JOURNAL OF MANAGEMENT INFORMATION SYSTEMS
2024, VOL. 41, NO. 3, 708–743
https://doi.org/10.1080/07421222.2024.2376384

Routledge
Taylor & Francis Group

Enhancing Vulnerability Prioritization in Cloud Computing Using Multi-View Representation Learning

Steven Ullman[a], Sagar Samtani[b], Hongyi Zhu[a], Ben Lazarine[b], Hsinchun Chen[c], and Jay F. Nunamaker Jr.[c]

[a]Alvarez College of Business, University of Texas at San Antonio, San Antonio, TX, USA; [b]Kelley School of Business, Indiana University, Bloomington, IN, USA; [c]Eller College of Management, University of Arizona, Tucson, AZ, USA

ABSTRACT

Cybersecurity is a present and growing concern that needs to be addressed with both behavioral and design-oriented research. Public cloud providers such as Amazon Web Services and federal funding agencies such as the National Science Foundation have invested billions of dollars into developing high-performance computing resources accessible to users through configurable virtual machine (VM) images. This approach offers users the flexibility of changing and updating their environment for their computational needs. Despite the substantial benefits, users often introduce thousands of vulnerabilities by installing open-source software packages and misconfiguring file systems. Given the scale of vulnerabilities, security personnel struggle to identify and prioritize vulnerable assets for remediation. In this research, we designed a novel unsupervised deep learning-based Multi-View Combinatorial-Attentive Autoencoder (MV-CAAE) to capture multi-dimensional vulnerability data and automatically identify groups of similar vulnerable compute instances to help facilitate the development of targeted remediation strategies. We rigorously evaluated the proposed MV-CAAE against state-of-the-art methods in three technical clustering experiments. Experiment results indicate that the MV-CAAE achieves V-measure scores (metric of cluster quality) 8 percent-48 percent higher than benchmark methods. We demonstrated the practical value through a comprehensive case study by clustering vulnerable VMs and gathering qualitative feedback from experienced security professionals through semi-structured interviews. The results indicated that clustering vulnerable assets can help prioritize vulnerable instances for remediation and enhance decision-making tasks. The present design-research work also contributes to our theoretical knowledge of cyber-defense.

KEYWORDS

Online vulnerability; multi-view representation learning; attention mechanisms; cybersecurity; deep learning; cyberinfrastructure; design science; cloud computing; asset clustering

Introduction

Cloud computing has emerged as a powerful technology to support an abundance of computational workflows across industries including banking, finance, e-commerce, and more [92]. Prevailing public cloud providers and U.S. federal agencies, including Amazon

CONTACT Steven Ullman steven.ullman@utsa.edu Alvarez College of Business, University of Texas at San Antonio, One UTSA Cir., San Antonio, TX 78249, USA

Supplemental data for this article can be accessed online at https://doi.org/10.1080/07421222.2024.2376384

Web Services and the National Science Foundation (NSF), have made significant investments in scalable computing resources and large-scale scientific cyberinfrastructure (CI) to support a range of high-impact research areas, from neuroimaging to DNA sequencing [83]. Users of public clouds and scientific CI can access computing resources through configurable virtual machines (VM) [21, 76] where they can install open-source software packages from third-party platforms (e.g., GitHub) and manipulate file systems to support their computational workflow. The global VM market, anticipating a valuation of $119 billion by 2031, comprises millions of users from organizations such as Apple, Netflix, Samsung, and more [16, 28]. Thousands of VMs are provisioned on demand to support various computational applications (e.g., analytical modeling and web hosting).

Despite the benefits, open-source software in VMs often contains vulnerabilities that can compromise CI [76]. Detecting vulnerabilities within VMs presents a unique set of challenges compared to detecting vulnerabilities in individual software packages. VMs are comprised of multiple categories of data with complex representations (e.g., file system tree structures and software package dependency networks). Unlike isolated software packages where vulnerabilities can be detected using scanners, VMs are comprised of thousands of open-source packages, each potentially harboring vulnerabilities. Furthermore, users who inadvertently install software with vulnerabilities can further increase the attack surface [15, 76]. This problem is further amplified by the number of active VMs and, if exploited, can lead to data loss, misuse of resources, and disrupt critical business operations [15]. Illustrated in Figure 1 is an example of a VM containing a vulnerable package from GitHub.

In Figure 1, the selected VM (Figure 1a) contains the PacificBiosciences package "*python-pbcommand*" (Figure 1b). This package is vulnerable to shell injection attacks (Figure 1c), allowing attackers to execute arbitrary code. Assessing software vulnerabilities in cloud environments requires analysts to extract and scan software from the VMs for

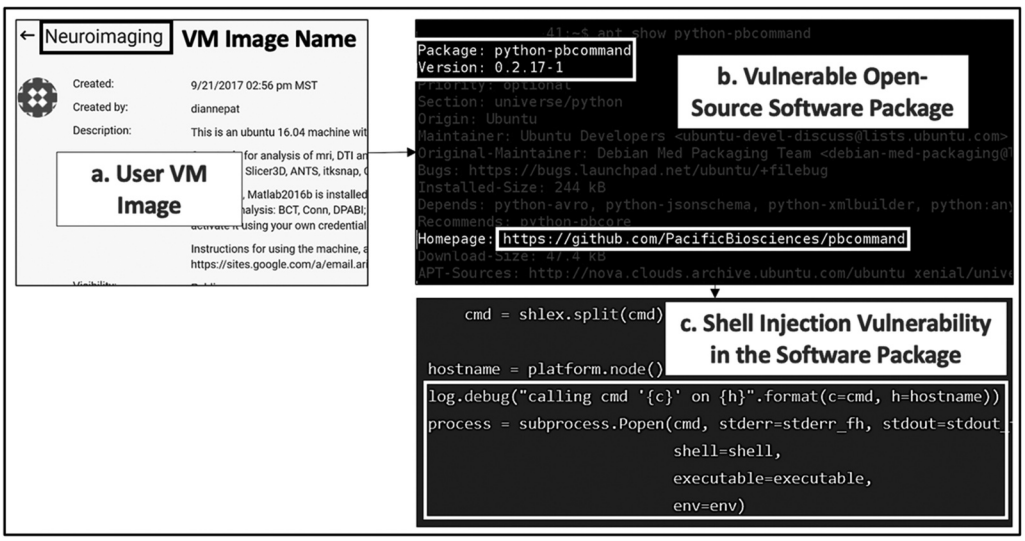


Figure 1. Selected virtual machine (VM) including a) a user VM image, b) a selected vulnerable open-source software package, and c) a shell injection vulnerability in the software package.

vulnerabilities. This process aligns with information security management principles, which have become a vital organizational concern with substantial managerial implications [71]. Information security management seeks to safeguard information technology (IT) infrastructure and assets and can be characterized by activities such as detecting, prioritizing, remediating, and preventing security risks (e.g., vulnerabilities) [46, 84]. Personnel such as analysts in security operations centers (SOC) and IT auditors must identify assets, scan for vulnerabilities, manually prioritize vulnerabilities for remediation [41], and then prioritize vulnerable assets for remediation by categorizing and grouping assets with similar vulnerabilities [33].

Despite the well-established nature of these processes, a key challenge in managing software vulnerabilities from an information security management perspective is overcoming the vast amount of data and high dimensionality. VMs and other IT assets encapsulate thousands of software packages, comprising varying distributions of severity and frequency [76]. Multiple vulnerabilities afflicting IT assets and software packages also do not allow for a single, definitive label for categorizing and prioritizing vulnerabilities, requiring an approach to overcome the lack of labeled data. The task of grouping (clustering) in cybersecurity has been used for anomaly detection to automatically identify malicious behavior and attacks using unlabeled data [6, 25]. Grouping VMs with similar vulnerabilities could help cybersecurity analysts develop targeted remediation strategies [72]. However, the magnitude of vulnerabilities returned is beyond comprehension for any individual [33]. Furthermore, public clouds and scientific CI often lack dedicated support staff to identify, prioritize, and remediate vulnerabilities that remain undetected for years [37, 56]. The magnitude of vulnerabilities and lack of sufficient labor requires a computational approach to group and prioritize vulnerable VMs for remediation.

Recent Information Systems (IS) cybersecurity analytics literature has primarily followed the computational design science paradigm to develop novel deep learning (DL) methods for Dark Web analytics [23, 64, 66]. DL has emerged as a powerful methodology capable of capturing rich data representations without manual feature engineering, which is critical in cybersecurity contexts [1,2, 67]. This alleviates security analysts who are managing vulnerabilities from dedicating time to manually engineer and select features [66, 67]. Additionally, DL can operate on complex data structures which are found in VMs (e.g., package dependency networks). Due to the deluge of software and vulnerability data, DL is a suitable methodological approach to group and prioritize vulnerable VMs for remediation. The difference in data and domain characteristics between software vulnerabilities and prior Dark Web analytics research in information systems (IS) necessitates a novel DL-based IT artifact [66]. In light of these concerns, we propose the following research questions (RQs):

Research Question 1 (RQ1): How can we automatically identify the vulnerabilities that afflict VMs in cloud computing?

Research Question 2 (RQ2): How can we design a DL-based approach to automatically group and prioritize vulnerable VMs for remediation?

In this research, we adopted the computational design science paradigm [60] to develop a novel research design that (1) automatically scans VMs to detect

vulnerabilities in open-source software on VMs and (2) groups VMs with similar vulnerabilities to facilitate vulnerability prioritization and remediation efforts. We propose a novel unsupervised DL-based Multi-View Combinatorial-Attentive Autoencoder (MV-CAAE) design artifact that draws upon state-of-the-art techniques in unsupervised graph embedding, multi-view representation learning, and attention mechanisms to fuse packages, vulnerabilities, and file systems from a VM into an embedding to support vulnerability prioritization. Our MV-CAAE has two novelties in its design:

- First, the MV-CAAE incorporates a Combinatorial Attention Mechanism (CAM) that weighs package, vulnerability, and file system data to measure the relevance of each input.
- Second, the MV-CAAE fuses the data using an inverse relevance score, creating a more comprehensive VM representation than using the input data separately.

Following the design science paradigm, we rigorously evaluated our MV-CAAE design artifact against benchmark graph embedding methods, fusion mechanisms, and multi-view autoencoder variants [34, 60]. The evaluation was performed on a gold standard dataset of VMs provided by two prevailing NSF-funded science gateways. The results demonstrated that the MV-CAAE outperformed state-of-the-art graph embedding and fusion mechanisms for grouping similar vulnerable VMs. We demonstrated the MV-CAAE's practical utility through a case study that sought to automatically group (cluster) vulnerable VMs for two major NSF-funded scientific CI partners. We elicited qualitative feedback about the utility and value of the MV-CAAE results through semi-structured interviews with four organizations, including two science gateway partners and two additional organizations that maintain research and education CI. The results of the case study illustrated that the MV-CAAE can create groups of vulnerable VMs and allow stakeholders to identify VMs with high-severity vulnerabilities that can be prioritized for remediation. The case study also demonstrated evidence of usefulness, which was corroborated by stakeholders. The proposed research design and the MV-CAAE have practical implications for analysts in SOCs and IT auditors for grouping and prioritizing vulnerable VMs.

The remainder of this paper is organized as follows. First, we review two categories of related research to gather a domain and methodological understanding for the design of our artifact. We review IS cybersecurity analytics, computational design science principles, and information security management research to guide the development of our artifact and introduce the theoretical background. We also establish the methodological background of our research by reviewing unsupervised graph embeddings, multi-view representation learning, and attention mechanisms. We then summarize the targeted domain and methodological research gaps our study targets. Third, we present our proposed research design. Specifically, we formulate meta-requirements from kernel theories in information security management and identifying methodological limitations that our design addresses. Fourth, we present the evaluation results against benchmark methods. Fifth, we demonstrate the practical utility of our proposed MV-CAAE framework with a two-part case study. Sixth, we summarize the practical implications and contributions of our research design. Finally, we conclude our research and discuss promising future directions.

Related Research

Developing artifacts under the computational design science paradigm requires a deep domain understanding to guide the artifact design [34, 60, 67]. Therefore, we review two categories of literature to inform the design of a novel DL-based vulnerability assessment IT artifact. First, we examine recent IS cybersecurity analytics research and information security management to ground and guide our research. Second, we review three areas of DL literature to enhance our methodological foundation and guide the development of our artifact. We review unsupervised graph embedding methods for representing package and file system relationships, multi-view representation learning for combining categories of VM data, and attention mechanisms for dynamically weighing package and file systems during the multi-view learning process.

IS Cybersecurity Analytics and Computational Design Science Principles

Cybersecurity analytics has emerged as an important research stream in the IS discipline. We summarize selected recent IS cybersecurity analytics research in Table 1. For each study, we present the year it was published, the author(s), the objective, the dataset(s) used, the analytical method, and if the study incorporated a vulnerability assessment (the focus of our study).

Recent cybersecurity analytics studies have centered around de-anonymizing cybercriminals [89], detecting and classifying hacker assets on the Dark Web for cyber threat intelligence (CTI) [7, 8, 23, 24, 64, 65, 91], assessing the impact of hacker attacks [68], and decision- making for managing risk [10, 38, 46, 96]. The analytical methods used in earlier hacker analysis studies are primarily machine learning (ML)-based methods [65, 89], while more recent studies have leveraged DL [7, 8, 23, 24, 64]. Despite providing essential contributions to the cybersecurity IS knowledge base, recent IS cybersecurity analytics research has largely focused on addressing challenges related to CTI, designing DL-based artifacts to address malicious hackers and Dark Web activity [7, 8, 23, 24, 64, 65, 91]. The emphasis on exploring hacker activity and Dark Web data leaves a dearth of research investigating the potential vulnerabilities (weaknesses that allow hackers to compromise IT infrastructure [64]) that many hackers target with their exploits. Additionally, recent studies have designed novel DL-based methods based on the unique data characteristics of Dark Web hacker forum posts and exploits, such as global text dependencies and sequential word patterns [7, 23, 24, 64]. However, the data characteristics of VMs are fundamentally different from Dark Web data as VMs contain large networks of data (software package dependency networks, file system hierarchies) and potential vulnerabilities.

