using preprocessed characteristics. The hugging face library is applied for binary classification in Transformers. Encoder only BERT is used
Design diagram	N/A
Consequences	Since the activation of malware is not required, using source code is less expensive in terms of resources and time

Table 12 Tactic 9–Network Transformer (NeT) [19]

Title	Description
Transformer Design Tactic Name	Tactic 9–Network Transformer (NeT) [19]
Cybersecurity problem	Network attack lifecycle
Quality factor	Interpretability
Aim of Transformer Modification/ Adaptation	To leverage our understanding of the system as a graph, communication network
Design decision	Graph features representing the computer network in a time window can be used. They provide a layered representation of the network in a clear format
Implementation	Hierarchical graph features with transformer encoder are utilized to obtain the encoded packet windows that are used to compute the graph features. Encoder-decoder transformer is used
Design diagram	Figure 6 Network Transformer Model
Consequences	Using semi-supervised learning the model does not need the labelled data (from raw values to global network graph representation). By means of the hierarchical design of network graph representation, predictions can be backtracked and analyzed in various granularity levels

research novel methods. Various phishing technical approaches are listed such as spear phishing, whaling, business email compromise, cross-site scripting, cross-site malicious CAPTCHA attack, QRishing, social engineering, drive-by download, malware, phishing, browser vulnerabilities, tab napping, typo squatting, sound squatting, 404 error manipulation, click jacking, malicious browsing extensions, man-in-the-middle, mobile phones, GUI-squatting, session fixation, JavaScript obfuscation. The primary motivation of this section is to analyze the performance of transformer-based architectures for phishing detection and discuss the results in terms of applicability considering several parameters.

For transformer architectures the defined design concerns in the previous sections are crucial and for each concern the corresponding tactic is defined. For the case of phishing detection, a subset of these tactics is important.

We started using the vanilla transformer architecture as the bases. As the commonly used architecture variant, we selected an encoder-based architecture. We have used the positional embedding as an architectural option from the feature diagram in Fig. 3. The transformer encoding and positional embedding implementation for video processing in [33, 34] is adapted to the phishing detection by enhancing the training data to a phishing dataset.

From Table 2, we can see that for phishing detection, performance, resilience, and optimization are important quality factors. The tactics that are listed for phishing detection are for performance Tactic 5 BPE tokenizers, for resilience Tactic 6 an auxiliary auto-regressive loss

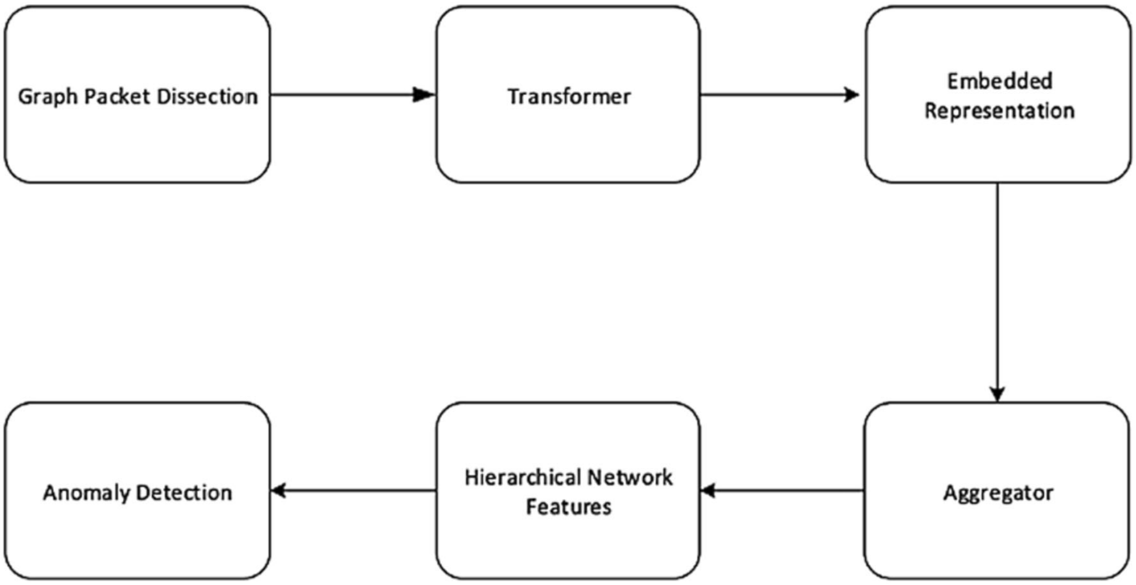


Fig. 6 Network transformer model

Table 13 Tactic-10 ClaimsBert

Title	Description
Transformer Design Tactic Name	Tactic 10: ClaimsBert [20]
Cybersecurity problem	Cybersecurity of industrial control systems, cybersecurity assessment
Quality factor	Accuracy, performance
Aim of Transformer Modification/ Adaptation	The human-readable documents are complicated for automation due to their formats and the natural language structure
Design decision	Cybersecurity claims in industrial control systems (ICS) device documents can be automatically identified by a framework using a classification model
Implementation	In this tactic, informative characteristics from transformers are extracted and then assigned to a classifier using a feature map created by combining a pre-trained BERT language model with a convolution neural network (CNN). BERT is used as an encoder only model
Design diagram	N/A
Consequences	The model uses the feature map generated via CNN which has small dimensionality so that the features are efficiently understood by the model

Table 14 Tactic 11- CuBERT (Code Understanding BERT)

