MIS Quarterly

RESEARCH ARTICLE

CREATING PROACTIVE CYBER THREAT INTELLIGENCE WITH HACKER EXPLOIT LABELS: A DEEP TRANSFER LEARNING APPROACH[1]

Benjamin M. Ampel
Department of Management Information Systems, University of Arizona,
Tucson, AZ, U.S.A. {bampel@arizona.edu}

Sagar Samtani
Department of Operations and Decision Technologies, Indiana University,
Bloomington, IN, U.S.A. {ssamtani@iu.edu}

Hongyi Zhu
Department of Information Systems and Cyber Security, University of Texas at San Antonio,
San Antonio, TX, U.S.A. {hongyi.zhu@utsa.edu}

Hsinchun Chen
Department of Management Information Systems, University of Arizona,
Tucson, AZ, U.S.A. {hsinchun@arizona.edu}

The rapid proliferation of complex information systems has been met by an ever-increasing quantity of exploits that can cause irreparable cyber breaches. To mitigate these cyber threats, academia and industry have placed a significant focus on proactively identifying and labeling exploits developed by the international hacker community. However, prevailing approaches for labeling exploits in hacker forums do not leverage metadata from exploit darknet markets or public exploit repositories to enhance labeling performance. In this study, we adopted the computational design science paradigm to develop a novel information technology artifact, the deep transfer learning exploit labeler (DTL-EL). DTL-EL incorporates a pre-initialization design, multi-layer deep transfer learning (DTL), and a self-attention mechanism to automatically label exploits in hacker forums. We rigorously evaluated the proposed DTL-EL against state-of-the-art non-DTL benchmark methods based in classical machine learning and deep learning. Results suggest that the proposed DTL-EL significantly outperforms benchmark methods based on accuracy, precision, recall, and F1-score. Our proposed DTL-EL framework provides important practical implications for key stakeholders such as cybersecurity managers, analysts, and educators.

Keywords: Hacker forums, cyber threat intelligence, cybersecurity analytics, deep transfer learning, deep learning, exploit labeling, computational design science

Introduction

The rapid proliferation of new information technology (IT) in recent years has created significant benefits for modern society. Despite its usefulness, IT often possesses numerous vulnerabilities that can allow unauthorized users unfettered access to an organization's networks, systems, private data, and other critical assets. Sophisticated hackers often develop exploits (e.g., SQL injections, cross-site scripting, etc.) in plain-text source code to execute cyber breaches (Samtani et al., 2017). Each cyber breach is estimated to cost an average of $7,910,000 to an organization (Sun et al., 2020). This cost

[1] Gediminas Adomavicius was the accepting senior editor for this paper. Nachiketa Sahoo served as the associate editor.

underscores the importance for organizations to proactively identify exploits. To this end, organizations are increasingly investing in cyber threat intelligence (CTI) capabilities to detect emerging exploits (Wagner et al., 2019).

To develop proactive CTI, academia and industry are increasingly examining openly accessible exploits in major international hacker forums, exploit darknet markets (DNMs), and public exploit repositories (Samtani et al., 2020). Hacker forums are large discussion boards that provide freely accessible exploits developed by the large and evolving international hacker community. These forums often contain millions of user-generated posts and span multiple geopolitical countries and regions such as Russia, the United States, the Middle East, and others. Exploit DNMs allow hackers to share and sell highly specialized exploits (e.g., 0-days). While less common than in hacker forums, the exploits found within exploit DNMs are more sophisticated than those found in hacker forums. Public exploit repositories are collections of exploits manually labeled and vetted by industry professionals and CTI experts. These repositories are more common than exploit DNMs and provide more exploit details (e.g., attack type, common vulnerabilities and exposures, etc.) than hacker forums. Sample exploits from a hacker forum, an exploit DNM, and a public exploit repository are shown in Figure 1.