Developing a novel IT artifact capable of assessing and prioritizing vulnerable VMs requires a design-based approach. The design science research paradigm offers prescriptive guidance to scholars aiming to develop novel IT artifacts for a practically motivated problem [34]. The breadth of research inquiries within the IS discipline has helped four genres of design science research to emerge: computational, optimization, representation, and economics [60]. Of the four genres, computational design science research provides several guidelines for developing novel IT artifacts (e.g., algorithms) [60]. First, the artifact's design can be practically motivated by key domain requirements or data characteristics. For example, Ebrahimi et al. [24] leveraged the webpage structure of DNMs to guide the development of a novel transductive SVM for identifying cyber threats. Computational

Table 1. Summary of selected recent IS cybersecurity analytics literature.

Year	Author	Objective	Dataset(s)	Analytical method	Vulnerability assessment?	Publication
2024	Ampel et al. [7]	Automatically label hacker exploits for proactive CTI	DNFs	BiLSTM, Attention	None	MISQ
2024	Ampel et al. [8]	Link exploits to the MITRE ATT&CK framework.	DNFs, CRMF	Transformer	None	JMIS
2022	Jensen et al. [38]	Improve phishing reporting through security gamification	Survey Data	Lab Experiment	None	JMIS
2022	Samtani et al. [64]	Link exploits found on hacker forums with identified vulnerabilities	DNFs, Vulnerability Scans	DSSM	Yes, Nessus	MISQ
2022	Ebrahimi et al. [23]	Detect hacker assets in non-English language on DNFs	DNFs	BiLSTM, GAN	None	MISQ
2020	Liu et al. [46]	Effect of IT governance on security breaches	Data Breaches	Regression	None	JMIS
2020	Ebrahimi et al. [24]	Cyber threat identification on the Dark Web	DNMs	SVM, BiLSTM	None	JMIS
2020	Sen et al. [68]	Determine the impact of hacker attacks in software markets	Software market	Regression	None	JMIS
2020	Zhuang et al. [96]	Determine if security awareness improves organizational security	Phishing websites	DID	None	JMIS
2019	Yin et al. [89]	Identify entities by de-anonymizing the Bitcoin blockchain	Bitcoin transactions	Boosted classifiers	None	JMIS
2019	Yue et al. [91]	Observe the impact of hacker forum discussion on DDoS attacks	DNFs	LDA	None	MISQ
2018	Benaroch [10]	Mitigating cybersecurity risk through IT investment decision-making	DoS incidents, emails	Real options model	None	ISR
2017	Samtani et al. [65]	Classification of malware code on DNFs	DNFs	SVM, LDA	None	JMIS

Abbreviations: BiLSTM, bi-directional long short-term memory; COC, convention on cybercrime; CRMF, Cybersecurity Risk Management Framekwork; DID, difference-in-differences; DNF, dark net forum; DNM, dark net marketplace; DoS, denial of service; GAN, generative adversarial network; IRC, internet-relay-chat; ISR, information systems research; JMIS, Journal of Management Information Systems; LDA, latent dirichlet allocation; MISQ, MIS Quarterly; SVM, support vector machine.

design science artifacts can also be informed by meta-requirements carefully derived from specific kernel theories [1]. Second, scholars should demonstrate the novelty of their proposed IT artifact's design through a series of evaluations that compares their artifact's performance against state-of-the-art approaches via well-established quantitative evaluation metrics (e.g., homogeneity, completeness, V-measure). Finally, novel artifacts should contribute situated implementations or nascent design theory (e.g., design principles) back to the IS knowledge base to guide future research [32, 60]. To inform the design of an IT

artifact that can identify and prioritize vulnerable VMs across multiple dimensions of vulnerabilities, we review information security management.

Information Security Management

Information security management has emerged as an important sociotechnical concern for IT managers and organizations and has drawn substantial research interest from IS scholars [46]. While information security management extends across a wide range of contexts, the objectives can be broadly categorized as (1) identifying assets, vulnerabilities, and threats, (2) assessing and prioritizing assets and vulnerabilities, (3) respond to and remediate vulnerable assets, and (4) continuously monitoring for further risks [5, 84]. The principles of objectives 1 and 2, which are the most pertinent to our study, are supported by two key theories: *Control and Auditing Theory* and *Risk Management Theory*. Control and Auditing Theory suggests that organizations should establish systems for preventing, detecting, and correcting illegal events [35, 85]. Frameworks such as the Control Objectives for Information and Related Technology (COBIT) and ISO 17799 frameworks have emerged to ensure appropriate IT governance and define information security management requirements [61, 69]. Under this theory, systems should be capable of auditing assets to identify potential security risks (vulnerabilities). Risk Management Theory comprises establishing and maintaining information systems security and suggests that threats and vulnerabilities can be assessed by analyzing security risks [84, 86]. Following this theoretical background, an IT artifact should be capable of auditing IT assets for vulnerabilities and allow key decision-makers to assess and prioritize vulnerable assets. IT infrastructure in public clouds and scientific CI comprises various IT assets like VMs and open-source software. Security analysts and managers need to identify potential vulnerabilities to determine appropriate actions for securing IT infrastructure. After identifying vulnerabilities, analysts categorize them into groups of similar types and severities to help managers prioritize and secure IT infrastructure. The abundance of vulnerabilities (types and severities) in clouds and CIs necessitates an IT artifact that automates the detection and grouping of vulnerable IT assets (i.e., vulnerable VMs and open-source software).

Auditing and assessing VMs for vulnerabilities, requires approaches for identifying assets and ascertaining their vulnerabilities, respectively. Device fingerprinting is a common IT auditing task that identifies and categorizes devices based on their characteristics by extracting device-specific features to create a unique signature representing the device [20, 48]. Recent literature indicates that scholars are developing supervised ML/DL-based methods to classify Internet-of-Things (IoT) devices and detect physical position by analyzing network traffic and radio frequency sequences [11, 47, 48, 73]. VMs offered by public clouds and scientific CI have three fundamental properties that prevent the direct application of extant ML/DL-based fingerprinting methods. First, VMs are typically hosted on a single server in a virtual environment. However, the prevailing task for IoT fingerprinting aims to detect a device's physical position. Second, extant fingerprinting methods leverage supervised learning methods to classify known devices. However, VMs in public clouds and scientific CI that offer VMs are not labeled with vulnerability types and severities. Finally, VMs allow users to install numerous open-source packages and alter the file system contents and structure. Such capabilities are not typically seen in IoT devices. Taken together, the differences between VMs and IoT devices necessitate an approach that can capture multiple modalities of VM data into a single embedding (representation).

Table 2. Summary of prevailing graph embedding methods.

Core operation	Description	Method	Node features?	Reference
Spectral Fingerprint	Extract statistical network measures to represent the graph	FeatherG	Yes	Rozemberczki and Sarakar [63]
		NetLSD	Yes	Tsitzulin et al. [75]
		Spectral Fingerprinting	No	De Lara and Pineau [43]
Implicit Factorization	Decompose a graph adjacency matrix to represent the entire graph as an embedding	Graph2vec	Yes	Narayanan et al. [53]

Unsupervised Graph Embedding Methods

Capturing multiple modalities of VM data into a single representation requires an approach that can encode the package, vulnerability, and file system data and their relationships. Therefore, we review unsupervised graph embedding methods for representing package and file system relationships. Packages and file systems in VMs both follow graph structures $G = (V, E)$, where G is the overall graph, V denotes the nodes in the graph (packages or file system directories), and E denotes the edges connecting the nodes (package dependency relationships, file paths) [3,50]. Creating a representation of an entire VM that can be used to group vulnerable VMs requires encoding (i.e., embedding) the entire graph [31]. Since VMs in public clouds and scientific CI do not contain labels detailing the vulnerability types and severities in the VMs, an unsupervised graph embedding approach is required. Graph-level embedding methods aim to represent entire graphs as points in vector space [62]. A graph embedding is a mapping function $\psi : g_i \rightarrow z_i \in \mathbb{R}^d$, where ψ embeds a graph g_i to an embedding z_i of d-dimensions; similar graph embeddings z_i and z_j will be mapped close together in vector space and indicate similarities between graph properties (nodes, edges, features). The embedding is then input into ML and DL models for clustering and classification tasks [18]. Prevailing graph embedding methods are categorized into two types based on core operation: implicit factorization (Table 2) [62].

FeatherG [63], graph2vec [53], and NetLSD [75] hold significant promise for our research as they can capture package vulnerabilities as node features. However, prevailing graph embedding methods are designed to operate on a single data view. Combining multiple views of VM data requires a multi-view representation learning strategy, which we review next.

Multi-View Representation Learning

Multi-view representation learning aims to learn a data representation by relating multiple data views (i.e., modalities) from the same source to boost learning performance [44]. Two major categories of multi-view representation learning exist: alignment and fusion. In alignment-based learning, separate models are trained for each view and the features are aligned after training. Given two datasets X and Y, alignment is denoted as $r(x; W_r) \leftrightarrow s(y; W_s)$, where $r(\cdot)$ and $s(\cdot)$ denote embedding functions to transform the datasets to vector space, and $\leftrightarrow$ is the alignment operation. Fusion-based learning employs a joint training strategy to create a shared embedding for two data views. Fusion operates

Table 3. Summary of prevailing multi-view representation learning approaches.

Multi-view category	Selected approach	Example model(s)	Operation to produce the representation	References
Alignment-based	Correlation	CCA, Sparse CCA, Kernel CCA, Deep CCA	CCA	Yang et al. [88]
	Distance	CFA, correspondence autoencoder	Frobenius norm	Gao and Guan [29];
	Similarity	Cross-modal ranking, cross-modal hashing, deep cross-view embedding	Dot-product similarity	Wang et al. [81]
Fusion-based	Graphical Model	Multi-modal topic learning, multi-view sparse coding, latent multi-view Markov networks	PCMF	Zhu et al. [95]
	Neural Network	Multi-modal autoencoder, multi-view CNN, multi-modal RNN	Sum, multiplication, concatenation, average	Cui et al. [19]; Kanezaki et al. [39]

Abbreviations: CCA, canonical correlation analysis; CFA, cross-modal factor analysis; MF, matrix factorization; PCMF, probabilistic collective matrix factorization.

through $h = \phi(x, y)$, where h denotes the joint representation and $\phi(\cdot)$ denotes the fusion operation between inputs x and y. Summarized in Table 3 are the prevailing multi-view representation alignment and fusion approaches with example models.

Compared to alignment-based approaches, fusion-based multi-view learning is suitable for our task since it allows us to combine multiple representations (i.e., VM packages and file systems) into a single embedding for downstream tasks (e.g., clustering). In particular, IS scholars are increasingly leveraging neural network-based techniques on vulnerability assessment data for CTI and vulnerability management [64, 66]. The lack of a priori knowledge (e.g., labels) about vulnerabilities in packages and file system hierarchies necessitates an unsupervised approach for producing an embedding for each VM [44]. The prevailing unsupervised neural network-based multi-view architecture is the Multi-View Autoencoder (MVA). Prevailing MVA methods primarily operate on image, text, and sensor data [12, 90]. The fusion process employs a concatenation or sum operation to merge both views, which is suitable for retaining information from complimentary multimodal data or synthetic unimodal data. However, fusing independent data views from the same source requires an operation to prioritize features. Attention mechanisms are a promising approach that can identify correlated features in package and file system views to determine which view should be prioritized during fusion.

Attention Mechanisms

Attention mechanisms are layers in DL architectures that assign trainable weights to input data features [9] and allow DL models to attend to input features that are highly correlated [22]. An attention layer is formally denoted by a query Q, and a key-value pair (K, V), where Q, K, and V denote the expected output, the input data, and the internal data representation, respectively. The attention is computed as the weighted sum of V, which is determined by a scoring function measuring the similarity between Q and K. Scoring functions are operations that determine how attention is calculated. The prevailing functions are summarized in Table 4 [14].

Additive and dot-product scoring mechanisms are among the most popular functions due to their simplicity. These functions are extended to scale the attention values when dimensionality is high (scaled dot-product), incorporate an additional weight matrix (general), or use bias terms and activation functions (bias and activated general). MVA architectures have leveraged additive and dot-product scoring functions to

Table 4. Summary of prevailing attention mechanism scoring functions.

Scoring functions	Description	Equation
Additive (Concatenate)	Non-linear activation applied to concatenated representations of query and key vectors.	$w^T \times \sigma(W_i \times Q + W_j \times K) + b$
Dot-product	Calculates attention weights by taking the dot-product between query and key vectors, followed by a softmax activation.	$Q^T K$
Scaled dot-product	Scales the dot-product between query and key vectors by a factor of the square root of the dimensionality.	$\frac{Q^T K}{\sqrt{d_k}} V$
General	Pairwise interactions between query and key vectors by applying a learned linear transformation.	$Q^T WK$
Biased general	Introduces bias terms to assign relevance to query-key pairs based on learned biases.	$Q^T WK + b$
Activated general	Incorporates an additional activation function after the linear transformation.	$\sigma(Q^T WK + b)$
Similarity	Calculates attention weights by measuring the similarity or distance between query and key vectors.	$sim(Q, K)$

capture word relevancy between user reviews and items in recommender systems [49, 93] and measure relevancy between words and pictures in multimedia retrieval [36]. Despite the important contributions, recent MVA research only adopts standard implementations of attention mechanisms to capture word relevancy before fusion, treating each embedding identically. How to formulate attention layers in an MVA to combine independent package, vulnerability, and file system representations requires further study.

Research Gaps

We identified four key research gaps from our review of related research as it pertains to our study's context. In Table 5, we categorize the reviewed bodies of literature and briefly summarize the main points, the associated research gaps, and their implications for our study.

First, from IS cybersecurity analytics and information security management, there is a lack of research that seeks to develop automated methods for detecting and prioritizing vulnerabilities. This dearth hinders decision-makers from operationalizing information security management principles to audit and assess IT asset vulnerabilities. Developing artifacts to enhance vulnerability management efforts can bolster organizational cyber-defense [66]. Second, prevailing ML/DL-based fingerprinting methods operate on a single view of data from IoT devices and omit software package vulnerabilities in the overall representation that can offer VMs can help facilitate vulnerable VM prioritization and remediation. Third, ML/DL-based fingerprinting methods analyze network traffic and radio frequency sequences to create a fingerprint. However, packages and file systems in VMs follow a graph representation structure. Graph embedding methods can represent entire graphs but only operate on a single view of data. Fourth, prevailing MVAs primarily operate on complementary and closely linked text and image data. However, the data characteristics of packages and file systems are independent and require a mechanism that can compute the relevancy between the inputs prior to fusion. We aim to address these research gaps with our proposed research design.