Title	Description
Transformer Design Tactic Name	Tactic 11: CuBERT (Code Understanding BERT) [21]
Cybersecurity problem	Detecting Software Vulnerabilities
Quality factor	Classification completeness
Aim of Transformer Modification/ Adaptation	To determine the models' robustness regarding how some of the sub-concepts could result in classification errors, the classification behavior should be compared to the defined sub-concepts and input source code attribute
Design decision	CuBert, a derivation of BERT, adjusted for processing source code instead of natural language, is applied
Implementation	The tactic utilizes CuBERT tokenizer, multi-headed pointer model, pre-trained models, and pre-training corpora is used. Encoder only BERT is used
Design diagram	N/A
Consequences	With shorter training and fewer labeled examples, the model has high performance

Table 15 Tactic–12 PELAT (Packet Embedding Method Based Language Model)

Title	Description
Transformer Design Tactic Name	Tactic 12: PELAT (Packet Embedding Method Based Language Model) [22]
Cybersecurity problem	Network attack lifecycle
Quality factor	Completeness
Aim of Transformer Modification/ Adaptation	Enhance the attack knowledge, reduce manual labelling burden
Design decision	Semi-supervised learning is performed to generate the tactic labels of the unlabeled packets
Implementation	After the transformer encoder layers, a classification layer is implemented. BERT is used
Design diagram	N/A
Consequences	Benefiting from the knowledge in the source domain, the effect of learning in the target domain is improved and optimized by transfer learning

Table 16 Tactic 13–Vision Transformer (ViT)

Title	Description
Transformer Design Tactic Name	Tactic 13: Vision Transformer (ViT) [23]
Cybersecurity problem	Network attack lifecycle
Quality factor	Performance, Scalability
Aim of Transformer Modification/ Adaptation	To process a network flow pattern within a specific time interval that is converted into a two-dimensional image
Design decision	Self-attention layers are used to capture long-term dependencies. Transformer asynchronously learns diverse interactions between spatial locations and input
Implementation	Classification is achieved by using transformer encoder with a Multiple Layer Perceptron (MLP) head. For position embedding, the flattened patches of the image are linearly projected
Design diagram	N/A
Consequences	The multi-class classification accuracy and feature selection is impacted negatively by the data imbalance since some attack type data is not enough. When the data in a category is not enough, features are not selected, and the method shows low performance

function & balanced mixed objective training and for optimization, Tactic 2 pre-trained transformer models available for immediate use.

However, for the specific case of this research, certain tactics may not be applicable. Instead of using a pre-trained transformer model (Tactic 2), a custom transformer architecture is designed and trained on selected datasets and measure the performance results. Additionally, the use of domain specific vocabulary (Tactic 5) is also not applicable as the datasets are not using domain specific vocabulary instead, they consist of numerical values. Lastly, Tactic 6 is used for the label prediction of the training data in [16]. The data we use is already labelled therefore this part of the tactic 6 is not applicable for our case.

6.2 Implementation

We aim to further apply the defined design tactics using an experimental setup and discuss the results afterwards. The vanilla transformer adaptations that are relevant for the phishing detection problem are elaborated to leverage the design process and consolidate the selected design tactics with the solution in the application development process.

To build the phishing detection models and conduct our experiments, we followed the steps below:

- *Setting up the development environment*

We have selected the TensorFlow transformer which consists of a set of libraries in the area of natural language processing. The implementation is for general usage and can have efficient implementations with PyTorch. In order to develop our transformer model, we use Jupiter Notebooks as the development environment, which provides terminal, text editor and directory within one view. The Jupyter notebook is used for data science and the code is inside the notebook. The drawbacks of the environment are that it does not provide a versioning system and does not support distribution or live collaboration.

Table 17 Tactic 14–SecFed-Transformer (secure federated learning)

Title	Description
Transformer Design Tactic Name	Tactic 14: SecFed-Transformer (secure federated learning) [24, 25]
Cybersecurity problem	Network attack lifecycle
Quality factor	Completeness Resilience
Aim of Transformer Modification/ Adaptation	Must thoroughly examine the relationship between various electrical quantities to discover FDIA with varying intensities. Detection of FDIA based on transformers. Transformer-Based Distributed Detector
Design decision	Transformer is utilized as a detector in the edge nodes. The multi-head self-attention mechanism enables examination of the relationship between electrical quantities in detail. Instead of relying on the original central workstation, the installed edge node detector at each node of the power system directly collects, stores, and detects data
Implementation	The transformer architecture's encoder, multi-head self-attention, add&norm, and feed forward features are employed. The measurement vector z, which includes power injections and power flows, serves as the inputs of the transformer-based detection model, which is labeled with = 1 or = 0. The transformer model is also trained to thoroughly extract attack features from both legitimate and compromised data, allowing it to identify the presence of covert (false data injection attack) FDIA
Design diagram	N/A
Consequences	Data is collected from all nodes to train the model and the data is stored locally such that the data privacy is preserved

Table 18 Detailed information on datasets

Dataset	Details
Datasets for Phishing Websites Detection [26]	The features reported in the datasets were taken from publicly available lists of authentic and phishing websites
	The data format was a CSV file. Only the phishing websites listed in the PhishTank directory, which have been confirmed by numerous users, were included. We included the websites from publicly accessible, community-labeled, and sorted lists and from the top-ranking Alexa websites as the legitimate websites. The features in the data were taken from collections of website addresses. There are 111 features in the data overall, 96 of which are taken directly from the website address, while the remaining 15 features were extracted using custom Python code
Phishing Dataset for Machine Learning [27]	This dataset, which was downloaded between January and May 2015 and May and June 2017, has 48 features that were taken from 5000 authentic websites and 5000 fraudulent websites. By utilizing the Selenium WebDriver browser automation framework, a better feature extraction method is used that is more accurate and reliable than the parsing method based on regular expressions. This dataset may be helpful for phishing features analysis, quick proof of concept tests, or benchmarking phishing classification models for anti-phishing researchers and experts

- *Datasets*

Table 18 provides information regarding the datasets. For the dataset, the data is composed in a text file as each line represents API calls consisting of integers as comma-separated values. The data is used as is.