Hacker forum posts with exploit source code (top left of Figure 1) often lack a clear exploit label (i.e., category). In contrast, exploit DNMs (top right of Figure 1) and public exploit repositories (bottom middle of Figure 1) offer a rich set of metadata and clear exploit labels (e.g., local, web). A recent CTI report from the renowned SANS Institute indicated that cyber analysts at leading CTI companies require assistance in identifying, collecting, and analyzing unknown exploits from hacker communities to proactively develop knowledge about hacker capabilities (Brown & Lee, 2021; Newman, 2020). This is partly in response to the Mirai malware that caused widespread damage across the United States. The source code for Mirai was posted on a hacker community platform two months before its large-scale usage (Yue et al., 2019). These attacks show that hacker forum posts can be a valuable source for identifying precursors to attacks and require up-to-date monitoring. However, most of the exploits found in hacker forums are unlabeled (Ampel et al., 2020). Unlabeled exploits can often prevent cyber analysts from identifying the specific tactics and strategies of hackers and stymie the production of timely and relevant CTI (Tounsi & Rais, 2018). Extant procedures to label exploits are currently manual, which is a time-consuming and nontrivial process (Wagner et al., 2019). Cyber analysts frequently cite these manual processes and high workloads as their primary challenges, leading to burnout and turnover (Agyepong et al., 2020). Automating the exploit labeling process can reduce manual processes and provide cyber analysts with tactical, strategic, and operational CTI capabilities. We present selected benefits that exploit labeling can provide to each CTI type in Table 1.

Tactical CTI is the techniques, tactics, and procedures used by threat actors (Wagner et al., 2019). Providing clear labels to hacker forum source code allows cyber analysts to quickly organize discovered exploits, operationalize them, and discover the capabilities of threat actors. For example, once a cyber analyst knows a set of exploits are SQL injections (as opposed to other exploits), the analyst can build a secure copy of their IT (e.g., a virtual machine with an SQL server), conduct targeted penetration testing with the labeled exploits, and identify suitable mitigations and cyber-defenses against the successfully executed exploits. This hacker emulation is known as red teaming and is essential to modern cybersecurity practices (Alomar et al., 2020). From an operational CTI perspective, labeled exploits can help cyber analysts monitor new exploits posted on hacker forums and choose relevant exploits to investigate further in a targeted fashion (Tounsi & Rais, 2018). This operational CTI can be combined with tactical CTI and external resources (e.g., SANS CTI reports) to strengthen cyber-systems against specific attacks (e.g., Mirai malware). For example, SANS CTI reports may link web application exploits to a specific advanced persistent threat group. An analyst can use the combined information of newly labeled exploits and known strategies employed by that APT group to select the appropriate system-hardening strategies. From a strategic perspective, exploit labels allow senior officials to identify attack trends (e.g., how exploits rise or fall over time) in a timely fashion such that they can make the appropriate security investments (Tounsi & Rais, 2018). Moreover, such information can be integrated into cyber-risk assessment models to create proactive, preventive, and dynamic insights that facilitate effective cybersecurity investments (Shin & Lowry, 2020).

In this study, we adopted the computational design science paradigm (Rai, 2017) to develop a novel *deep transfer learning exploit labeler* (DTL-EL) framework that leverages the rich metadata and large quantities from exploit DNMs and public exploit repositories (source domain) to automatically label each exploit's source code found in hacker forums (target domain). The proposed framework is based on deep transfer learning (DTL), a prevailing approach in cybersecurity analytics and CTI literature in IS. DTL can transfer knowledge from a large and labeled source domain to improve model performance when the target domain is not large or rich enough to create a fit model (Zhuang et al., 2020). DTL has particular value in CTI applications that do not often have rich labeled datasets, such as hacker forum analysis (Samtani et al., 2020).