Table 5. Summary of reviewed literature, research gaps, and implications.

Bodies of literature	Main points	Research gaps	Implications for our study
-IS Cybersecurity Analytics -Information Security Management	-Recent IS cybersecurity analytics research has investigated hacker activity on the Dark Web. -Theories from information security management offer guidelines for identifying, prioritizing, and remediating security risks. -Device auditing and asset inventories are completed using fingerprinting methods.	-IS cybersecurity analytics research lacks artifacts to address vulnerabilities. -Fingerprinting methods operate on single-view data, primarily from IoT devices, and do not consider device vulnerability data.	-Designing IT artifacts for vulnerability management can assist in cyber-defense. -Devices often include multiple views of data that characterize them. Including vulnerability data in device fingerprints can bolster auditing tasks with vulnerability grouping and prioritization.
-Unsupervised Graph Embedding Methods	-Graph embeddings capture statistical and feature data from graphs into a latent representation (embedding). -Classes of methods are capable of embedding entire graphs.	-Prevailing methods are designed to only operate on single-view graph representations	-VMs are characterized by multiple types of data views (e.g., packages and file systems). Packages and file systems from VMs are both represented as graphs and capture different VM properties.
-Multi-View Representation Learning and Attention Mechanisms	-Learns representations of multiple views of the same input for alignment or fusion. -Assigns weights to relevant input features.	-Input embeddings are primarily treated identically instead of independently.	-In order to combine multiple, independent views of VM data (e.g., packages and file systems), the views must be carefully weighed.

Abbreviations: IS, information systems; IT, information technology; IoT, internet of things; VM, virtual machine.

Research Design

Computational design science research has leveraged kernel theories and justificatory knowledge to inform the design of novel artifacts [34, 79]. Furthermore, studies following the computational genre of design science should seek to address the limitations of the methodological knowledge base from which the IT artifacts are derived [34, 60, 67]. Following these guidelines, we propose a novel research design to automatically detect vulnerabilities in open-source software on VMs and group similar vulnerable VMs to facilitate prioritization and remediation efforts. Our research design is informed by two meta-requirements derived from kernel theories: Control and Auditing Theory and Risk

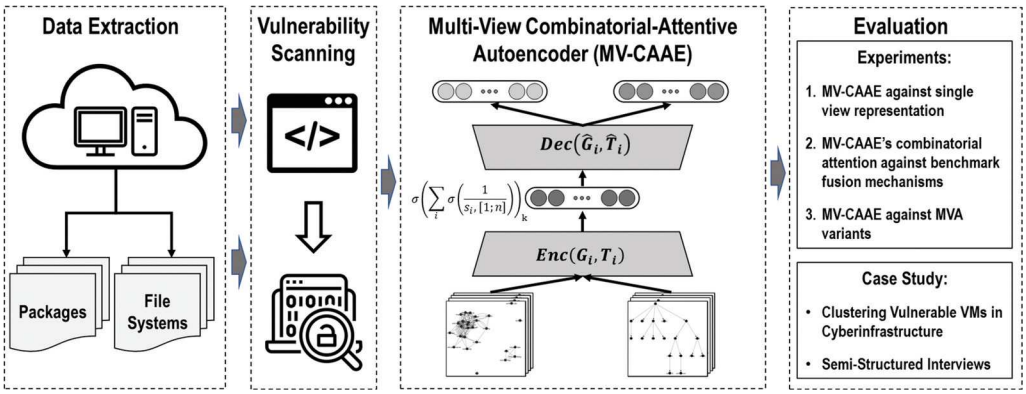


Figure 2. Proposed research design.

Management Theory. We develop a novel DL-based IT artifact that addresses limitations of unsupervised graph embeddings, multi-view learning, and attention mechanisms in prioritizing vulnerable VMs in cloud environments. Illustrated in Figure 2 is our research design with four major components: (1) Data Extraction, (2) Vulnerability Scanning, (3) Multi-View Combinatorial-Attentive Autoencoder (MV-CAAE), and (4) Evaluation.

We formulated two key meta-requirements to guide our research design that align with dimensions of our kernel theories, Control and Auditing Theory and Risk Management Theory. Summarized in Table 6 are the originating statements from our kernel theories, derived meta-requirements from the theories, the corresponding research design components, and methodological limitations addressed by our MV-CAAE artifact.

According to our kernel theories, Control and Auditing Theory and Risk Management Theory, we derived two meta-requirements to guide our design choices for our artifact. First, public clouds and scientific CI should establish systems capable of auditing and assessing assets (e.g., VMs and open-source software) for vulnerabilities, thereby identifying potential security risks. We account for this requirement in our artifact design by auditing each VM hosted by two NSF-funded scientific CIs to extract the software packages and identify their

Table 6. Design guidelines for MV-CAAE framework.

ISM Kernel theories	Meta-requirements	Research design component	Identified domain and methodological gaps
-*Control and Auditing Theory:* Auditing procedures for information systems and IT assets should be conducted to measure control performance [35, 85].	Public clouds and scientific CI should establish systems capable of auditing assets to identify potential vulnerabilities.	*Data Extraction:* VMs from partner CIs are audited to extract relevant data (e.g., software, file systems) for producing comprehensive VM representations. *Vulnerability Scanning:* Software is scanned for vulnerabilities to determine security risks.	-Clouds and scientific CIs lack an artifact to assess internal software package vulnerabilities within VMs.
-*Risk Management Theory:* Organizations must assess potential threats and vulnerabilities of asset groups or information systems by identifying, prioritizing, and mitigating risks [84, 86].	Vulnerabilities can be assessed by analyzing security risks, allowing stakeholders to make informed decisions based on prioritized assets and vulnerabilities.	*MV-CAAE:* Comprises design features to capture and represent vulnerabilities in VMs: -*Unsupervised Graph Embeddings:* Produces latent representations (embeddings) of software package and file system graphs encapsulated into VMs. -*Multi-View Learning:* Fuses multiple dimensions of VM data to create a comprehensive representation. -*Combinatorial Attention Mechanism:* Dynamically measures relevancy of each view during fusion. *Downstream Clustering Task:* Output representations are clustered together to help prioritize groups of vulnerable VMs.	-*ML/DL Device Fingerprinting:* Fingerprinting methods do not take into account device vulnerability data. -*Unsupervised Graph Embeddings:* Prevailing methods only operate on single-view data. -*Multi-View Learning and Attention Mechanisms:* Input embeddings are primarily treated identically instead of independently -

Abbreviations: MV-CAAE, multi-view combinatorial-attentive autoencoder; CI, cyberinfrastructure; VM, virtual machine; ISM, information security management; ML/DL, machine learning/deep learning.

vulnerabilities. Second, vulnerabilities can be assessed by analyzing the security risks, allowing key stakeholders to make informed decisions based on the prioritized vulnerabilities. Our proposed MV-CAAE satisfies the second meta-requirement by creating a single representation (embedding) from the data and vulnerabilities extracted from each VM which are subsequently input into a downstream clustering task to create groups of similar vulnerable VMs. The procedure of clustering vulnerable VMs allows stakeholders to identify specific groups of VMs that should be prioritized and addressed based on the severity or frequency of vulnerabilities. We further describe each component of our research design (Figure 2 and Table 5) in the following sub-sections.

Data Extraction and Vulnerability Scanning

We partnered with two major NSF-funded science gateways, referred to as "CI-A" and "CI-B," to preserve their anonymity for our own NSF-funded project. Both gateways have received more than $140M of funding from the NSF, support more than 400 projects funded by the NSF, and have a user-base of over 70,000 life science researchers from over 7,000 institutions. Each gateway hosts its VMs on cloud computing platforms accessible to users through web browsers. We designed a custom script for each science gateway that automatically (1) launches each VM; (2) collects operating system (OS), package, and file system data for each VM; and (3) parses the data into a database. We collected the source code for each package maintained on GitHub, the prevailing open-source software platform [17], and scanned for code-based vulnerabilities, resulting in a large and valuable scientific CI and VM research testbed.

We selected two popular and well-regarded scanners to detect vulnerabilities in software package code. Bandit reconstructs the source code into abstract syntax trees (AST) to scan for nine insecurities and attacks in Python [74]. Flawfinder matches syntactic patterns in source code files to scan for three insecurities and attacks in C [40]. Bandit and Flawfinder provide descriptions of detected vulnerabilities and categorize vulnerabilities based on severity scores using data from the Common Weaknesses and Enumeration database. Bandit and Flawfinder are suitable for our study since the leading programming languages in our collection are Python and C. In particular, they are open-source scanners that organizations can implement at minimal cost [76]. We summarize the type, description, and example of vulnerabilities from each scanner in Online Supplemental Appendix 1. Scanning for vulnerabilities using open-

Table 7. Data extraction vulnerability scan summary for each science gateway.

Gateway	Number of VMs	Number of gitHub packages	Number of file systems	Number of vulnerable packages*	Vulnerability severity distribution		Top vulnerable repository/ package	Number of vulnerabilities
CI-A	126	817,646	4,014	233,090	High	2,355	usit-gd/zabbix	104
					Medium	9,959	PacificBiosciences/ pbcore	1,599
					Low	246,318	sympy/sympy	66,344
					Total:	258,632	-	-
CI-B	89	851,180	3,367	260,535	High	2,182	usit-gd/zabbix	104
					Medium	9,944	PacificBiosciences/ pbcore	1,599
					Low	165,496	annulen/webkit	9,599
					Total:	177,622	-	-

Notes: Packages can contain multiple severities of vulnerabilities. Vulnerabilities are measured across all VMs combined.

source or commercial tools is a standard practice in vulnerability management research and allows us to capture unique vulnerability characteristics of the VMs in our testbed [41]. Summarized in Table 7 are the data extraction and vulnerability scan results including the number of VMs, GitHub-based packages, file systems, vulnerable packages, vulnerability severity distribution, and top vulnerable package for each gateway.

In total, package and file system data were collected from 215 VMs (CI-A = 126, CI-B = 89). Over 800,000 GitHub-maintained packages were collected from all VMs for each gateway, ranging between 810 and 12,920 packages per VM. 4,014 file systems were collected from CI-A, and 3,367 file systems were collected from CI-B. Bandit and Flawfinder detected 233,090 vulnerable packages for CI-A and 260,535 vulnerable packages for CI-B. The package *zabbix*, a network monitoring package, contained the most high-severity vulnerabilities at 104. The package *pbcore* (designed for processing Pacific Biosciences data files) contained the most medium-severity vulnerabilities at 1,599. The packages *sympy* (66,344) and *webkit* (9,599) contained the most low-severity vulnerabilities for CI-A and CI-B, respectively.

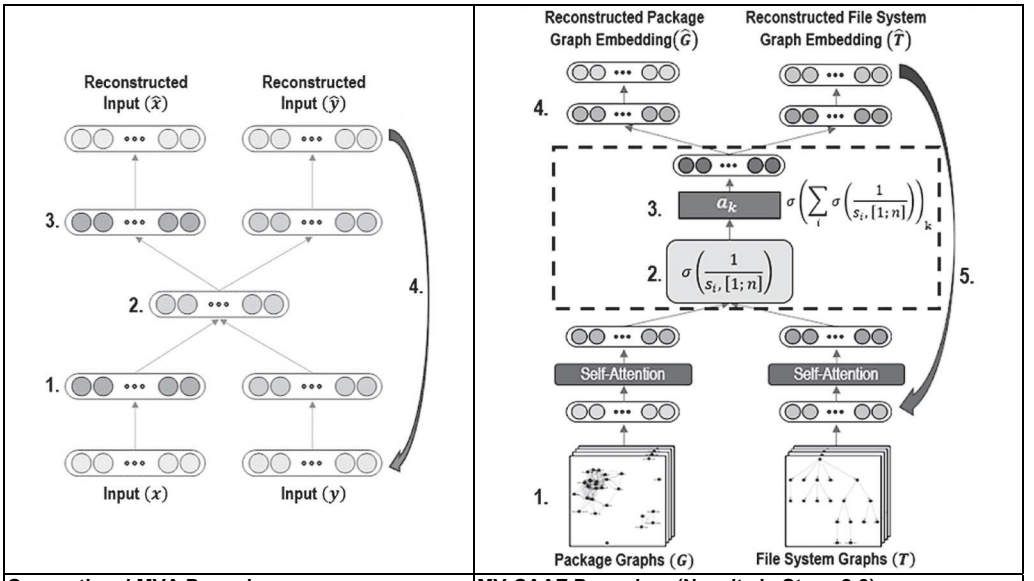
The VM testbed in this study is among the first in academia to include thousands of open-source packages, their vulnerabilities, and file systems from over 200 VMs collected from two significant scientific CI. Our testbed differs from Dark Web data used in past IS cybersecurity analytics research in several important ways [23, 64]. First, the open-source packages provide a comprehensive summary of the software in cloud VMs. Second, the code-based vulnerabilities capture vulnerabilities overlooked by general-purpose scanners used in recent IS literature [64]. Third, the file system data indicates how users configure their VMs and store data. Taken together, the unique data characteristics of our testbed can help facilitate important IS cybersecurity analytics research on open-source software security, vulnerability prioritization, and more.

Multi-View Combinatorial-Attentive Autoencoder (MV-CAAE)

Given the unique characteristics of our VM testbed and limitations of extant unsupervised graph embeddings, MVL, and attention mechanisms, we design a novel MV-CAAE that extends the MVA with a combinatorial attention mechanism (CAM) and aggregation fusion via inverse relevance scoring. The proposed MV-CAAE is contrasted against the conventional MVA in Figure 3, with the key novelties residing within Steps 2 and 3. The MV-CAAE fuses graph embeddings of package and file system data to generate an aggregated embedding to group similar VMs.

The MV-CAAE comprises five steps: (1) graph construction and embedding, (2) combinatorial attention mechanism, (3) aggregation fusion via inverse relevance scoring, (4) decoder, and (5) MSE calculation and backpropagation. The core novelty for the MV-CAAE lies in Steps 2 and 3. Each step in the MV-CAAE and its design rationale is described further below:

- **Step 1 (Graph Construction and Embedding)**: We structure the package and file system views as graphs to capture their relationships [3,50]. The package view is defined as $G = (A_G, E_G, F_G)$, where G is an undirected graph, A_G is the node set, $\{u_1, \ldots, u_n\}$, of all packages in a VM, E_G is the edge set $\{e_1, \ldots, e_n\}$ connecting the packages based on shared dependencies, and F_G is a feature matrix of vulnerabilities.