- *Implementation of transformer-based models*

The pseudo code of the transformer model implementation is defined in the following steps:

1. Positional embedding and transformer encoder are used as is, as stated in [8, 9]. The pseudocode is provided below:

```
class PositionalEmbedding:
  method: initialize
          initialize properties
          position_embeddings := Embedding with inputs as input_dimention as
                    sequence_length, output_dimention
  method: call self
          embedded_positions := self.position_embeddings with input as positions
      return inputs + embedded_position
  method: compute_mask:
      mask := tensorflow.reduce_any with inputs as inputs caste“ to ”bool", -1
      return mask
```

```
class TransformerEncoder:
   method: initialize
        initialize properties
        .attention := layers.MultiHeadAttention with inputs as
            number of heads, embedding dimentions, dropout as 0.3
        dense_projection := keras.Sequential with inputs as
            layers.Dense with inputs as dense_dimention, Gaussian Error Linear Unit
                          (GELU) activation function, layers.Dense with input as
                         embedding_dimention
        layernormalization1 := normalization layer
        layernormalization2 := normalization layer
   method: call
        if mask is not None:
            mask := mask[:, tensorflow.new axis, :]

   attention_output := attention with inputs as inputs, attention_mask
   projection_input := layernorm1 with inputs as inputs, attention_output
   projection_output := dense_projection with input as projection_input
   return layernormalization2 with inputs as projection_input, projection_output
```

2. A functional model with sequential layers is initiated with positional embedding and transformer encoder [8]:

```
positional_embedding := PositionalEmbedding(
 sequence_len25mbeddingeding_dimention), stacked with inputs

transformer_encoder := TransformerEncoder with inputs as embedding_dimention,
        dense_dimention, number of heads, stacked with positional_embedding

layer := dropout layer as 0.5, stacked with transformer_encoder
outputs :=  dense layer as 1 unit, sigmoid activation function, stacked with layer
model := keras.Model(inputs, outputs)
compile model as adam optimizer, binary crossentropy loss, using metrics as accuracy,
        f1, precision, recall
```

3. During training, the dropout layer initializes the input units with 0 at a frequency equal to the rate given for each step.
4. The model is enhanced with the addition of a dense layer with a single output unit and “the “si”moid” activation function (between 0 and 1, wi “h”a “S”-shaped curve).
5. The inputs and outputs of the model are created.
6. The model updates its properties (weights, learning rate, etc.) for the loss reduction using the optimizer. As

the optimizer, the stochastic gradient me “hod”adam“ is utilized.

7. For classification with two categories, the constructed model employs “adam” as the optimizer and “binary_crossentropy” as the loss function.
8. The metrics to be gathered “are ”acc”ra “y“, ”f1”core“ (weighted average of the precision and reca “l), ”preci”ion,“ “and ”r”call“.

6.3 Application of design tactics for transformer model adaptation

In order to further adapt our model for phishing datasets we applied design tactics suitable for phishing datasets, including Tactic 3 memory and Tactic 13 vision transformer. The associated quality factors for these tactics are completeness and performance.

Tactic 3, originally applied for malware classification, is applicable to our problem as both problems use binary classification. Therefore, to adapt the vanilla transformer using this tactic, the output embedding layer is excluded and a single neuron is used instead of the decoder having the inspiration from replacing the linear layer with a single neuron. Although the justification of usage of memory adaptation is not explained in [13], we infer that the application of memory adaptation in this tactic serves a similar purpose to the LSTM memory concept that aims to effectively capture and remember information in long sequences of data. However, our dataset consists of numerical values, therefore, we do not have a long sequence of data in our dataset. Considering the quality factor completeness for the Tactic 3, an applicable criterion can be defined as describing high level features in terms of low-level features such that information related to the lower-level features that form the high-level features are captured in memory. In our case, the features are not grouped to higher level features. Consequently, we did not use the memory matrix adaptation.

For Tactic 13, in Vision Transformer, classification is achieved by employing a transformer encoder with a Multiple Layer Perceptron (MLP) head and the flattened patches of the image are linearly projected for position embedding. The quality factors for the tactic are performance and scalability. In the scope of the implementation, we consider performance as the quality factor. In our case, we used the video transformer architecture. The criterion on data type is alpha-numerical. Consequently, excluding the vision adaptation from the positional encoding, and we applied the same transformer encoder implementation of VIT for phishing detection to conduct an experiment on the applicability of the defined tactic to the described phishing detection problem.

Table 19 Classification report using Transformers for the dataset 1

	Precision	Recall	f1-score	Support
0	0.96	0.96	0.96	14,484
1	0.92	0.93	0.93	7676
Accuracy			0.95	22,160
Macro avg	0.94	0.95	0.94	22,160
Weighted avg	0.95	0.95	0.95	22,160

6.4 Results

Table 3 displays the outcomes of the use of transformer models for phishing detection on dataset 1. 100 epochs are run with the models. The optimizer ”adam“ employs the stochastic gradient descent technique, and the loss function is ”binary_crossentropy (cross-entropy loss between true labels and predicted labels)“. Table 19 summarizes the results of applying transformers for phishing detection on the dataset 1, using the classification_report. Figure 7 presents the results of the application of the transformer-based model on dataset in terms of the AUC (area under the ROC curve) parameter. Figure 8 presents model accuracy.