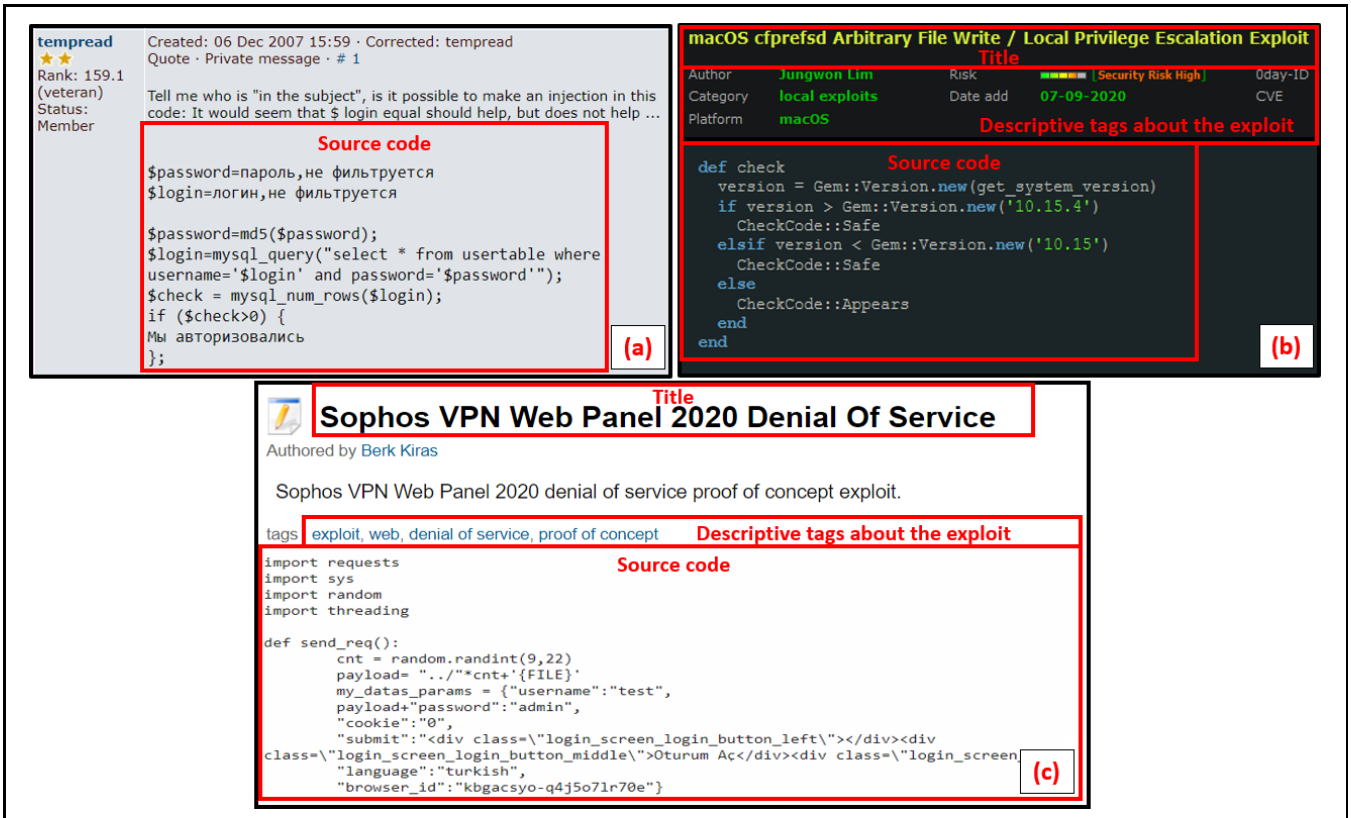

Figure 1. Examples of Posts with Exploits from: (a) a Hacker Forum, (b) an Exploit DNM, and (c) a Public Exploit Repository

Table 1. Selected Benefits of Exploit Labeling for CTI

CTI type	Selected benefits of exploit labeling	References
Tactical	Improved profiling and operationalization of hacker capabilities to guide cyber-defense strategies, including red-teaming exercises	Wagner et al., 2019
Operational	Automated real-time statistics and threat prioritization to guide internal control implementations	Brown & Lee, 2021
Strategic	Trend analysis of emerging exploits and hackers for executives and security managers to more effectively make security investments	Tounsi & Rais, 2018