Conventional MVA Procedure:	MV-CAAE Procedure (Novelty in Steps 2-3):
1. Encoder 2. Fusion Operation 3. Decoder 4. MSE Calculation and Backpropagation	1. Graph Construction and Embedding 2. Combinatorial Attention Mechanism 3. Aggregation Fusion via Inverse Relevance Scoring 4. Decoder 5. MSE Calculation and Backpropagation

Figure 3. Comparison of conventional multi-view autoencoder (MVA) (left) and proposed multi-view combinatorial-attentive autoencoder (MV-CAAE) (right).

The file system view is defined as $T = (A_T, E_T, F_T)$, where T is an acyclic graph, A_T is the node set, $\{v_1, \ldots, v_n\}$, of all file system directories, E_T is the edge set $\{p_1, \ldots, p_n\}$ of directed edges between each directory, and F_T is the feature matrix of file system node features (e.g., permissions, file system type). We generate graph embeddings with graph2vec since it (1) embeds entire graphs, and (2) captures node features (e.g., vulnerabilities) in the embedding, denoted as $Enc(G_i) \rightarrow g_i \in \mathbb{R}^d$ and $Enc(T_i) \rightarrow t_i \in \mathbb{R}^d$, where g_i and t_i are embeddings with real values $\mathbb{R}$ of d dimensions.

- **Step 2 (Combinatorial Attention Mechanism)**: We first employ a scaled dot-product self-attention operation on each embedding to identify important features from each view separately [77]. Each embedding g_i and t_i is weighted by the self-attention operation such that $H_G = softmax\left(g_i^T g_i / \sqrt{d_g}\right) g_i$ and $H_T = softmax\left(t_i^T t_i / \sqrt{d_t}\right) t_i$, where d_g and d_t are the scaling factors that scale each weighted embedding according to the feature dimensions for each embedding. We propose a novel combinatorial attention mechanism (CAM) to measure the relevancy of MV-CAAE's independent embeddings for a comprehensive representation. For a pair of self-attended views H_i and H_j, we extract salient information combinations $s_{i,j}$, where $s_{i,j}$ is a scalar representing how relevant the views are to each other. The function $S = \left\{s_{i,j} | s_{i,j} = \phi\left(W_{i,j}\left[H_i; H_j\right]\right), i, j \in [1, \ldots, n]\right\}$ computes the relevance, where ϕ is the non-linear activation function, $W_{i,j}$ is a set of trainable weights, and S is a matrix that stores the relevance score. The relevancy score for the self-

attended package and file system views H_G and H_T is calculated as $S = \{s_{G,T} | s_{G,T} = \phi(W_{G,T}[H_G; H_T]), G_i, T_i \in [1, \ldots, n]\}$. The output relevance score matrix S is then used in Step 3 to weigh and combine the embeddings H_G and H_T. Our proposed CAM is adapted from the conventional additive attention, where the attention score is learned from the non-linear combination of the concatenated queries and keys to understand word context [9,77]. In the conventional additive attention mechanism, encoder and decoder hidden representations of source word sequences are concatenated and input into a non-linear activation function to predict a target sequence (e.g., neural machine translation). The attention score then determines how much of the hidden state of the source sequence is considered in predicting the target sequence. In contrast, our proposed CAM produces an attention matrix based on the relevancy between independent input views to determine how much from each view's hidden state should be considered in producing a combined embedding.

- **Step 3 (Aggregation Fusion via Inverse Relevance Scoring)**: To produce a comprehensive embedding for the input VM, we need to condense the representation from each view based on S that includes all non-overlapping information. Producing a representation without overlapping information creates a comprehensive representation of a VM without data redundancy from the embeddings. Based on the relevance score output S from the CAM in Step 2, we apply a softmax function to the inverse of the relevance scores $\sigma(1/s_{i,[1:n]})$ to produce a probability distribution, guiding the model to focus on feature dimensions of the embeddings H_G and H_T with few overlapping information. The probability distributions are summed across all views to obtain the relative contribution of H_G and H_T to the aggregate of the views. The attention weights are obtained from the aggregated distribution using a second softmax function: $a_k = \sigma\left(\sum_i \sigma(1/s_{i,[1:n]})\right)_k$. The condensed embedding E is produced by an attention-weighted linear-combination of the representations of all views: $E = \sum_k a_k H_k$. For the packages and file systems, the condensed embedding is represented as $E = \sum_k a_k H_G H_T$, where E is the weighted sum of the attention weights a_k multiplied by the self-attended embeddings H_G and H_T. Since each VM is unique, the relative contribution from each view will vary from VM to VM, but the package view H_G or the file system view H_T is prioritized based on the product of the attention weights a_k and the embeddings H_G and H_T.
- **Step 4 (Decoder)**: Following MVA principles, two decoder tasks $\widehat{G}_i = \sigma_{\hat{G}}(W_{\hat{G}}E + b_{\hat{G}})$ and $\hat{T}_i = \sigma_{\hat{T}}(W_{\hat{T}}E + b_{\hat{T}})$ reconstruct the inputs $\hat{G}_i$ and $\hat{T}_i$ from the shared representation E.
- **Step 5 (MSE Calculation and Backpropagation)**: After the decoder reconstructs the inputs, two *MSE* functions $MSE_{G_i} = 1/n \sum_{i=1}^{n} \left(G_i - \widehat{G}_i\right)^2$ and $MSE_{T_i} = 1/n \sum_{i=1}^{n} \left(T_i - \hat{T}_i\right)^2$ calculate the loss between the original and reconstructed package and file system inputs. The internal MV-CAAE weights are iteratively updated through backpropagation until convergence and repeated until both *MSE* functions are minimized.

The final output of the converged MV-CAAE is the fused embedding E from Step 3. This embedding is a unique representation for each VM that can be used in a downstream clustering task to group VMs with similar packages, file systems, and vulnerabilities.

Our proposed MV-CAAE has two key novelties in its design. First, the MV-CAAE's CAM computes relevance scores between multiple views of embeddings to determine which dimensions are more relevant prior to fusion. Unlike the standard additive attention, the CAM in this context allows the MV-CAAE to weigh independent features separately as opposed to identically. The conventional MVA only uses non-linear activation functions and therefore weights the inputs identically. However, packages and file systems of a VMs are fundamentally different (i.e., independent); packages contain source code with vulnerabilities whereas file systems control how data is stored on a VM. Second, the aggregation fusion via inverse relevance scoring combines non-overlapping information from each embedding to produce a comprehensive VM representation. The conventional MVA concatenates embeddings in the fusion process and does not prioritize features across both views. The MV-CAAE's aggregation fusion mechanism weights the features from the package and file system views to produce a single embedding for the VM.

Evaluation

Research guided by the computational design science paradigm should be rigorously evaluated against state-of-the-art methodologies available from the knowledge base using appropriate quantitative metrics [34, 60]. In particular, DL-based artifacts are evaluated using technical and non-technical experiments [67]. Technical experiments compare the performance of a proposed DL-based model against prevailing ML and DL benchmarks in a downstream task [67]. Performance metrics are selected based on the downstream task type (e.g., classification or clustering) and learning paradigm (e.g., supervised or unsupervised), which are informed by the domain requirements and data characteristics [67]. Non-technical experiments measure whether the proposed model can address a higher-level problem in the application environment [57, 58]. Case studies, interviews, focus groups, questionnaires, simulations, and illustrative scenarios are possible approaches to evaluate the proposed artifact in a real-world setting [57, 58, 67]. Since we followed the computational design science paradigm to develop a novel DL-based artifact, we systematically evaluated our proposed MV-CAAE with technical and non-technical approaches [8, 57, 60, 64, 67]. First, we conducted a series of rigorous technical benchmark experiments to quantitatively evaluate the performance of our MV-CAAE compared to methodologies from the knowledge bases it was designed from. Second, we performed a two-part case study to demonstrate the potential practical value of our artifact: we instantiated the MV-CAAE in a real-world setting to cluster vulnerable VMs from CI-A and subsequently gathered feedback regarding its usefulness through semi-structured interviews with four organizations.

For the technical benchmark experiments, we follow evaluation procedures from prior IS computational design science research developing DL-based artifacts, where evaluating unsupervised algorithms is often conducted by inputting the algorithm's generated embedding into a downstream task [63, 67]. The selection of a downstream evaluation task is

Table 8. Summary of benchmark experiments.

Experiment		Justification	Benchmark method	Description	References	Evaluation metrics*
1	MV-CAAE against single view representation.	Single view models are the prevailing approach for device fingerprinting.	SF	Calculates k lowest eigenvalues of normalized Laplacian matrix	de Lara and Pineau [43]	ARI, AMI, Completeness, Homogeneity, V-measure
			FeatherG	Node features and random walk weights are pooled to create graph-level statistics	Rozemberczki and Sarkar [63]	
			NetLSD	Calculates kernel heat trace of normalized Laplacian matrix	Tsitsulinet al. [75]	
			Graph2vec	Document-feature co-occurrence matrix is decomposed to generate graph representation	Narayanan et al. [53]	
2	MV-CAAE's combinatorial attention against benchmark fusion mechanisms.	Evaluate combinatorial attention weighting over benchmark fusion mechanisms that do not weigh features during fusion.	Subtraction	Element-wise subtraction of both embeddings	Blandfort et al. [13]; Francis et al. [27]; Li et al. [44]	
			Sum	An element-wise sum of both embeddings		
			Average	Average of both embeddings		
			Concatenation	Concatenation of both embeddings		
			Multiplication	Multiplication of both embeddings		
3	MV-CAAE against MVA variants.	Evaluate MV-CAAE against MVA variants with input noise, sparsity constraints, and without attention mechanisms.	Conventional MVA	MVA with no feature weighting or regularization	Goodfellow et al. [30]; Samtani et al. [64]	
			Denoising MV-CAAE	Gaussian noise added to the inputs		
			Sparse MV-CAAE	Regularization penalty applied to the inputs		
			MV-CAAE without self-attention			
			MV-CAAE without CAM and aggregation fusion			

Abbreviations: MV-CAAE, multi-view combinatorial-attentive autoencoder; SF, Spectral Fingerprinting; ARI, Adjusted Rand Index; AMI, Adjusted Mutual Information; MVA, multi-view autoencoder.

typically based on the task that the model output is designed for [80]. Our research objective is to create groups of vulnerable VMs which is naturally represented as a clustering task. Clustering refers to the task of grouping similar data points based on inherent similarities or patterns [58]. In our context, we evaluated the quality of the embeddings produced by MV-CAAE and benchmark graph embedding, multi-view fusion mechanism, and autoencoder variants such that we can provide direct comparisons between our proposed MV-CAAE and each class of methods that inspire its design. We summarize each benchmark experiment in Table 8. For each experiment, we present a justification, the benchmark methods used in the experiment, and evaluation metrics. In Experiment 1, we evaluated the MV-CAAE against state-of-the-art graph embeddings of both package and file system views independently to identify if multi-view learning creates a more comprehensive VM representation. The benchmark graph embedding methods include Spectral Fingerprinting (SF), FeatherG, NetLSD, and graph2vec [43, 53, 63, 75]. In Experiment 2, we examined the performance of the MV-CAAE's CAM against approaches that do not re-weight features during fusion. The methods in Experiment 2 include subtraction, sum, average, concatenation, and multiplication [13, 27, 44]. In Experiment 3, we investigated how the MV-CAAE performed against alternative designs. Since each component in the MV-CAAE can be varied based on autoencoder principles, we evaluated the MV-CAAE against the conventional MVA, sparse MV-CAAE, denoising MV-CAAE, MV-CAAE without self-attention, and MV-CAAE without CAM and aggregation fusion via inverse relevance scoring [30, 64]. The MV-CAAE was implemented in Python 3.7 using Keras, TensorFlow, and numpy. To facilitate scientific reproducibility, we provide full parameters of the MV-CAAE in Online Supplemental Appendix 2.

The embeddings generated from each method were clustered using K-means and evaluated using prevailing extrinsic cluster evaluation metrics [82]: adjusted rand index (ARI), adjusted mutual information (AMI), completeness, homogeneity, and V-measure. We select these evaluation metrics since other common ML evaluation metrics (e.g., accuracy, precision, recall, F1-score) are designed for supervised learning tasks (e.g., classification), where methods have prior knowledge of input labels. Our unsupervised approach learns patterns or groups (clusters) of data without prior knowledge of the label. Therefore, we require metrics that can measure the method's ability to effectively produce pre-defined clusters. ARI and AMI measure the similarity between normal and unbalanced clusters, respectively, based on true labels y and predicted labels $\hat{y}$. Scores range between -1 and 1, where scores closer to -1 or 1 are optimal. Completeness measures whether all the data points that are members of a class are elements of the same cluster. Homogeneity measures whether all the clusters contain only data points that are members of a single class. V-measure is the harmonic mean between completeness and homogeneity. Scores for completeness, homogeneity, and V-measure range between 0 and 1, where scores closer to 1 are optimal. In the context of our study, achieving strong ARI, AMI, completeness, homogeneity, and V-measure scores indicates that each cluster contains VMs with identical vulnerabilities. Security analysts responsible for assessing and managing vulnerabilities in cloud environments would use these clusters to automatically identify groups of vulnerable VMs that should be prioritized. Since each VM in their respective clusters would have near identical vulnerability distributions, security analysts could apply the same remediation strategy to each VM in the same cluster, alleviating the need to audit and assess each individual VM for vulnerabilities manually and providing greater visibility of the

vulnerabilities within each user VM in their infrastructure. Each method was performed ten times and averaged, and one-sided t-tests were used to measure statistically significant differences between method. All experiments were conducted on a Windows 10 workstation with 32GB of Random Access Memory (RAM), a NVIDIA GeForce RTX 2070 Super Graphical Processing Unit (GPU), an AMD Ryzen 7 3700x Central Processing Unit (CPU), and one terabyte of disk space.