The results of the application of transformers for phishing detection on the dataset 2 are listed in Table 20. Figure 9 illustrates the results of the application of the transformer-based model on dataset measured in terms of the AUC (area under the ROC curve) parameter (Fig. 10).

The method separates phishing data from other entries and categorizes the chosen phishing data as phishing. The phishing type is identified using binary classification, and the model can be improved to simultaneously identify numerous phishing types if necessary. The tests were carried out on a macOS computer with an 8 GB memory and a 1.4 GHz quad-core Intel Core I5 processor.

A sequential model with one input and one output is the basis upon which the Keras functional model is built. The model may include one or more layers, depending on the data structure. Regular neural network layer with dense connections makes up the dense layer. During network training, the dropout strategy is used to remove certain neurons from the layer. For transformer-based implementation, a transformer encoder is used, with parameters such as embedding dimension, dense dimension, and the number of attention heads. In addition, positional encoding is implemented to consider the order information.

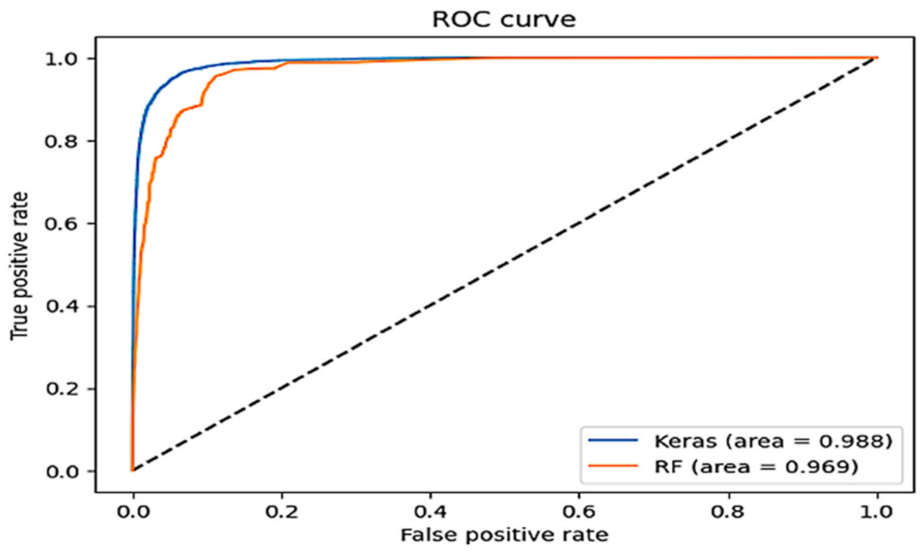


Fig. 7 ROC for dataset 1

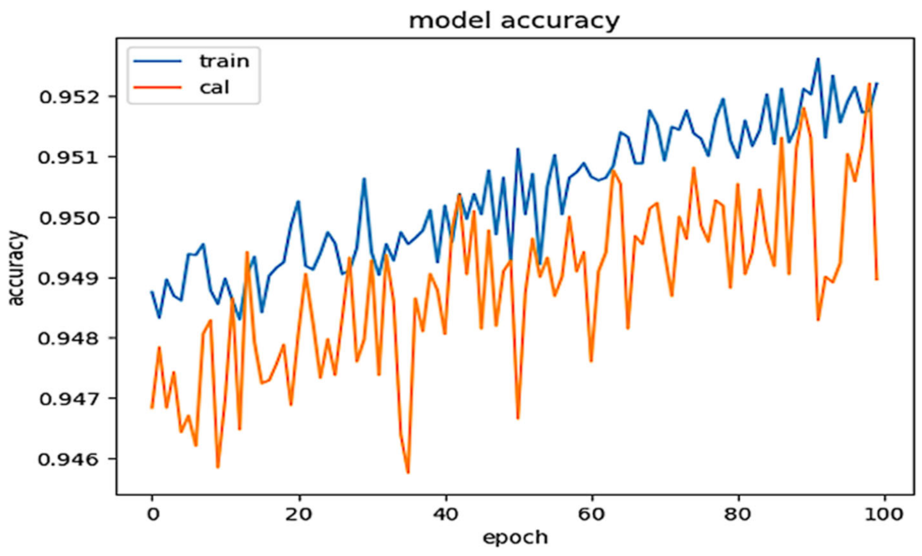


Fig. 8 Transformer-based model accuracy for the dataset 1

Table 20 Classification report using Transformers for the dataset 2

	Precision	Recall	f1-score	Support
0	1.00	0.99	1.00	1249
1	0.99	1.00	1.00	1250
Accuracy			1.00	2499
Macro avg	1.00	1.00	1.00	2499
Weighted avg	1.00	1.00	1.00	2499

7 Discussion

Using two separate data sets, we performed a performance analysis of the transformer-based models for the identification of phishing data in the preceding section, where we also highlighted the use of transformers in cybersecurity. We may now respond to our original research questions and elaborate based on this analysis.

What are the design tactics for cybersecurity architectures based on transformers?

Section 4 lists design tactics for transformer-based cybersecurity solutions. Although new tactics can be introduced in the upcoming times, we have formed a tactic base for this study by means of the tactics present in the known literature. There are 14 design tactics that are listed in Table 2. The range of the tactics start from pre-training the transformer model to implementing a memory for transformers. There are also domain specific transformer architectures such as vision transformer or network transformer.

How well do transformer-based models for phishing prediction work?