Moreover, recent information systems (IS) cybersecurity research has shown the applicability of deep transfer learning (DTL) for CTI, where a model trained on a larger labeled source domain (English darknet market postings) provided significant benefits to three smaller target domain models (Russian, French, and Italian darknet market postings) (Ebrahimi et al., 2022). In addition to DTL, the proposed DTL-EL framework also incorporates several state-of-the-art methods in deep learning (DL) and attention mechanisms that were carefully designed for exploit labeling. First, DTL-EL includes a bidirectional long short-term memory (BiLSTM) with its hidden and cell states pre-initialized with the titles of professionally vetted exploits in exploit DNMs and public exploit repositories. The pre-initialization design emulates how a cybersecurity professional reads the title of an exploit before examining its source code. Second, DTL-EL transfers the pre-initialized BiLSTM model's learned representation of the source domain dataset (i.e., exploit DNMs and public exploit repositories) to help label exploit source code in hacker forums (target domain). Third, the DTL-EL incorporates a self-attention mechanism to identify and capture semantic

and long sequential dependencies within the exploit source code to improve DTL-EL's performance. Consistent with the guidelines of the computational design science paradigm, we rigorously evaluated the proposed DTL-EL with a series of benchmark experiments (Rai, 2017). The DTL-EL framework can help future IS researchers execute targeted cybersecurity analytics on exploit labeling, trend analysis, real-time monitoring, and other critical CTI tasks.

This paper is organized as follows. First, we review IS cybersecurity analytics research, computational design science guidelines, and hacker-forum exploit analysis literature. Second, we identify research gaps from prior literature and pose research questions. Third, we present DTL-EL's methodological foundation by reviewing the BiLSTM with attention mechanism and DTL. Fourth, we describe the proposed DTL-EL. Fifth, we present the results of our experiments. Sixth, we summarize our proposed DTL-EL's practical implications and contributions to the IS knowledge base. Finally, we conclude this research and discuss potential future directions.

Literature Review

IS Cybersecurity Analytics Literature and Computational Design Science Guidelines

IS scholars have made remarkable advances in cybersecurity analytics research in recent years. We summarize selected recent cybersecurity analytics literature published in prevailing IS journals in Table 2. For each study, we summarize the year of publication, author(s), cybersecurity focus, whether the approach was multi-dataset or not, analytical method(s), and if a system or UI was built to present the analytical results.

Most prior cybersecurity analytics studies have examined hacker forums to proactively identify, detect, and mitigate cyber threats (Benjamin et al., 2019; Ebrahimi et al., 2020; Samtani et al., 2017; Yin et al., 2019; Yue et al., 2019). Earlier cybersecurity analytics traditionally relied on classical machine learning (ML) or text mining methods (Benaroch, 2018; Karhu et al., 2018; Samtani et al., 2017; Sen et al., 2020; Yin et al., 2019), while more recent studies have leveraged the DL-based BiLSTM to automatically extract feature representations (i.e., embeddings) from the inputted hacker forum post content (Ebrahimi et al., 2022; Samtani et al., 2022). Despite providing timely contributions to our understanding of hacker forum content, most studies do not conduct the critical CTI task of assigning a specific label to each exploit within hacker forums. Additionally, researchers rarely embed their algorithm(s) into a system or UI. Consequently, CTI researchers or practitioners may have difficulty leveraging a cybersecurity analytics IT artifact for their cyber-defense workflow and CTI tasks (Samtani et al., 2017; Silic & Lowry, 2020).