Executing benchmark evaluations requires ground truth (i.e., gold-standard) datasets [54, 64]. Consistent with best practices in IS cybersecurity analytics literature, we constructed gold-standard datasets by randomly sampling 50 percent of the VMs from both datasets and recruited three IS cybersecurity researchers with vulnerability assessment experience to label the datasets [23, 64]. The random sample resulted in 63 VMs from CI-A and 45 VMs from CI-B. For labeling the datasets, each panel member was tasked with assigning group labels to the VMs based on the similarity between the installed packages, their vulnerabilities (severity and type), and file systems. For example, two or more VMs would be assigned to the same group if they had similar packages, vulnerabilities, and file systems. Grouping vulnerable VMs emulates how a SOC analyst or IT auditor will prioritize vulnerabilities. After the first round of grouping, we measured interrater reliability with Fleiss' kappa since our panel consisted of more than two raters [26], which resulted in 0.68 and 0.81 for CI-A and CI-B, respectively, indicating substantial agreement. We then met with the panel members to discuss the disagreements and instructed them to go through a second round of grouping. After the second round, the Fleiss' kappa for CI-A was 0.89 and 0.83 for CI-B, near-perfect agreement [51]. The remaining differences were resolved between the lead author and each panel member. This resulted in five groups of VMs for CI-A and three groups of VMs for CI-B. We summarize the gold-standard datasets in Table 9, including the number of VMs in each cluster. We provide sample data used by the panel to group the VMs, including, the average number of GitHub packages across all VMs in each cluster, the average number of vulnerable packages, the average number of file systems, and selected packages found in the clustered VMs.

Table 9. Summary of gold-standard datasets.

Dataset	Gold-standard cluster	Number of VMs	Average number of gitHub packages	Average number of vulnerable packages	Average number of file systems	Sample packages
CI-A	A_1	7	1,240	196	16	Go, Jenkins, openCASCADE
	A_2	20	2,011	430	18	Compass, NodeJS, PHP, Openstack
	A_3	15	7,131	2,248	31	Docker, JQuery, Hashicorp, Kodi
	A_4	10	10,755	2,667	35	Budgie, Caja, Ejabberd, Twitch
	A_5	11	12,917	3,987	35	Ayatana, Cloudkitty, Kopano, Staden
Total:	5	63	34,054	9,528	135	-
CI-B	B_1	7	1,227	206	15	openCASCADE, sqlite, OpenStack
	B_2	12	7,153	2,225	35	Django, Docker, Hashicorp Jquery
	B_3	26	12,846	3,963	34	Ayatana, Casacore, Cloudkitty
Total:	3	45	21,226	6,394	84	-

Abbreviations: VM, virtual machine; CI-A and CI-B, anonymous NSF-funded science gateways.

The gold-standard dataset for CI-A contained 63 VMs across five clusters, where cluster A_1 had the lowest average number of packages (1,240), and Cluster A_5 had the highest average number of packages (12,917). The number of vulnerable packages ranged from 196 to 3,987 for Clusters A_1 to A_5. The gold-standard dataset for CI-B contains three clusters with 7, 12, and 26 VMs (45 total). The average number of GitHub packages was 1,227 for B_1, 7,153 packages for B_2, and 12,846 packages for B_3. Clusters B_1, B_2, and B_3 have an average of 206, 2,225, and 3,963 vulnerable packages, respectively. The packages in the gold-standard clusters represent the different types of software installed on each VM within the assigned cluster. For example, *Docker* for containerization, and *sqlite* for database management.

Results and Discussion

Experiment 1: MV-CAAE Against Single View Representation

In Experiment 1, we evaluated our proposed MV-CAAE against single view representations. We present each method's ARI, AMI, completeness, homogeneity, and V-measure scores in Table 10. The best scores are highlighted in boldface.

The MV-CAAE outperformed state-of-the-art graph embedding methods in ARI, AMI, completeness, and V-Measure for both the package view file system view for CI-A and CI-B with statistical significance. FeatherG and graph2vec obtained higher homogeneity scores on the package view from CI-A, indicating that its clusters contain VMs primarily belonging to the same class. However, the MV-CAAE's higher completeness score (0.506 for CI-A, 0.309 for CI-B) indicates it assigned VMs from the same class to the same cluster more effectively than the second-best performing methods FeatherG and graph2vec. Interestingly, the results of the benchmark methods for the package view in CI-B were identical for each respective

Table 10. Results for Experiment 1: MV-CAAE against single view representation.

Dataset	Method		Evaluation metric				
			ARI	AMI	Completeness	Homogeneity	V-measure
CI-A	Package View	SF	0.236**	0.347**	0.435**	0.383**	0.407**
		FeatherG	0.240**	0.355**	0.440**	0.398	0.418**
		NetLSD	0.251**	0.357**	0.446**	0.391**	0.417**
		Graph2vec	0.257**	0.377**	0.432**	**0.442**	0.437**
	File System View	SF	0.217**	0.298**	0.375**	0.361**	0.368**
		FeatherG	0.247**	0.344**	0.445**	0.354**	0.394**
		NetLSD	0.155**	0.201**	0.296**	0.238**	0.264**
		Graph2vec	0.092**	0.099**	0.183**	0.188**	0.185**
	MV-CAAE		0.264	0.389	0.506	0.393	0.443
CI-B	Package View	SF	-0.014**	0.120**	0.168**	0.168**	0.168**
		FeatherG	-0.014**	0.120**	0.168**	0.168**	0.168**
		NetLSD	-0.014**	0.120**	0.168**	0.168**	0.168**
		Graph2vec	-0.014**	0.120**	0.168**	0.168**	0.168**
	File System View	SF	-0.053**	0.064**	0.153**	0.106**	0.125**
		FeatherG	-0.032**	0.106**	0.156**	0.153**	0.155**
		NetLSD	-0.036**	0.106**	0.214**	0.135**	0.165**
		Graph2vec	-0.062**	0.041**	0.125**	0.095**	0.108**
	MV-CAAE		0.386	0.256	0.309	0.286	0.297

Abbreviations: MV-CAAE, multi-view combinatorial-attentive autoencoder; ARI, Adjusted Rand Index; AMI, Adjusted Mutual Information; CI-A and CI-B, anonymous NSF-funded science gateways; SF, spectral fingerprinting.</TFN10>

metric, suggesting that the factorization and spectral fingerprinting operations could be restricted by the limited data. The MV-CAAE's higher V-measure score (0.443 for CI-A, 0.297 for CI-B) indicates that the MV-CAAE overall groups VMs from the same class more correctly and provides better intra-cluster coherence between the VMs. Included in Online Supplemental Appendix 3 is a selected example illustrating how the MV-CAAE embedding was clustered correctly and the graph2vec embedding (second-best method) was incorrectly clustered. Overall, the results of Experiment 1 suggest that capturing multiple data views provides a more complete VM representation than a single view alone [47, 76].

Experiment 2: MV-CAAE's Combinatorial Attention Against Benchmark Fusion Mechanisms

In Experiment 2, we evaluated the MV-CAAE against benchmark MVA fusion mechanisms to test whether the weighting process of the additive attention gate outperforms benchmark fusion mechanisms. We present each method's ARI, AMI, completeness, homogeneity, and V-measure scores in Table 11. The top scores are highlighted in boldface.

The MV-CAAE outperformed all fusion mechanisms with statistical significance ($p < 0.01$) in all but one metric (homogeneity) for CI-A and all metrics for CI-B. The MV-CAAE attained V-measure scores of 0.443 and 0.297 for CI-A and CI-B, respectively, surpassing the next best performing fusion mechanisms: multiplication for CI-A (0.436), and average and subtraction for CI-B (0.258). Conventional fusion mechanisms do not iteratively re-weight package and file system embeddings [27, 44]. However, MV-CAAE prioritizes the package or file system views by iteratively re-weighting the embeddings using the relevance score . Illustrated in Online Supplemental Appendix 3 is an example where the MV-CAAE embedding of a selected VM in CI-B was clustered correctly, but the embedding produced by average fusion (next best performing method) was misclustered.

Table 11. Experiment 2 results: MV-CAAE's combinatorial attention against benchmark fusion mechanisms.

		Evaluation metric				
Dataset	Method	ARI	AMI	Completeness	Homogeneity	V-measure
CI-A	Subtraction	0.225**	0.319**	0.428**	0.378**	0.402**
	Sum	0.226**	0.318**	0.435**	0.367**	0.398**
	Average	0.226**	0.318**	0.435**	0.367**	0.398**
	Concatenation	0.226**	0.318**	0.435**	0.367**	0.398**
	Multiplication	0.241**	0.353**	0.462**	0.414	0.436**
	MV-CAAE	0.264	0.389	0.506	0.393	0.443
CI-B	Subtraction	0.185*	0.207**	0.244**	0.273**	0.258**
	Sum	-0.014**	0.12**	0.168**	0.168**	0.168**
	Average	0.185*	0.207**	0.244**	0.273**	0.258**
	Concatenation	-0.014**	0.12**	0.168**	0.168**	0.168**
	Multiplication	-0.054**	0.008**	0.08**	0.063**	0.071**
	MV-CAAE	0.386	0.256	0.309	0.286	0.297

Abbreviations: MV-CAAE, multi-view combinatorial-attentive autoencoder; ARI, Adjusted Rand Index; AMI, Adjusted Mutual Information; CI-A and CI-B, anonymous NSF-funded science gateways.

Table 12. Experiment 3 results: MV-CAAE against MVA variants.

		Evaluation metric				
Dataset	Method	ARI	AMI	Completeness	Homogeneity	V-measure
CI-A	Conventional MVA	0.226**	0.318**	0.435**	0.367**	0.398**
	Denoising MV-CAAE	0.231**	0.323*	0.411**	0.363**	0.386**
	Sparse MV-CAAE	0.227**	0.346**	0.431**	0.391*	0.409**
	MV-CAAE w/o Self-Attention	0.187**	0.321**	0.466**	0.333**	0.388**
	MV-CAAE w/o CAM and Aggregation Fusion	0.226**	0.348**	0.435**	0.367**	0.398**
	MV-CAAE	0.264	0.389	0.506	0.393	0.443
CI-B	Conventional MVA	0.173**	0.213**	0.249**	0.281**	0.264**
	Denoising MV-CAAE	0.13**	0.169**	0.204**	0.22**	0.212**
	Sparse MV-CAAE	-0.014**	0.005**	0.052**	0.057**	0.055**
	MV-CAAE w/o Self-Attention	-0.014**	0.069**	0.138**	0.115**	0.126**
	MV-CAAE w/o CAM and Aggregation Fusion	-0.032**	0.106**	0.156**	0.153**	0.155**
	MV-CAAE	0.386	0.256	0.309	0.286	0.297

Abbreviations: MV-CAAE, multi-view combinatorial-attentive autoencoder; MVA, multi-view autoencoder; ARI, Adjusted Rand Index; AMI, Adjusted Mutual Information; CI-A and CI-B, anonymous NSF-funded science gateways.

Experiment 3: MV-CAAE Against MVA Variants

In Experiment 3, we evaluated how the MV-CAAE performed against the conventional MVA, denoising MV-CAAE, sparse MV-CAAE, MV-CAAE without self-attention, and MV-CAAE without the CAM and aggregation fusion. The results for each method are presented in Table 12. The top performances appear in boldface.

The MV-CAAE outperformed all MVA variants across all metrics. MV-CAAE attained statistically significant V-measure scores of 0.45 for CI-A and 0.318 for CI-B. The second-best performing methods were Sparse MV-CAAE, V-measure score of 0.409 for CI-A, and MVA, V-measure score of 0.264 for CI-B. The sparse MV-CAAE adds a regularization function that penalizes the activation functions, which omits latent features in the learning process [30]. The MV-CAAE performance decreased when either the self-attention or CAM were removed. Including self-attention indicates that capturing highly correlated features contributes to a robust fused representation. The performance of the MV-CAAE with the CAM and aggregation fusion suggests that scoring the embedding relevance and fusing based on non-overlapping information more effectively captures key features of each data view compared to the benchmark methods.

Case Study

Evaluating computational design artifacts often includes various forms of post-hoc analysis to gather evidence of the artifact's effectiveness and utility for a particular problem [58, 67]. This often comprises surveys, demonstrations, simulations, questionnaires, or interviews that gather client feedback in a real-world setting to measure the artifact's utility [58, 67, 70]. Prior computational design science research has used case studies as a frequent post-hoc analysis to demonstrate the potential value of their artifact [8, 64, 94]. Following extant computational design science research, we performed a case study with two major activities to illustrate the potential practical value of our MV-CAAE artifact. First, we demonstrate how the MV-CAAE can be operationalized by clustering embeddings generated by the MV-CAAE for each VM from CI-A. Second, we conducted four semi-structured interviews

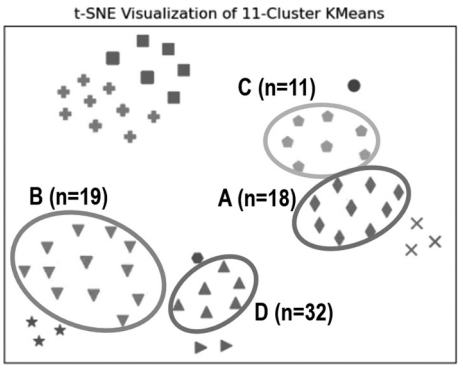
where we presented the VM clusters to participants from four different organizations to elicit qualitative feedback regarding the usefulness of the MV-CAAE.

Case Study: Clustering Vulnerable VMs in Cyberinfrastructure

The purpose behind our MV-CAAE artifact is to automatically group VMs with similar vulnerabilities together, allowing SOC analysts to prioritize groups of selected VMs for remediation. To this end, we generated embeddings for every VM from CI-A (126 VMs, 233,090 vulnerable packages, 4,014 file systems) using the MV-CAAE and subsequently clustered each embedding with K-means. The optimal number of clusters is determined by scores from Silhouette, Calinski-Harabasz, and Davies Bouldin metrics [78]. Silhouette scores range from -1 to 1, where 1 is the best value. Davies Bouldin scores range from 0 to 1, where 0 is the best value. Calinski-Harabasz scores range from 0 to, where a higher score is preferred. Eleven clusters are selected, resulting in the best Silhouette, Davies Bouldin, and Calinski-Harabasz scores at 0.885, 0.145, and 31,335, respectively. Clusters of the CI-A VMs are visualized and selected packages and their vulnerabilities from four key clusters are summarized in Table 13.