Fig. 9 Transformer-based model accuracy for the dataset 2

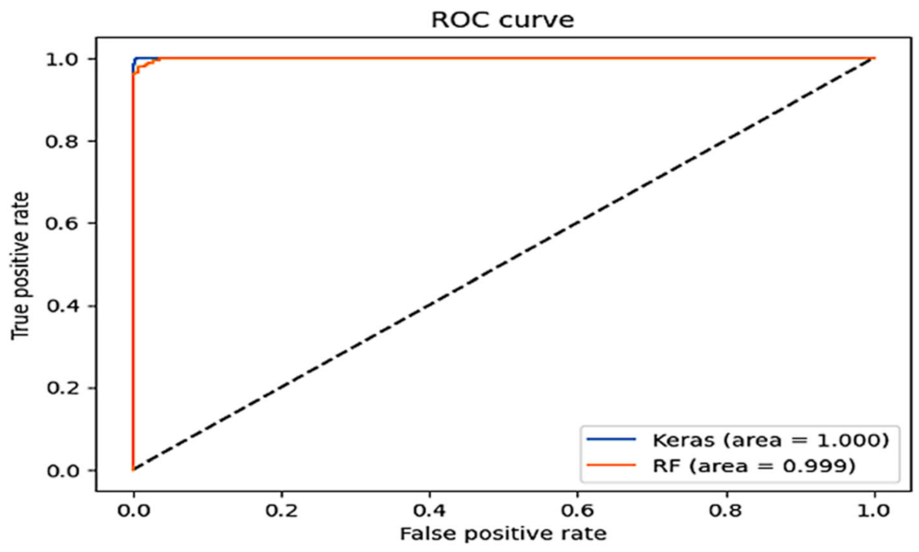
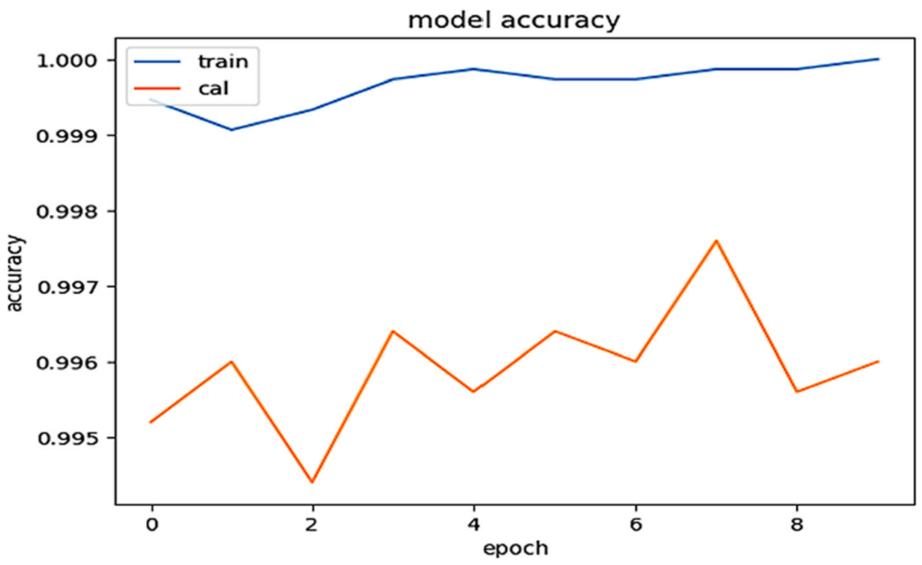


Fig. 10 ROC for dataset 2



The trials showed that transformer-based models had a dataset accuracy of about 95%. The maximum f1 score for the dataset is 0.96, which denotes successful prediction and recall. When the curve in the ROC graph is more closely aligned with the diagonal, it is assumed that the accuracy of the test is low. However, the results are within an acceptable range, and we can conclude that transformer-based models are indeed feasible and effective for cybersecurity applications.

Designing a transformer model, initially, vanilla transformer model is considered as a solution. While this is a valid approach, it is not mandatory to use the architecture as is. The architectural overhead of overdesign can introduce further complexity in the system that weakens the capability of meeting the quality requirements.

As we can see in the feature model presented in Fig. 3, optional design choices and components are available and enables the designer to apply a flexible design strategy by tailoring the feature model to a model instance leading to the application architecture. In this feature model, the variation points are shown.

Is using the design tactics for transformers effective?

The aim of studying the design tactics for transformers is to form a structured methodology for the design of transformer architectures. Although there are various studies in the known literature supporting transformer architecture design, many implementations are based on architectures that are designed by an experimental approach. In this study, we aimed to form a guideline for transformer architecture design, presenting a design tactic catalog that enables the architect to select among the design options to obtain the desired result. For the selection process of the relevant design tactic, we constructed a feature diagram that has optional features to be tailored for the target application architecture. The tailored feature diagram maps to a defined design tactic which shapes the design strategy of the transformer architecture. The overall approach is a product line strategy which aims to design an application architecture from a reference architecture. The application of the design method is shown with a case in Sect. 5. The detailed explanation of the application of the method shows that it is applicable to the target problem and using the proposed method with the design tactics is effective for the target application area.

Are the identified design tactics effective?

The effectiveness of the identified design tactics could be validated because the tactics are derived from the studies from the known literature such that their implementation and results are already published to the community. Furthermore, the applicability of the design tactic 3 and design tactic 13 are discussed in Sect. 5.1.5. As a result, while selected design tactics are applicable and effective in the defined problem, their implementation requires their adaptation to the target implementation.

What are the limitations and challenges?

The limitations and challenges for the proposed design methodology of using design tactics for transformer architectures in cybersecurity problems can be discussed from the aspects of computational demands, data privacy concerns, and adaptability to rapidly evolving threats. In [31], the efficiency of the transformers is discussed. Deploying the transformer models efficiently in data centers equipped with GPU is challenging due to the constraints of latency and throughput. Furthermore, transformer model input dimensions can vary leading to difficulties in optimizing services and managing the memory. Another aspect is the data privacy. Before the training or testing process with transformer models, the data used shall be classified into the categories such as sensitive, public, or private such that the data privacy requirements are met with the relevant implementation (f.e. encryption). Another challenge is that due to the evolving threats, the design tactics and the transformer model implementations shall be regularly updated.