Designing a novel IT artifact for a high-impact societal application (e.g., hacker exploit labeling) requires a principled approach. Past IS cybersecurity analytics literature has leveraged the computational design science paradigm (Hevner et al., 2004; Rai, 2017) to guide and ground their work. The computational design science paradigm provides IS scholars with three concrete guidelines to design and evaluate novel algorithms, computational models, and systems for advanced data analytics applications (Rai, 2017). First, the IT artifact's design can be inspired by key domain requirements or data characteristics. In a recent example within extant cybersecurity analytics literature, the webpage structure from DNMs guided the design of a novel transductive support vector machine (SVM) (Ebrahimi et al., 2020). Second, IS scholars should demonstrate the novelty of their design by comparing the quantitative performance (e.g., accuracy, precision, recall, F1-score) of their proposed IT artifact against well-established baseline methods. Finally, the IT artifact should contribute back to the IS knowledge base. Contributions can include a situated implementation of the IT artifact (e.g., user interface) and/or nascent design theory (e.g., design principles) (Rai, 2017). Executing each guideline requires understanding the application for which the artifact is being developed. For this study, we identify what data characteristics have been used to analyze hacker forum exploits.

Hacker Forum Exploit Analysis

Hackers use forums, carding shops, DNMs, and internet-relay-chat (IRC) to share goods (e.g., credit cards) and assets (e.g., exploits) (Benjamin et al., 2019). Hackers freely post tens of thousands of assets in forum posts, making them a viable and attractive data source for developing CTI (Samtani et al., 2017; Yue et al., 2019). Moreover, assets found within hacker forums (e.g., exploits) have been used in recent cyberattacks (Samtani et al., 2020). As a result, a growing body of literature aims to explore and categorize exploits in hacker forums. We summarize selected recent literature analyzing exploits in hacker forums in Table 3. Each study is summarized based on the year, author(s), objective, data source, the data type used, analytics, and identified exploits.

Table 2. Summary of Selected Recent IS Cybersecurity Analytics Literature

Year	Author(s)	Cybersecurity focus	Multi-dataset?	Analytical method(s)	System or UI?
2022	Samtani et al.	Exploit-vulnerability linking	Yes	DSSM with BiLSTM	No
2022	Ebrahimi et al.	Detecting hacker assets in multilingual hacker forums	No	GAN with BiLSTMs	No
2020	Ebrahimi et al.	Identifying threats in DNMs	No	Transductive SVM	No
2020	Sen et al.	Quantifying the impact of cyberattacks on software markets	No	Regression	No
2020	Silic & Lowry	Improving organizational security	No	HMSAM	Yes
2019	Yue et al.	Correlating DDoS mentions in hacker forums and DdoS victims	Yes	LDA, dynamic panel fixed effects	No
2019	Yin et al.	De-anonymization of blockchain transactions amongst criminals	No	Gradient boosting classifier	No
2019	Benjamin et al.	Darknet predictive analytics	No	OLS Regression	No
2018	Benaroch	Proactively mitigating risk for cybersecurity investments	No	Real options model	No
2018	Karhu et al.	Opening digital platforms while protecting them from exploitation	Yes	Resource-based View	No
2017	Samtani et al.	Code classification in hacker forums	No	LDA, SVM	Yes

Note: UI = user interface; BiLSTM = bidirectional long-short term memory; COC = convention on cybercrime; DdoS = distributed denial of service; DSSM = deep structured semantic model; GAN = generative adversarial network; HMSAM = hedonic-motivation system adoption model; IRC = internet-relay-chat; LDA = latent Dirichlet allocation; OLS = ordinary least squares; SVM = support vector machine.