We identified four clusters that SOC analysts can consider when prioritizing their VMs for remediation. First, VMs in Clusters A and B contain the most high-severity and total vulnerabilities. Second, VMs in Clusters C and D contain the least number of vulnerabilities and therefore can be deprioritized. Clusters E and F contain an average of 3,452 vulnerable packages per VM with 909 high-severity, 1,374 medium-severity, and 2,889 low-severity vulnerabilities. Clusters A and B contain an average of 328 vulnerable packages per VM with 62 high-severity, 117 medium-severity, and 233 low-severity vulnerabilities. Given the severity and quantity of vulnerable packages in Clusters E and F, the VMs in these clusters should be prioritized for remediation.

Table 13. t-SNE visualization of clusters (k=11) and selected vulnerable packages within clusters.



Cluster	Vulnerability	Severity	Package	Count
Clusters A and B	Insecure Function	High	Libblockdev2	41
			Yadm	29
			Youker	28
	Insecure Input	High	Zabbix-cli	103
			Cupp	29
			Elastalert	13
	XSS Vulnerability	High	Libqt5webkit5	4
			Python3-spyder	3
Clusters C and D	Insecure Input	Low	node-gyp	34
			bup	31
	Insecure Module	Low	bup	20
			python-sympy	15
	Insecure Function	Medium	python-sympy	60
			python-html5lib	4

*Notes:*Clustered results contain overlapping points.

Case Study: Semi-Structured Interviews

The clusters generated by the MV-CAAE in our technical case study provide valuable insights for analysts to assess VMs across multiple dimensions of vulnerabilities and design targeted remediation strategies for vulnerable VMs. We conducted semi-structured interviews using the results from our technical case study to evaluate the potential usability and proof-of-value of our MV-CAAE artifact. Semi-structured interviews have been used in prior design science research to qualitatively evaluate design artifacts by gathering feedback from individuals with experience with a particular domain or design problem [4]. In our case, with limited organizations and participants, semi-structured interviews are preferred compared to surveys and questionnaires which typically require larger sample sizes to measure statistical significance [4, 64]. Following guidelines for conducting semi-structured interviews, we prepared a brief script that included introductions, the purpose of the interviews, and a questioning route to evaluate the utility of our research design and MV-CAAE artifact [52].

To execute the semi-structured interviews, we recruited seven participants from four different organizations with more than five years of experience and were directly involved in provisioning and securing their respective CI environments. In addition to our partner institutions CI-A and CI-B, we interviewed two additional CIs who maintain cloud and VM environments, CI-C and CI-D, to strengthen our case study and further investigate the generalizability of our MV-CAAE artifact across multiple organizations. CI-C is an external education and research SOC and provides security services for more than 250 academic and research institutions. CI-D is an academic computing infrastructure that supports and maintains a university infrastructure. The primary author was the main interviewer during the interviews and a co-author observed and took notes. In the interviews, we presented the results from the technical case study of clustering vulnerable VMs. We developed a questioning route with two vignettes to evaluate the utility of the MV-CAAE output and whether clustering vulnerable VMs for prioritization would be useful. The interviews were conducted using Zoom to accommodate out-of-state participants and the meeting was transcribed using Otter.ai transcription software. We provide further information regarding the interviews in Online Supplemental Appendix 4. Summarized in Table 14 First mention of Table 14. are the results measuring the utility of the MV-CAAE based on feedback from each participant group, where we provide indicators of evidence and counterevidence.

In summary, the technical case study of clustering vulnerable VMs and the semi-structured interviews demonstrate the practical value and provide evidence of utility for our MV-CAAE artifact. The participants from all four organizations indicated that clustering vulnerable VMs would be useful in identifying groups of vulnerable VM images to prioritize, corroborating evidence of the utility of our artifact. One participant from CI-A indicated the output of grouped VMs would be useful and further elaborated on the potential use case of the clustering output from the MV-CAAE. The participant described how the clustered results could allow them to decide whether the use of a particular VM should be stopped entirely. The participants from CI-A also indicated that the MV-CAAE output would be useful from an organizational standpoint and aid in decision-making processes. They suggested that the output could

Table 14. Semi-structured interview results measuring utility of MV-CAAE output.

Participant group	Discussion topic	Evidence of utility	Counterevidence of utility	Quotes
CI-A	MV-CAAE Cluster Results	Yes. Indicated this would help prioritize vulnerable VMs.	None.	*"Yeah . . . from an intuition point of view, I think that certainly would be on the surface good."*
	Making Decisions Using Results	Yes.	None.	*"Yeah, certainly. And then . . . depending on the severity, we could make decisions on do we need to . . . stop the use of that [VM] entirely."*
CI-B	MV-CAAE Cluster Results	Yes.	Less applicability based on deployment.	*"Yeah, I think you know . . . with the differences in deployment methodology . . . there's a little, probably less applicable notion because there's fewer images in doing this so you're not going to have the large clusters."*
	Making Decisions Using Results	Yes. Suggested that this would be helpful for vulnerable images at the project-level.	None.	*"I think the interesting thing that we could have taken away is, is to say, okay, within this cluster, is there a certain community that needs additional health support?"*
CI-C	MV-CAAE Cluster Results	Yes.	Less applicability based on returned vulnerabilities.	*"I think it's useful and it's helpful seeing the clustering, it helps you understand the relationship between the things that may not be immediately apparent."* *"At the end of the day, I already intuitively know that I have vulnerable packages, which ones actually matter?"*
	Making Decisions Using Results	Yes.	None.	*"Being able to cluster around [VMs] and make decisions [. . .] but be able to tune it to classes of vulnerabilities, that would be interesting."*
CI-D	MV-CAAE Cluster Results	Yes.	Less applicability due to purpose of IT assets.	*"The groupings of vulnerabilities, particularly severity, and which packages are causing those groupings is helpful to be aware of, particularly as you observe it over time."* *"We have a pretty small scope, but I could imagine for a very large environment, something like clustering is probably one of the few effective ways you could get your arms around certain problems."*
	Making Decisions Using Results	Yes	None.	*"Yeah [. . .], seeing the vulnerabilities that the systems have and clusters, I think it would help make our decision-making around priorities and what to handle and where to handle it just much easier to understand."*

Abbreviations: MV-CAAE, multi-view combinatorial-attentive autoencoder; VM, virtual machine; IT, information technology; CI-A, CI-B, CI-C, and CI-D, anonymous NSF-funded science gateways.

potentially help define user policies, wherein individuals who install vulnerable packages or launch VMs with vulnerabilities could be notified of the inherent flaws. The participant from CI-B expressed similar thoughts to the participants from CI-A regarding the utility of our MV-CAAE artifact but also indicated counterevidence of

utility as it pertains to the difference in VM deployment. The participant from CI-B commented on how this artifact could be used, stating that it would be interesting to assess project (community)-level VM deployments. The participants from CI-C echoed similar remarks, expressing that the MV-CAAE could be useful for clustering vulnerable VMs and enhancing decision-making capabilities. They also indicated that the clusters would be more beneficial if they could focus on specific classes of vulnerabilities. While our MV-CAAE does not directly account for this, the participants' insight offers a valuable future direction for this research. Participants from CI-D indicated that the clusters helped assess and prioritize groups of vulnerable VMs and that the process would aid in decision-making. Overall, the participants from all four organizations indicated that the clustering output from our MV-CAAE would be useful and indicated that it could help prioritize groups of vulnerable VMs for remediation or further assessment (i.e., alternative deployment strategies or permanent VM image removal). While the results were less applicable for CI-C and CI-D based on the vulnerability coverage and difference in computing environments, the feedback validated the potential utility of our approach and offered additional directions for future research.

Practical Implications and Contributions to the Is Knowledge Base

Practical Implications

The case study demonstrates the MV-CAAE's proof-of-concept and potential value by grouping similar vulnerable VMs for a major NSF-funded science gateway. The automated procedure reduces the labor-intensive process of individually scanning and assessing the types and severities of vulnerabilities in VMs. The MV-CAAE can assess and prioritize vulnerable VMs in public and scientific infrastructures. We further discuss the practical value and impact of the proposed research and how the MV-CAAE can help augment the practices for two key cybersecurity-related stakeholders as evidenced by our semi-structured interviews: SOC analysts and IT auditors [55].

SOC Analysts

Assessing the vulnerability types and severities of devices is a common task for many SOC analysts, but enterprise computing environments typically involve thousands of devices with hundreds of thousands of vulnerabilities, which surpasses human cognitive capacity [42, 59, 66]. SOC analysts can leverage the automated collection and vulnerability assessments in the proposed research design to gather device information and identify code-based vulnerabilities in open-source software installed on the devices. The output of the MV-CAAE can help analysts automatically group and prioritize vulnerable devices to guide targeted remediation efforts.

IT Auditors

IT auditors are responsible for assessing an organization's infrastructure, policies, and IT operations to evaluate security risks and associated costs. During their review, auditors will identify potential weaknesses, including vulnerabilities in VMs and software that may expose intellectual property. By utilizing clustering results from MV-CAAE's embeddings,

auditors can categorize vulnerable VMs and prioritize assets for safeguarding and remediation based on the financial value of the intellectual property contained within them.

Contributions to the IS Knowledge Base

Design science research contributions are manifested in various forms, such as design artifacts, foundations (i.e., novel methods that extend design knowledge), and methodologies [34]. IS scholars have indicated that novel IT artifacts developed under the computational design science paradigm should contribute prescriptive knowledge back to i) application environments and ii) the methodological knowledge base [34, 54, 58]. Design artifacts can also offer contributions to design theory which consist of situated implementations (Level 1), nascent design theory (Level 2), or well-developed design theory (Level 3) [32]. Level 2 contributions include methods, models, or design principles that develop nascent design theory [32]. Prevailing computational design science research has contributed design principles to formulate nascent design theory in various contexts, such as chronic care risk profiling and content viewership prediction [45, 87]. Our study consists of two primary contributions: our proposed MV-CAAE research framework (design artifact) and the two major novelties of the proposed MV-CAAE: the CAM and aggregation fusion via inverse relevance scoring (design principles).

- *MV-CAAE Research Framework*: Design artifact contributions should address problems in the application area for which they are designed and produce value to its constituent community [34]. Our novel MV-CAAE research framework contributes to existing vulnerability assessment procedures within information security management. Informed by meta-requirements derived from Control and Auditing Theory and Risk Management Theory, we contribute a design artifact to the information security management and the IS knowledge base that audits and automatically prioritizes vulnerable VMs in cloud infrastructures. This artifact empowers IT managers to make effective security decisions by automatically grouping similar vulnerable VMs and enhancing vulnerability prioritization efforts. We illustrated the potential practical value and utility of our MV-CAAE artifact in a case study, where stakeholders from four different organizations indicated the value of the artifact and clustering output. Compared to conventional approaches for managing vulnerabilities, the DL-driven approach of our MV-CAAE framework allows key stakeholders to measure and assess VMs across multiple dimensions of vulnerabilities. Reports generated by conventional vulnerability scanners provide lists of vulnerabilities that are organized individually. SOC analysts or IT auditors can sort and filter vulnerabilities by severity or type. In our context, analysts and auditors would have to repeat this procedure for every active VM instance. The benefit of our DL-based framework allows analysts to automate the data extraction and produce VM representations to automatically measure similarities to identify groups of vulnerable VMs.
- *CAM and Aggregation Fusion via Inverse Relevance Scoring*: DL-based contributions in design science research consist of novel methods or components that extend the existing methodological knowledge base that the method originated from [34, 64]. Our MV-CAAE framework contains two design principles based on our CAM and aggregation fusion via inverse relevance scoring novelties: (1) relevancy weighting for independent input embeddings and (2) fusing embeddings without overlapping information. These

design principles that can be generalized to facilitate further research inquiry in other IS domains [32, 45, 87]. Two potential bodies of IS research that these design novelties can contribute to are cybersecurity and mobile analytics. Design Principle 1 could be considered in malware analysis research by measuring the relevancy of both exploit and malware embeddings for static (i.e., raw code) and dynamic (i.e., malware behavior) analysis. Design Principle 1 dynamically weighs independent data views and it can account for key characteristics from different aspects of malware that should be prioritized. Design Principle 2 could be considered in mobile analytics research to determine a mobile device's physical position for targeted advertising. For example, a mobile device's Wi-Fi and Bluetooth connections can be represented as two heterogeneous networks and fused without overlapping shared connections. The inverse relevance scoring could account for unique signals from each connection and omit overlapping features that may be captured from radio frequency transmissions.

The MV-CAAE and design principles follow design science guidelines and contribute prescriptive knowledge to the IS and methodological knowledge bases [34]. Taken together, the design artifact and design principles of this work also contribute to a nascent design theory that can be considered in different IS domains [32]. Feedback from the semi-structured interviews with participants from four organizations provides evidence for the proof-of-concept and proof-of-value of our design artifact for grouping and prioritizing vulnerable IT assets [55].

Limitations

As with any research, our study has several limitations. First, while the MV-CAAE demonstrates how vulnerable VMs can be embedded and clustered to aid in vulnerability prioritization efforts, the vulnerabilities are dependent on the quality and accuracy of the vulnerability scanners. In our study, we used two vulnerability scanners, Bandit and Flawfinder, to detect vulnerabilities in software packages written in Python and C++/C#. However, there could be additional code-based vulnerabilities from packages in different languages, as well as network vulnerabilities (e.g., exposed ports) that afflict the VMs and are not captured by our MV-CAAE. Second, while we demonstrated the generalizability of our approach by evaluating the MV-CAAE with data from two organizations, our approach does not account for internal security policies that may vary between organizations. Third, our MV-CAAE creates embeddings of vulnerable VMs for clustering that can be used to aid in vulnerability prioritization efforts. While the clusters are automatically generated using the embeddings, our MV-CAAE framework does not indicate which cluster of VMs to prioritize. Security analysts or end users need to review the summary of vulnerabilities that are associated with the VMs in the clusters to decide further actions.