What are the threats to validity?

This study is quantitative research and the threats to validity are considered based on [35]. The methods used in this research are experimentation and data analytics. The categories of threats to validity are listed as internal and external. The validity categories in [35] are conclusion validity, internal validity, construct validity, external validity. The limitations of the study of while reaching to conclusions from the independent and dependent variable relations are threats to conclusion validity. The independent variable in this study is the feature diagram. The feature diagram is formed using the selected studies from the literature. Further relevant features could be added to the feature diagram to extend its context as other features emerge in the literature such that the validity of the method could reach broader coverage within the area of cybersecurity problems. The dependent variables that are impacted by the extendibility of the feature diagram are design tactics catalog, since the catalog can also be updated with design tactics including the extended features of the feature diagram. The conclusion is affected due to the use of the design tactics in the transformer application architecture. According to the threats to internal validity, the change in design tactics is expected to be solely caused by the changes in feature diagram. Although the formation of the design tactics is based on the derived feature diagrams of the reference feature diagram, we cannot say that the design tactic is fully represented by the derived feature diagram. However, the base structure of the transformer application architecture can be mapped to the corresponding features in the relevant feature diagram. For the construct validity, the theoretical concept is accurately explained using the feature diagram and the design tactics. Finally, to achieve the external validity, the application of the method is discussed and the results of a transformer application architecture are presented using a software implementation in Sect. 5. The applicability and adaptability of the selected design tactics is discussed and justified in Sect. 5.1.5.1. At the implementation stage of the case study, the practitioners shall be aware of the fact that as the style of adaptation aligned with the design tactic and the dataset changes, the measures for the effected quality metric can vary due to the nature of the problem and the solution.

8 Related work

As a result of using a technique while designing an architecture, the quality attribute model and the design are correlated. To achieve the desired quality measures, the parameters of the quality model driving the design shall be described including the inputs, the characteristics of the model elements and the independent variables. The architectural choices also drive the implementation for the desired architectural response [36].

Transformers and their modifications are used for cybersecurity applications in various studies. The details regarding the description, technology, and architectures of the transformer implementations are listed in Table 21.

Study [39] discusses transformer modifications. The transformer modifications mentioned in [39] are lightweight transformers, connections between transformer blocks, adaptive computation time, recurrence relations between transformer blocks, hierarchical transformers, transformers with modified multi-head self-attention

Table 21 Classification report using Transformers for the dataset 2

Study	Details related to transformers
Detecting Cybersecurity Attacks in Internet of Things Using	The study indicates that transformer-based models are efficient for misinformation detection
Artificial Intelligence Methods: A Systematic Literature Review [37]	
End-to-End Transformer-Based Models in Textual-Based NLP [38]	Transformers are used in a wide range of applications for sequence modelling
Transformers for Machine Learning: A Deep Dive [39]	Transformer modifications: Lightweight Transformers, connections between transformer blocks, adaptive computation time, recurrence relations between transformer blocks, hierarchical transformers
	Transformers with modified multi-head self-attention: Structure of Multi-head self-attention, reducing the complexity of self-attention, improving multi-head-attention, biasing attention with priors, prototype queries, compressed key-value memory, low-rank approximations
	Modifications for training task efficiency: ELECTRA, T5
Do Transformer Modifications Transfer Across Implementations and Applications? [40]	The study discusses the reason of adoption of modifications proposed to the transformer
A survey of transformers [1]	The study investigates the architecture modifications

(structure of multi-head self-attention), reducing the complexity of self-attention, improving multi-head-attention, biasing attention with priors, prototype queries, compressed key-value memory, low-rank approximations, modifications for training task efficiency (ELECTRA, T5).

9 Future research directions

The technologies are evolving, and threats are emerging in the area of cybersecurity. Therefore, the necessity of updating and extending the list of design tactics formed in this study can be beneficial. The tactics can be further discussed for other domains such as cybersecurity for distributed systems or IoT or the scalability of the tactics for intensive data can be an area of research. The listed tactics can be integrated with broader cybersecurity frameworks and guidelines. Examples for integration with broader frameworks are provided in Sect. 5.4.

Studies for comparative analysis of other cybersecurity architectures or methodologies can be implemented to justify the choice of the transformer architectures. Comparative analysis of the presented design tactics on selected problems can also be an area of future research. Such studies can also analyze unique benefits or challenges of the design tactics for the selected problem domain and context. To develop design guidelines specific for the selected cybersecurity problem, case studies can be conducted to gather empirical evidence. Formation of such design guidelines can strengthen the arguments for selection of the design tactic for the application.

10 Conclusion

In this study, transformer-based models for cybersecurity architectures are discussed, and the design tactics which aim to support architectural design decisions that can affect the design and implementation of a cybersecurity system are listed. Following a systematic research methodology, relevant information on the subject and its stakeholders are gathered, appropriate datasets are chosen, current empirical approaches are applied, and the experiment design and outcomes are presented. The selected configuration among several transformer-based cybersecurity systems is used as a case study for phishing detection. The experimental technique and the applied methodology are finally given and thoroughly analyzed, offering useful insights for this field’s practitioners.