Table 3. Selected Recent Literature Analyzing Exploits in Hacker Forums

Year	Author(s)	Objective	Data source	Data type used	Analytical method(s)	Identified exploits
2021	Zhao et al.	Attack event prediction	Hacker forums	Post content, attachments	SNA	DoS, overflow, SQLi
2019	Schafer et al.	Trend identification	Hacker forums	Titles, users, posts, topics, keywords	SNA, LDA	Leaks, botnets, DdoS
2019	Benjamin et al.	Darknet identification, collection	Hacker forums	Post content, attachments, code, keywords, reputation	OLS	Rootkit, XSS, SQLi, DdoS, drive-by
2018	Williams et al.	Exploit categorization	Hacker forums	Posts content, attachments	LSTM	Keyloggers, DdoS
2018	Goyal et al.	Cyberattack prediction	Hacker forums, Twitter, blogs	Post content, tweets, blogs	LSTM, RNN	Trojan, phishing
2018	Deliu et al.	Exploit categorization	Nulled.IO leak	Post content	SVM, CNN	Botnet, crypter, keylogger
2017	Samtani et al.	Exploit categorization	Hacker forums	Post content, authors, source code	LDA, SVM	Crypters, keyloggers

Note: CNN = convolutional neural network; DdoS = distributed denial of service; LDA = latent Dirichlet allocation; LSTM = long-short term memory; OLS = ordinary least squares; RNN = recurrent neural network; SNA = social network analysis; SQLi = structured query language injection; SVM = support vector machine; XSS = cross-site scripting

Most hacker-forum exploit analytics studies have analyzed the post content, which often contains significant jargon, surrounding an exploit's source code to categorize posts into broad categories (e.g., botnets, keyloggers, malware). Similar to IS cybersecurity analytics literature, earlier studies (2017-2018) often used classical ML (e.g., SVM) based on their tasks (Deliu et al., 2018; Samtani et al., 2017). Since classical ML approaches rely on a manually defined set of features (which is labor intensive and time-consuming to construct), scholars in recent years have relied on the DL-based LSTM or BiLSTM for their tasks (Goyal et al., 2018; Williams et al., 2018). Irrespective of the analytical method, past studies have not examined how to assign a specific label to each exploit in hacker forums based on its source code or the metadata (e.g., exploit titles) in exploit DNMs or public exploit repositories.

Research Gaps and Questions

We identified several research gaps from prior literature. First, past studies analyzing exploits in hacker forums primarily analyze post content and often omit source code. However, source code contains significant, often precise, syntax and information (e.g., function names, exploit actions, etc.) that is not in natural language (Nuñez-Varela et al., 2017). Second, hacker-forum exploit source code is nontrivial to categorize, as it often lacks clear exploit labels and metadata (e.g., informative titles). Additionally, prevailing DL-based methods (e.g., LSTM, BiLSTM) used for hacker forum analysis were designed for natural language and often struggle to capture long semantic relationships (e.g., dependencies) within exploit code. Finally, despite containing professionally vetted exploits, rich metadata, and overlapping hacker content, exploit DNMs and public exploit repositories have not been leveraged in a multi-dataset model to help label hacker forum exploits. Based on these research gaps, we pose the following research questions:

How can we extend a BiLSTM model to capture the long semantic dependencies found within exploit source code?

How can we transfer the knowledge from metadata-rich exploit DNMs and public exploit repositories to help label exploits in hacker forums based on their source code?

Methodological Foundation

To set the methodological foundation for this work, we first review the BiLSTM with attention model. BiLSTM is the prevailing algorithm for DL-based hacker community text analytics (Ebrahimi et al., 2022, 2018) and other text classification tasks (Thangaraj & Sivakami, 2018). In the following subsections, we review DTL as the prevailing approach for transferring knowledge from an information-rich source domain (e.g., exploit DNMs and public exploit repositories) to a target domain (e.g., hacker forums).

Bidirectional LSTM (BiLSTM) with Attention Model

The BiLSTM improves upon other popular DL-based text classification models (e.g., LSTM, GRU, RNN) by using hidden states (working memory) with past and future contexts of the input tokens to make a prediction (Yenter & Verma, 2017). The state-of-the-art BiLSTM for text classification incorporates a convolutional layer and an attention mechanism to capture local correlations of temporal structures and long-range phrases with sequential dependencies (Liu & Guo, 2019). A single LSTM cell within the BiLSTM with attention model is presented in further detail on the bottom left and bottom right, respectively, of Figure 2.