Conclusion and Future Directions

The public VM market is estimated to reach $119 billion by 2031 [28]. Public clouds and scientific CI offer millions of users across industry and academia VMs to access computing resources to execute computational workflows. However, VMs often contain vulnerabilities from third-party software packages, where exploiting vulnerabilities and misconfigurations can impede business workflows and disrupt high-impact scientific research. Drawing from

kernel theories in information security management, we developed a novel DL-based research design that audits VMs in public clouds for vulnerabilities and automatically groups similar vulnerable VMs for prioritization. We developed a novel MV-CAAE that creates an embedding for each VM based on their packages, vulnerabilities, and file systems. The embedding is then clustered to create groups of similar vulnerable VMs. The MV-CAAE operates by fusing graph embeddings of packages and file systems from VMs into a single embedding with an unsupervised multi-view learning strategy. The MV-CAAE iteratively re-weights package and file system features through a novel attention-based encoder to generate a unique, VM-specific embedding. Through a series of benchmark experiments, we demonstrated that MV-CAAE outperforms state-of-the-art graph embedding and fusion mechanisms in clustering vulnerable VMs for two prevailing NSF-funded science gateways. We executed a case study that illustrated the potential practical utility of the MV-CAAE by grouping VMs for prioritization and remediation in a major NSF-funded scientific CI. There is significant potential for adopting the MV-CAAE for general VM vulnerability assessment in enterprise systems.

There are several promising directions for future research. First, vulnerability severity could be accounted for in the research design to further enhance the VM representations, where vulnerabilities are weighted according to their severity (e.g., high severity would equate to a higher weight). Second, the MV-CAAE can be expanded to assess microservice architectures. Emerging container environments (e.g., Docker) supported by microservice architectures offer a dynamic layout to develop and host many services. Multi-view container data could include Dockerfiles and their network connections. Third, the multi-view learning component can be extended to incorporate additional VM data (e.g., network connections) to capture additional vulnerabilities. Fourth, a longitudinal analysis that captures VM representations over time can reveal the evolution of vulnerabilities as users modify VMs. Each direction can benefit vulnerability assessment practices for public clouds and scientific CI that offer VMs.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The funding source is the National Science Foundation [Grant no. CNS- 2338479].

Notes on contributors

Steven Ullman is an Assistant Professor in the Department of Information Systems and Cyber Security at the Alvarez College of Business at the University of Texas at San Antonio. He graduated with his PhD from the Artificial Intelligence (AI) Lab in the University of Arizona's Management Information Systems Department in the Eller College of Management. Dr. Ullman's research focuses on developing AI-enabled methods for cybersecurity, particularly for open-source software and enterprise IT infrastructure security. He has published in *AIS Transactions on Replication Research* and the proceedings of *IEEE ISI* conference. He has contributed to multiple research projects funded by the National Science Foundation's cybersecurity programs, including the Secure and Trustworthy Cyberspace (SaTC) and Cybersecurity Innovation for Cyberinfrastructure (CICI) programs.

Sagar Samtani is an Assistant Professor and Arthur M. Weimer Fellow in the Department of Operations and Decision Technologies at the Kelley School of Business at Indiana University (IU).

He is the Founding and Executive Director of IU's Data Science and Artificial Intelligence Lab. He graduated with his PhD from the Artificial Intelligence Lab in the University of Arizona's Management Information Systems Department in the Eller College of Management, where he served as a National Science Foundation's CyberCorps Scholarship-for-Service (SFS) Fellow. Dr. Samtani's research develops AI-enabled analytics for cybersecurity (open-source software security, Cyber Threat Intelligence, advanced cyberinfrastructure security, AI risk management), mental health, and business intelligence. He has published over 85 journal and conference papers in leading Information Systems venues such as *Information Systems Research, Journal of Management Information Systems MIS Quarterly*, as well as in cybersecurity venues such as *IEEE TDSC* and *ACM TOPS*, machine learning venues such as *ACM TKDD, IEEE TKDE*, and health outlets such as IEEE ICDH. His research has received over $5M in funding from NSF's programs, including an NSF CAREER Award. Dr. Samtani has also won several awards for his research, and was named a Top 50 Best Undergraduates Business Professor by Poets and Quants (2022). Dr. Samtani was inducted into the NSF/CISA CyberCorps Hall of Fame in 2022 for his outstanding contributions to cybersecurity. He has also received over 100 media citations.

Hongyi Zhu is an Assistant Professor in the Department of Information Systems and Cyber Security at Carlos Alvarez College of Business at The University of Texas at San Antonio. He received his PhD in Management Information Systems from the University of Arizona. Dr. Zhu's research focuses on developing advanced analytics for mobile and mental health, cybersecurity, and business intelligence. He has multidisciplinary research interests and has published in various prestigious journals, conferences, and workshops, including *MIS Quarterly, Journal of Management Information Systems, IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Privacy and Security, ACM Transactions on Management Information Systems, Journal of Biomedical Informatics*, and others.

Ben Lazarine is a PhD student in the Operations and Decision Sciences – Information Systems Department in the Kelley School of Business at Indiana University. He serves as a Research Associate in the Data Science and Artificial Intelligence Laboratory, focusing on AI-enabled cybersecurity analytics for open-source software and AI development. He has published in the proceedings of several leading conferences. He has contributed to multiple projects supported by the National Science Foundation's Secure and Trustworthy Cyberspace (SaTC) and Cybersecurity Innovation for Cyberinfrastructure (CICI) programs.

Hsinchun Chen is Regents Professor and Thomas R. Brown Chair in Management and Technology in the Management Information Systems Department at the Eller College of Management, University of Arizona. He received his PhD in Information Systems from New York University. Dr. Chen is the author/editor of over 20 books, 25 book chapters, 320 SCI journal papers, and 220 refereed conference articles covering web computing, search engines, digital library, intelligence analysis, biomedical informatics, data/text/web mining, and knowledge management. He founded the AI Lab at The University of Arizona, which has received significant research funding ($60M+) from NSF and other agencies. Dr. Chen is director of the UA AZSecure Cybersecurity Program, with $10M+ funding from NSF and other government agencies. His COPLINK/i2 system for security analytics was commercialized and acquired by IBM as its leading government analytics product. The system is in use in 5,000+ law enforcement jurisdictions and intelligence agencies in the U.S. and Europe, making significant contribution to public safety worldwide. He received the IEEE Computer Society Technical Achievement Award twice, the INFORMS Design Science Award, the AIS Impact Award, and the IEEE Big Data Security Pioneer Award. He is a Fellow of the ACM, IEEE, AAAS, and AIS. Dr. Chen has served as Editor-in-Chief, Senior Editor or Associate Editor of major journals and conference/program chair of major conferences in the areas of digital libraries, information systems, security informatics, and health informatics.

Jay F. Nunamaker, Jr. is Regents and Soldwedel Professor of MIS, Computer Science and Communication, and director of the Center for the Management of Information and the National Center for Border Security and Immigration at the University of Arizona. He received his PhD in Operations Research and Systems Engineering from Case Institute of Technology. Dr.

Nunamaker has held a professional engineer's license since 1965. Dr. Nunamaker's specialization is in the fields of system analysis and design, collaboration technology, and deception detection. He was inducted into the Design Science Hall of Fame and received the LEO Award for Lifetime Achievement from the Association for Information Systems. He was featured in the July 1997 issue of *Forbes* as one of eight key innovators in information technology. The commercial product GroupSystems ThinkTank, based on his research, is often referred to as the gold standard for structured collaboration systems. Dr. Nunamaker founded the MIS Department at the University of Arizona and served as department head for 18 years.

ORCID

Steven Ullman http://orcid.org/0000-0003-2393-8440

References

1. Abbasi, A., and Chen, H. CyberGate: A Design Framework and System for Text Analysis of Computer-Mediated Communication. *MIS Quarterly*, *32*, 4 (2008), 811–837.
2. Abbasi, A.; Chiang, R.H.L., and Xu, J. Data Science for Social Good. *Journal of the Association for Information Systems*, *24*, 6 (2023), 1439–1458.
3. Abbasi, A.; Zahedi, F.; Zeng, D.; Chen, Y.; Chen, H., and Nunamaker, J.F. Enhancing Predictive Analytics for Anti-Phishing by Exploiting Website Genre Information. *Journal of Management Information Systems*, *31*, 4 (2015), 109–157.
4. Adomavicius, G.; Bockstedt, J.C.; Gupta, A., and Kauffman, R.J. Making Sense of Technology Trends in the Information TechnologyLandscape: A Design Science Approach. *MIS Quarterly*, *32*, 4 (2008), 779–809.
5. Ahmad, A.; Hadgkiss, J., and Ruighaver, A.B. Incident Response Teams – Challenges in Supporting the Organisational Security Function. *Computers & Security*, *31*, 5 (2012), 643–652.
6. Al-Haj Baddar, S.; Merlo, A., and Migliardi, M. Behavioral-Anomaly Detection in Forensics Analysis. *IEEE Security & Privacy*, *17*, 1 (2019), 55–62.
7. Ampel, B.M.; Samtani, S.; Zhu, H., and Chen, H. Creating Proactive Cyber Threat Intelligence with Hacker Exploit Labels: A Deep Transfer Learning Approach. *MIS Quarterly*, *48*, 1 (2024), 137–166.
8. Ampel, B.M.; Samtani, S.; Zhu, H.; and Chen, H. Improving Threat Mitigation Through a Cybersecurity Risk Management Framework: A Computational Design Science Approach. *Journal of Management Information Systems*, *41*, 1 (2024), 236–265.
9. Bahdanau, D.;, Cho, K.H., and Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015*. San Diego, CA, USA, 2015, pp. 1–15.
10. Benaroch, M. Real Options Models for Proactive Uncertainty-Reducing Mitigations and Applications in Cybersecurity Investment Decision Making. *Information Systems Research*, *29*, 2 (2018), 315–340.
11. Bezawada, B.;, Bachani, M.;, Peterson, J.; Shirazi Shirazi, H. H., and Ray, I. Behavioral Fingerprinting of IoT Devices. In *Proceedings of the ACM Conference on Computer and Communications Security*. Toronto, CA, 2018, pp. 41–50.
12. Bhatt, G.; Jha, P., and Raman, B. Representation Learning Using Step-based Deep Multi-Modal Autoencoders. *Pattern Recognition*, *95*, (2019), 12–23.
13. Blandfort, P.; Karayil, T.; Raue, F.; Hees, J.; and Dengel, A. Fusion Strategies for Learning User Embeddings with Neural Networks. *arXiv*, (2019).
14. Brauwers, G.; and Frasincar, F. A General Survey on Attention Mechanisms in Deep Learning. *IEEE Transactions on Knowledge and Data Engineering*, *35*, 4 (2023), 3279–3298.

15. Cai, M.; Huang, H., and Huang, J. Understanding Security Vulnerabilities in File Systems. In *Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems, APSys*. Hangzhou, CN, 2019, pp. 8–15.
16. Cave, J. Who's Using Amazon Web Services? [2020 Update] | Contino | Global Transformation Consultancy. *Contino*, 2020. https://www.contino.io/insights/whos-using-aws.
17. Chen, W.; Jin, F.; and Xue, L. Flourish or Perish? The Impact of Technological Acquisitions on Contributions to Open-Source Software. *Information Systems Research*, *33*, 3, (2021), 867–886.
18. Cui, P.; Wang, X.; Pei, J.; and Zhu, W. A Survey on Network Embedding. *IEEE Transactions on Knowledge and Data Engineering 31*, (2019), 833–852.
19. Cui, Q.; Wu, S.; Liu, Q.; Zhong, W.; and Wang, L. MV-RNN: A Multi-view Recurrent Neural Network for Sequential Recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *32*, 2 (2020), 317–331.
20. Dalai, A.K.; Jena, A.; Sharma, S., Mohapatra, A.; Sahoo, B.; Obaidat, M.S.; Sadoun, B.; and Puthal, D.;. A Fingerprinting Technique for Identification of Wireless Devices. In *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*. Colmar, FR, 2018, pp. 1–5.
21. Deelman, E.;, Mandal, A., and Pascucci, V., ;. Cyberinfrastructure Center of Excellence Pilot: Connecting Large Facilities Cyberinfrastructure. In *Proceedings - IEEE 15th International Conference on eScience*. San Diego, CA, USA, 2019, pp. 449–457.
22. Du, M.; Liu, N., and Hu, X. Techniques for Interpretable Machine Learning. *Communications of the ACM*, *63*, 1 (2019), 68–77.
23. Ebrahimi, M.; Chai, Y.; Samtani, S., and Chen, H. Cross-Lingual Cybersecurity Analytics in the International Dark Web with Adversarial Deep Representation Learning. *MIS Quarterly*, *46*, 2 (2022), 1209–1226.
24. Ebrahimi, M.; Nunamaker, J.F., and Chen, H. Semi-Supervised Cyber Threat Identification in Dark Net Markets: A Transductive and Deep Learning Approach. *Journal of Management Information Systems*, *37*, 3 (2020), 694–722.
25. Fernandes, T.; Dias, L., and Correia, M. C2BID: Cluster Change-Based Intrusion Detection. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. Guangzhou, CN, 2020, pp. 310–319.
26. Fleiss, J.L.; Levin, B.; and Paik, M.C. *Statistical Methods for Rates and Proportions*, *3rd Edition*. Hoboken, NJ: Wiley, 2003.
27. Francis, D.; Nguyen, P.A.; Huet, B., and Ngo, C.-W. Fusion of Multimodal Embeddings for Ad-Hoc Video Search. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. Seoul, KR, 2019, pp. 1868–1872.
28. Future Market Insights. Virtual Machine Market by Type, Platform & Region for 2021 – 2031 | Global Sales Analysis and Opportunity - 2031 | FMI. *Future Market Insights*, 2021. https://www.futuremarketinsights.com/reports/virtual-machine-market/table-of-content.
29. Gao, L., and Guan, L. Information Fusion via Deep Cross-Modal Factor Analysis. In *Proceedings - IEEE International Symposium on Circuits and Systems*. Sapporo, JP, (2019), pp. 1–5.
30. Goodfellow, I.J.; Bengio, Y.; and Courville, A. *Deep Learning*. MIT Press, (2016).
31. Goyal, P. and Ferrara, E. Graph Embedding Techniques, Applications, and Performance: A Survey. *Knowledge-Based Systems*, *151*, (2018), 78–94.
32. Gregor, S., and Hevner, A.R. Positioning and Presenting Design Science Research for Maximum Impact. *Management Information Systems Quarterly*, *37*, 2 (2013), 337–355.
33. Handa, S.; and Lawson, C. Market Guide for Vulnerability Assessment. *Gartner*, 2021. https://www.gartner.com/doc/reprints?id=1-27GRFVF6&ct=210917&st=sb.
34. Hevner, A.R.; March, S.T.; Park, J., and Ram, S. Design Science in Information Systems Research. *MIS Quarterly*, *28*, 1 (2004), 75–105.
35. Hong, K.; Chi, Y.; Chao, L.R., and Tang, J. An Integrated System Theory of Information Security Management. *Information Management & Computer Security*, *11*, 5 (2003), 243–248.