We have described 14 design tactics for transformer-based architectures. While all of the design tactics are equally important, we can list a few of them as Packet Embedding Method Based Language Model [22], Code Understanding BERT [21] and Vision Transformers [23], named after the corresponding implementations. The quality factors related to these design tactics are not limited to but include performance and completeness. While the description of the design tactic can be initiated by tailoring the feature model, among the defined design tactics, Tactic 13 Vision Transformer and Tactic 3 Memory, list of trainable parameters is discussed in the case study for phishing detection. It is observed that although a tactic is defined for a particular cybersecurity problem, it can also be applied for another cybersecurity problem.

Throughout this study, we understood that the design process of transformer models can be leveraged with the design tactics so that we can infer the modifications and adaptations with their justifications based on the quality factors of the problem under discussion and implement a solid solution accordingly. The experiments are conducted for the phishing cybersecurity problem, and we observed that the solutions based on transformer models are applicable for phishing. Moreover, the quality factors that are relevant for the cybersecurity problems are resilience, completeness, performance, optimization, accuracy, and interpretability. Finally, the adaptations and modifications are mostly applied on embedding, input, and decoder transformer layers.

Author contributions CA: Conceptualization, Methodology, Software, Writing-Original Draft, Writing-Review & Editing. BT: Conceptualization, Methodology, Software, Writing-Original Draft, Writing-Review & Editing, Supervision. CC: Conceptualization, Methodology, Software, Writing-Original Draft, Writing-Review & Editing, Supervision.

Funding The authors did not receive support from any organization for the submitted work.

Data availability Data availability is not applicable to this article as no new data were created or analysed in this study.

Declarations

Conflict of interest The authors have not disclosed any competing interests.

References

1. Lin, T., Wang, Y., Liu, X., Qiu, X.: A survey of transformers. AI Open. **3**, 111–132 (2022)
2. Dellarocas, C.: A coordination perspective on software system design. In: Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering, pp. 318–325. (1997)
3. Evans, E.: Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional, Boston (2004)
4. Tekinerdogan, B., Verdouw, C.: Systems architecture design pattern catalog for developing digital twins. Sensors **20**(18), 5103 (2020)
5. Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Sun, L.: A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. arXiv preprint https://arxiv.org/abs/2302.09419. (2023)
6. Cruzes, D.S., Ben Othmane, L.: Threats to validity in empirical software security research. In: Empirical research for software security, pp. 275–300. CRC Press, Boca Raton, FL (2017)
7. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: a survey. ACM comput. Surv. (CSUR) **54**(10s), 1–41 (2022)
8. Firesmith, D.: Using quality models to engineer quality requirements. J. Object Technol. **2**(5), 67–75 (2003)
9. https://github.com/acmsigsoft/EmpiricalStandards/tree/master/docs
10. Ullah, F., Babar, M.A.: Architectural tactics for big data cybersecurity analytics systems: a review. J. Syst. Softw. **151**, 81–118 (2019)
11. Yang, R., Wang, X., Chi, C., Wang, D., He, J., Pang, S., Lau, W.C.: Scalable detection of promotional website defacements in black hat {SEO} campaigns. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 3703–3720 (2021)
12. Haynes, K., Shirazi, H., Ray, I.: Lightweight URL-based phishing detection using natural language processing transformers for mobile devices. Procedia Comput. Sci. **191**, 127–134 (2021)
13. Liu, X., Lu, H., Nayak, A.: A spam transformer model for SMS spam detection. IEEE Access **9**, 80253–80263 (2021)
14. Ranade, P., Piplai, A., Mittal, S., Joshi, A., Finin, T.: Generating fake cyber threat intelligence using transformer-based models. In: 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1–9). IEEE. (2021)
15. Maneriker, P., Stokes, J.W., Lazo, E.G., Carutasu, D., Tajaddodianfar, F., Gururajan, A.: URLTran: Improving Phishing URL Detection Using Transformers. MILCOM 2021–2021 IEEE Military Communications Conference (MILCOM), pp. 197–204. IEEE. (2021)
16. Rudd, E.M., Abdallah, A.: Training Transformers for Information Security Tasks: A Case Study on Malicious URL Prediction. arXiv preprint https://arxiv.org/abs/2011.03040 (2020)
17. Lan, Y., Truong-Huu, T., Wu, J., Teo, S.G. Cascaded multi-class network intrusion detection with decision tree and self-attentive model. In: 2022 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 1–7. IEEE (2022)
18. Rahali, A., Akhloufi, M.A.: MalBERT: Using transformers for cybersecurity and malicious software detection. arXiv preprint https://arxiv.org/abs/2103.03806. (2021)
19. Marino, D.L., Wickramasinghe, C.S., Rieger, C., Manic, M.: Self-supervised and interpretable anomaly detection using network transformers. arXiv preprint https://arxiv.org/abs/2202.12997. (2022)
20. Ameri, K., Hempel, M., Sharif, H., Lopez, J., Jr., Perumalla, K.: An accuracy-maximization approach for claims classifiers in document content analytics for cybersecurity. J. Cybersecur. Pri. **2**(2), 418–443 (2022)
21. Kanade, A., Maniatis, P., Balakrishnan, G., Shi, K.: Learning and evaluating contextual embedding of source code. In International conference on machine learning, pp. 5110–5121. PMLR (2020)
22. Lin, L.H., Hsiao, S.W.: Attack tactic identification by transfer learning of language model. arXiv preprint https://arxiv.org/abs/2209.00263. (2022)