The BiLSTM with attention model first converts a textual input into word embeddings for input into a convolutional layer. The convolutional layer extracts contextual and sequential features from the embeddings to learn local and latent representations of the input. The BiLSTM layer processes the features and concatenates the results of the forward ($\overrightarrow{h_t}$) and backward ($\overleftarrow{h_t}$) hidden states to output a single hidden state, $h_i = \left[\overleftarrow{h_t}\ \overrightarrow{h_t}\right]^T$ at each time step (t). The same process also occurs with the BiLSTM cell states to output a single matrix, C_i (long-term memory). In an untrained model, the forward and backward LSTM cell and hidden memories are initialized to zero ($C_{t=0} = 0$ and $h_{t=0} = 0$) and updated over time (t). However, using the final $\overrightarrow{h_t}$, $\overleftarrow{h_t}$, $\overrightarrow{C_t}$, and $\overleftarrow{C_t}$ of a BiLSTM model to pre-initialize the starting hidden and cell state weights of a subsequent BiLSTM model can reduce error to improve model performance (Elsheikh et al., 2019).

In a BiLSTM with attention, the attention weight matrix takes the hidden states $H = (h_1, h_2, \ldots h_n)$ as input and calculates a weighted sum for each hidden state. The weighted sum extracts the most informative hidden states to represent the input sequence, which can be used as input into another layer for subsequent tasks (e.g., classification). Attention mechanisms operate with a query and a set of key-value pairs. The query determines which aspects of the input embedding the attention mechanism will attend according to the weighted sum of the values and their corresponding key-value pairs.

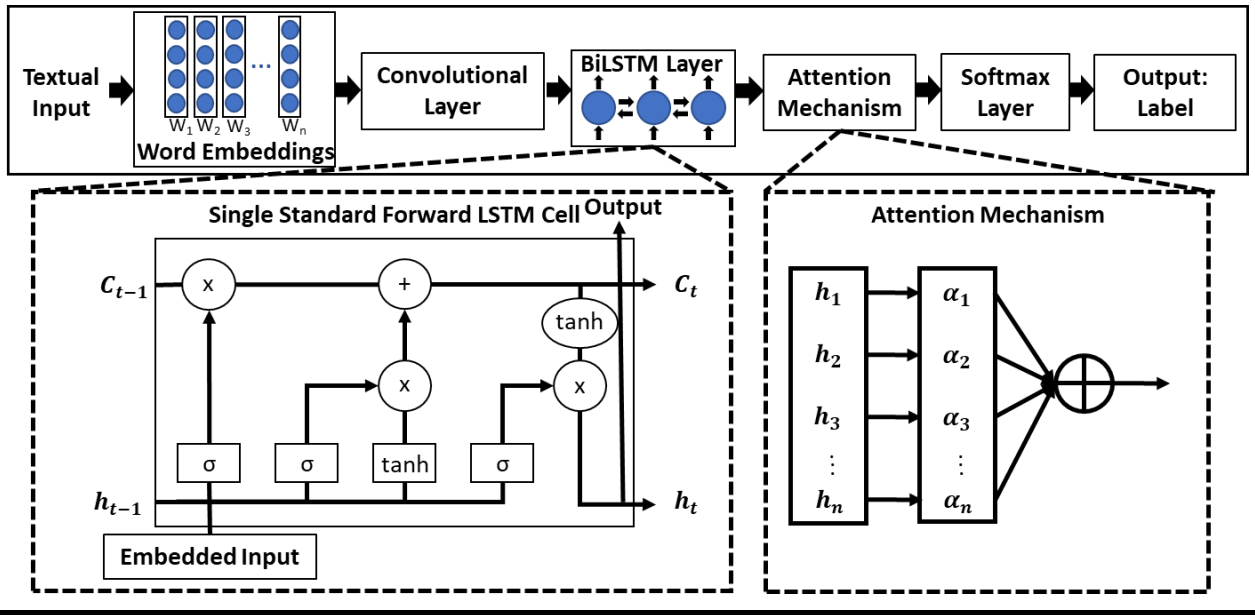

Figure 2. BiLSTM with Attention Mechanism (Adapted from Liu & Guo, 2019)