36. Huang, F.; Zhang, X.; Li, C.; Li, Z.; He, Y., and Zhao, Z. Multimodal Network Embedding via Attention Based Multi-View Variational Autoencoder. In *Proceedings of the 2018 ACM International Conference on Multimedia Retrieval*. Yokohama, JP, 2018, pp. 108–116.
37. JASON. Fundamental Research. 2019. https://www.nsf.gov/news/special_reports/jasonsecurity/JSR-10-21FundamentalResearchSecurity_12062019FINAL.pdf.
38. Jensen, M.L.; Wright, R.T.; Durcikova, A., and Karumbaiah, S. Improving Phishing Reporting Using Security Gamification. *Journal of Management Information Systems*, *39*, 3 (2022), 793–823.
39. Kanezaki, A.; Matsushita, Y., and Nishida, Y. RotationNet for Joint Object Categorization and Unsupervised Pose Estimation from Multi-View Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*, 1 (2019), 269–283.
40. Kaur, A., and Nayyar, R. A Comparative Study of Static code Analysis Tools for Vulnerability Detection in C/C++ and JAVA Source Code. In *Procedia Computer Science* 171 , 2020, pp. 2023–2029.
41. Kenna Security. What is Vulnerability Management Prioritization? 2021. https://www.kennasecurity.com/blog/what-is-vulnerability-management-prioritization/.
42. Kiser, R.; Welch, V., and Wheeler, B. University Data Can Fuel AI/ML Cybersecurity Research: A Vision with an OmniSOC Proofpoint. In *ACM KDD AI4Cyber: The 1st Workshop on Artificial Intelligence-enabled Cybersecurity Analytics*. Virtual, 2021, pp. 1–5.
43. de Lara, N., and Pineau, E. A Simple Baseline Algorithm for Graph Classification. In *Relational Representation Learning Workshop, NIPS 2018*. Montreal, Canada, 2018, 1–7.
44. Li, Y.; Yang, M., and Zhang, Z.M. A Survey of Multi-View Representation Learning. *IEEE Transactions on Knowledge and Data Engineering*, *31*, 10, pp. 1863–1883. , (2018).
45. Lin, Y.-K.; Chen, H.; Brown, R.A.; Li, S.-H., and Yang, H.-J. A Bayesian Multitask Learning Approach. *MIS Quarterly*, *41*, 2 (2017), 473–496.
46. Liu, C.-W.; Huang, P., and Lucas, H.C. Centralized IT Decision Making and Cybersecurity Breaches: Evidence From U.S. Higher Education Institutions. *Journal of Management Information Systems*, *37*, 3 (2020), 758–787.
47. Liu, Y.; Wang, J.; Li, J.; et al. Zero-Bias Deep Learning for Accurate Identification of Internet-of-Things (IoT) Devices. *IEEE Internet of Things Journal*, *8*, 4 (2021), 2627–2634.
48. Luo, Y.; Hu, H.; Wen, Y., and Tao, D. Transforming Device Fingerprinting for Wireless Security Via Online Multitask Metric Learning. *IEEE Internet of Things Journal*, *7*, 1 (2020), 208–219.
49. Ma, C.; Kang, P.; Wu, B.; Wang, Q., and Liu, X. Gated Attentive-Autoencoder for Content-Aware Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. Melbourne, VIC, AUS, 2019, pp. 519–527.
50. Mathur, A.; Cao, M., and Dilger, A.: Ext4: The Next Generation of the Ext3 File System *USENIX*. Boston, MA, USA. In. 2007, pp. 25–30.
51. McHugh, M.L. Interrater Reliability: The Kappa Statistic. *Biochemia Medica*, *22*, 3 (2012), 276.
52. Myers, M.D., and Newman, M. The Qualitative Interview in IS Research: Examining the Craft. *Information and Organization*, *17*, 1 (2007), 2–26.
53. Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y., and Jaiswal, S. graph2-vec: Learning Distributed Representations of Graphs. In *13th International Workshop on Mining and Learning with Graphs*. Halifax, NS, CA, 2017, pp. 1–8.
54. Nunamaker, J.F.; Chen, M., and Purdin, T.D.M. Systems Development in Information Systems Research. *Journal of Management Information Systems*, *7*, 3 (1990), 89–106.
55. Nunamaker, J.F., Jr; Briggs, R.O.; Derrick, D.C., and Schwabe, G. The Last Research Mile: Achieving Both Rigor and Relevance in Information Systems Research. *Journal of Management Information Systems*, *32*, 3 (2015), 10–47.
56. Osborne, C. Open Source Software Security Vulnerabilities Exist for Over Four Years Before Detection. *Zero Day ZDNet*, 2020. https://www.zdnet.com/article/open-source-software-security-vulnerabilities-exist-for-over-four-years-before-detection-study/.

57. Peffers, K.; Rothenberger, M.; Tuunanen, T., and Vaezi, R. Design Science Research Evaluation. In *Design Science Research in Information Systems. Advances in Theory and Practice*, 2012, pp. 398–410.
58. Peffers, K.; Tuunanen, T.; Rothenberger; M.A., and Chatterjee, S. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*, 3 (2007), 45–77.
59. Peisert, S., and Welch, V. The Open Science Cyber Risk Profile: The Rosetta Stone for Open Science and Cybersecurity. In *IEEE Security and Privacy*, 2017, pp. 94–95.
60. Rai, A. Editor's Comments: Diversity of Design Science Research. *MIS Quarterly*, *41*, 1 (2017), iii–xviii.
61. Ridley, G.; Young, J., and Carroll, P. COBIT and its Utilization: A Framework from the Literature. In *Proceedings of the 37th Annual Hawaii International Conference on System Science*. Big Island, HI, USA, 2004, p. 8.
62. Rozemberczki, B.; Kiss, O., and Sarkar, R. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *ACM International Conference on Information & Knowledge Management*. Virtual, 2020, pp. 3125–3132.
63. Rozemberczki, B., and Sarkar, R. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *ACM International Conference on Information and Knowledge Management*, Virtual, 2020, pp. 1325–1334.
64. Samtani, S.; Chai, Y., and Chen, H. Linking Exploits from the Dark Web to Known Vulnerabilities for Proactive Cyber Threat Intelligence: An Attention-based Deep Structured Semantic Model. *MIS Quarterly*, *46*, 2 (2022), 911–946.
65. Samtani, S.; Chinn, R.; Chen, H., and Nunamaker, J.F. Exploring Emerging Hacker Assets and Key Hackers for Proactive Cyber Threat Intelligence. *Journal of Management Information Systems*, *34*, 4 (2017), 1023–1053.
66. Samtani, S.; Kantarcioglu, M., and Chen, H. Trailblazing the Artificial Intelligence for Cybersecurity Discipline: A Multi-Disciplinary Research Roadmap. *ACM Transactions on Management Information Systems*, *11*, 4 (2020), 1–19.
67. Samtani, S.; Zhu, H.; Padmanabhan, B.; Chai, Y.; Chen, H., and Nunamaker, J.F. Deep Learning for Information Systems Research. *Journal of Management Information Systems*, *40*, 1 (2023), 271–301.
68. Sen, R.; Verma, A., and Heim, G.R. Impact of Cyberattacks by Malicious Hackers on the Competition in Software Markets. *Journal of Management Information Systems*, *37*, 1 (2020), 191–216.
69. von Solms, B. Information Security Governance: COBIT or ISO 17799 or Both? *Computers & Security*, *24*, 2 (2005), 99–104.
70. Sonnenberg, C., and vom Brocke, J. Evaluations in the Science of the Artificial – Reconsidering the Build-Evaluate Pattern in Design Science Research *Design Science Research in Information Systems. Advances in Theory and Practice: 7th International Conference, DESRIST 2012*. Las Vegas, NV, USA (Springer), 2012, pp. 381–397.
71. Straub, D.W., and Welke, R.J. Coping with Systems Risk: Security Planning Models for Management Decision Making. *MIS Quarterly*, *22*, 4 (1998), 441–469.
72. Tenable. Group Vulnerabilities (Nessus). *Tenable*. https://docs.tenable.com/nessus/Content/GroupVulnerabilities.htm .
73. Thangavelu, V.; Divakaran, D.M.; Sairam, R.; Bhunia, S.S., and Gurusamy, M. DEFT: A Distributed IoT Fingerprinting Technique. *IEEE Internet of Things Journal*, *6*, 1 (2019), 940–952.
74. Torkura, K.A., and Meinel, C. Towards Vulnerability Assessment as a Service in OpenStack Clouds. In *Proceedings - Conference on Local Computer Networks, LCN* Dubai, UAE, 2016, pp. 1–8.
75. Tsitsulin, A.; Mottin, D.; Karras, P.; Bronstein, A., and Müller, E. NetLSD: Hearing the Shape of a Graph. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2018), pp. 2347–2356.
76. Ullman, S.; Samtani, S.; Lazarine, B.; Zhu, H.; Patton, M., and Chen, H. Smart Vulnerability Assessment for Scientific Cyberinfrastructure: An Unsupervised Graph Embedding Approach.

In *Proceedings - 2020 IEEE International Conference on Intelligence and Security Informatics*. Arlington, VA, USA, 2020, pp. 1–6.
77. Vaswani, A.; Shazeer, N.; Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.;. Attention Is All You Need. *31st Conference on Neural Information Processing Systems (NIPS 2017)*. Long Beach, CA, USA, 2017, pp. 5999–6009.
78. Vergara, V.M.; Salman, M.; Abrol, A.; Espinoza, F.A., and Calhoun, V.D. Determining the Number of States in Dynamic Functional Connectivity Using Cluster Validity Indexes. *Journal of Neuroscience Methods*, *337*, (2020), 108651.
79. Walls, J.G.; Widmeyer, G.R., and El Sawy, O.A. Building an Information System Design Theory for Vigilant EIS. *Information Systems Research*, *3*, 1 (1992), 36–59.
80. Wang, B.; Wang, A.; Chen, F.; Wang, Y., and Kuo, C.C.J. Evaluating Word Embedding Models: Methods and Experimental Results. *APSIPA Transactions on Signal and Information Processing*, *8*, (2019), 1–14.
81. Wang, K.; Yang, M.; Yang, W., and Yin, Y. Deep Cross-View Label Embedding with Correlation and Structure Preserved for Multi-Label Classification. In *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*. Volos, GR, 2018, pp. 12–19.
82. Wang, R.; Li, H.; Chen, M.; Dai, Z., and Zhu, M. MIC-KMeans: A Maximum Information Coefficient Based High-Dimensional Clustering Algorithm. In *Advances in Intelligent Systems and Computing*, 2019, pp. 208–218.
83. Wang, Z.; Chu, T.; Christie, M.A., Abeysinghe, T.C., Marru, S., Pierce, M., Danko, C.G.;. Building a Science Gateway For Processing and Modeling Sequencing Data Via Apache Airavata. In *Proceedings of the Practice and Experience on Advanced Research Computing: Seamless Creativity (PEARC '18)*. Pittsburgh, PA, pp. 1–7, 2018.
84. Webb, J.; Ahmad, A.; Maynard, S.B., and Shanks, G. A Situation Awareness Model for Information Security Risk Management. *Computers & Security*, *44*, (2014), 1–15.
85. Weber, R. *Information Systems Control And Audit*. Englewood Cliffs, NJ: Prentice-Hall, 1999.
86. Wright, M. Third Generation Risk Management Practices. *Computer Fraud & Security, 1999*, 2 (1999), 9–12.
87. Xie, J.; Chai, Y., and Liu, X. Unbox the Black-Box: Predict and Interpret YouTube Viewership Using Deep Learning. *Journal of Management Information Systems*, *40*, 2 (2023), 541–579.
88. Yang, X.; Liu, W.; Tao, D.; Cheng, J., and Li, S. Multiview Canonical Correlation Analysis Networks for Remote Sensing Image Recognition. *IEEE Geoscience and Remote Sensing Letters*, *14*, 10 (2017), 1855–1859.
89. Yin, H.H.S.; Langenheldt, K.; Harlev, M.; Mukkamala, R.R., and Vatrapu, R. Regulating Cryptocurrencies: A Supervised Machine Learning Approach to De-Anonymizing the Bitcoin Blockchain. *Journal of Management Information Systems*, *36*, 1 (2019), 37–73.
90. Yuan, Y.; Xun, G.; Jia, K., and Zhang, A. A Multi-View Deep Learning Framework for EEG Seizure Detection. *IEEE Journal of Biomedical and Health Informatics*, *23*, 1 (2019), 83–94.
91. Yue, W.T.; Wang, Q.H., and Hui, K.L. See No Evil, Hear No Evil? Dissecting the Impact of Online Hacker Forums. *MIS Quarterly*, *43*, 1 (2019), 73–95.
92. Zhang, Z.; Nan, G., and Tan, Y. Cloud Services vs. On-Premises Software: Competition Under Security Risk and Product Customization. *Information Systems Research*, *31*, 3 (2020), 848–864.
93. Zhou, J.P.; Cheng, Z.; Perez, F., and Volkovs, M. TAFA: Two-headed Attention Fused Autoencoder for Context-Aware Recommendations. In *14th ACM Conference on Recommender Systems*. Online, 2020, pp. 338–347.
94. Zhu, H.; Samtani, S.; Chen, H., and Nunamaker, J.F. Human Identification for Activities of Daily Living: A Deep Transfer Learning Approach. *Journal of Management Information Systems*, *37*, 2 (2020), 457–483.
95. Zhu, X.; Vo, K.D.; Guo, J., and Long, J. Multiple Manifold Regularized Sparse Coding for Multi-View Image Clustering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. Torino, IT, 2018, pp. 1723–1726.
96. Zhuang, Y.; Choi, Y.; He, S.; Leung, A.C.M.; Lee, G.M., and Whinston, A. Understanding Security Vulnerability Awareness, Firm Incentives, and ICT Development in Pan-Asia. *Journal of Management Information Systems*, *37*, 3 (2020), 668–693.