23. Ho, C.M.K., Yow, K.C., Zhu, Z., Aravamuthan, S.: Network intrusion detection via flow-to-image conversion and vision transformer classification. IEEE Access **10**, 97780–97793 (2022)
24. Li, Y., Wei, X., Li, Y., Dong, Z., Shahidehpour, M.: Detection of false data injection attacks in smart grid: a secure federated deep learning approach. IEEE Trans. Smart Grid **13**(6), 4862–4872 (2022)
25. Baul, A., Sarker, G.C., Sadhu, P.K., Yanambaka, V.P., Abdelgawad, A.: XTM: a novel transformer and LSTM-based model for detection and localization of formally verified FDI attack in smart grid. Electronics **12**(4), 797 (2023)
26. Vrbančič, G., Fister, I., Jr., Podgorelec, V.: Datasets for phishing websites detection. Data Brief **33**, 106438 (2020)
27. https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning?resource=download
28. Tay, Y., Dehghani, M., Bahri, D., Metzler, D.: Efficient transformers: a survey. ACM Comput. Surv. **55**(6), 1–28 (2022)
29. Buccella, A., Cechich, A., Porfiri, J., Diniz Dos Santos, D.: Taxonomy-oriented domain analysis of GIS: a case study for paleontological software systems. ISPRS Int. J. Geo Inf. **8**(6), 270 (2019)
30. Ranade, P., Joshi, A., Finin, T.: Study shows AI-generated fake cybersecurity reports fool experts. Conversation. (2021)
31. Fang, J., Yu, Y., Zhao, C., Zhou, J.: Turbotransformers: an efficient gpu serving system for transformer models. In: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 389–402. (2021)
32. Sohal, A.S., Sandhu, R., Sood, S.K., Chang, V.: A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. Comput. Secur. **74**, 340–354 (2018)
33. https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/vision/ipynb/video_transformers.ipynb
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Polosukhin, I.: Attention is all you need. Adv. Neural Inf. Process. Syst., **30** (2017)
35. Bass, L., Clements, P., Kazman, R.: Software architecture in practice. Addison-Wesley Professional, Boston (2003)
36. Vairo, T., Lecca, M., Trovatore, E., Reverberi, A., Fabiano, B.: A Bayesian Belief Network for Local Air Quality Forecasting. Chem. Eng. Trans. **74**, 271–276 (2019). https://doi.org/10.3303/CET1974046
37. Abdullahi, M., Baashar, Y., Alhussian, H., Alwadain, A., Aziz, N., Capretz, L.F., Abdulkadir, S.J.: Detecting cybersecurity attacks in internet of things using artificial intelligence methods: a systematic literature review. Electronics **11**(2), 198 (2022)
38. Rahali, A., Akhloufi, M.A.: End-to-end transformer-based models in textual-based NLP. AI **4**(1), 54–110 (2023)
39. Kamath, U., Graham, K.L., Emara, W.: Transformers for Machine Learning: A Deep Dive. CRC Press, Boca Raton, FL (2022)
40. Narang, S., Chung, H. W., Tay, Y., Fedus, W., Fevry, T., Matena, M., Raffel, C.: Do transformer modifications transfer across implementations and applications?. arXiv preprint https://arxiv.org/abs/2102.11972 (2021)
41. Bachmann, F., Bass, L., Klein, M.: Deriving architectural tactics: a step toward methodical architectural design. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst. (2003)

Cigdem Avci holds a BSc and an MSc in Computer Engineering from Middle East Technical University, and a PhD from Wageningen University in The Netherlands. She worked for European Space Agency, Turkish Aerospace and other organizations. She is a lecturer at Middle East Technical University. Her research focuses on system design, cybersecurity, machine learning, and software engineering.

Bedir Tekinerdogan holds the position of full professor and chair of the Information Technology group, as well as the chair of the Business Section, which encompasses five distinct chairgroups, at Wageningen University in The Netherlands. He has over 25 years of experience in software/systems engineering and information technology. He earned both his MSc degree (1994) and a PhD degree (2000) in Computer Science from the University of Twente, The Netherlands. He was a faculty member at the University of Twente from 2003 to 2008 before joining Bilkent University, where he worked until 2015. At Bilkent University, he founded and led the Bilkent Software Engineering Group to promote software engineering research and education in Turkey. Throughout his career, he has authored more than 400 peer-reviewed scientific papers and 10 edited books. His research collaborations extend globally, engaging in numerous national and international projects with leading software companies, serving as a principal researcher and chief software/system architect. His professional interests and expertise cover a wide array of fields such as consumer electronics, enterprise and automotive systems, critical and cyber-physical infrastructures, defense, and energy systems, to name a few. His approach to solving industrial challenges is holistic, systemic, and interdisciplinary, leveraging his deep knowledge in areas including software and systems architecting, product line engineering, model-driven and aspect-oriented software engineering, global software development, data science, and AI.

Cagatay Catal is a Full Professor at Qatar University, Department of Computer Science & Engineering. He obtained his BSc and MSc degrees in Computer Engineering from Istanbul Technical University in 2002 and 2004, respectively. He later pursued his Ph.D. in Computer Engineering at Yildiz Technical University in Istanbul, completing it in 2008. From 2020 to 2021, he held the position of Full Professor in the Department of Computer Engineering at Bahcesehir University in Istanbul. Prior to that, he worked for two years as a full-time faculty member at Wageningen University &

Research (WUR) in the Netherlands. Before his time at WUR, Dr. Catal worked for six years in the Department of Computer Engineering at Istanbul Kultur University, where he served as an Associate Professor and Head of the Department. In April 2014, he was awarded the title of Associate Professor by the Inter-University Council of Turkey. Before entering the academic field, Dr. Catal had 8 years of experience at the Scientific and Technological Research Council of Turkey (TUBITAK), Information Technologies Institute, where he held the positions of Senior Researcher and Project Manager. His research interests include various areas, including Artificial Intelligence, Deep Learning, Large Language Models, Software Testing, Software Architecture, and Precision Agriculture.