Currently, three categories of attention mechanisms can be incorporated into a BiLSTM: (1) hard attention, (2) soft attention, and (3) self-attention. Hard attention focuses on part of the input (Luong et al., 2015), while soft attention focuses on the entire input space and learns weights for all input features (Bahdanau et al., 2014). Hard attention and soft attention are often used when a parallel corpus is available (e.g., machine translation). Self-attention mechanisms relate to various positions of the same input sequence, finding the context vector between each value of an input, and applying a weight. When the dataset has one long-sequence input, the hidden state matrix H from the prior layer is linearly projected into a query (Q), key (K), and value (V) (Vaswani et al., 2017). Dot-product multiplication operations are applied to each to create learned parameters. Formally, self-attention is calculated as:

$$SelfAttn(H) = softmax\left(\frac{HW^Q(HW^K)^T}{\sqrt{d_k}}\right)HW^V,$$

where d_k is the dimension of K (i.e., HW^K) and W^Q, W^K, and W^V are learned parameters. Self-attention-based models consistently outperform models without attention mechanisms in various natural language processing (NLP) tasks (Shen et al., 2018). Past studies have indicated that the self-attention mechanism's weighting approach can improve the performance of predictive models by capturing short- and long-range contextual information within the input text (Adadi & Berrada, 2018). Additionally, researchers can implement and evaluate adversarial perturbations to identify if the feature weights learned from the self-attention mechanism explain how the model reached its end decision (Wiegreffe & Pinter, 2019). However, a BiLSTM with self-attention cannot address the issue of missing metadata (e.g., descriptive titles, labels) in hacker forum exploits. DTL is an emerging DL-based method that can leverage the learned knowledge from the rich metadata in public exploit repositories and exploit DNMs to help label exploits in hacker forums.

Deep Transfer Learning (DTL)

DTL aims to improve the performance of a task $\mathcal{T}$ (e.g., classification) in a target domain $\mathcal{D}_T$ by transferring knowledge from a source domain $\mathcal{D}_S$ (Zhuang et al., 2020). Generally, each domain $\mathcal{D}$ is represented by $\mathcal{D} = \{\mathcal{X}, P(X)\}$, where $\mathcal{X}$ is the feature space and $P(X)$ is the marginal distribution of data instances $X = \{x_1, x_2, \ldots, x_n\}$. For each $\mathcal{D}$, $\mathcal{T}$ is represented by $\mathcal{T} = \{y, f(\cdot)\}$, defined by a label space $\mathcal{Y}$ and a function $f(\cdot)$, which is a conditional probability function $P(\mathcal{Y}|\cdot)$. DTL transfers the latent knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$ to improve the predictive function $f_T(\cdot)$ for $\mathcal{T}_T$. DTL is commonly used when a $\mathcal{T}_T$ conducted in $\mathcal{D}_T$ achieves low performance due to insufficient training data ($\mathcal{N}$), and $\mathcal{N}_S > \mathcal{N}_T$ (Pan & Yang, 2010).

Four general types of DTL approaches exist: (1) instance transfer, (2) mapping transfer, (3) adversarial transfer, and (4) network transfer (Tan et al., 2018). Instance-transfer selects instances from the source domain dataset X_S based on a similarity distance formula to supplement the target

domain dataset X_T. However, this approach cannot account for different features in the source and target domains (e.g., exploit titles in one domain but not another). Mapping transfer merges instances from the source and target dataset to one data space for tasks such as multi-task domain adaptation learning for sequence tagging (Peng & Dredze, 2017) and multilingual text classification (Ebrahimi et al., 2018). Mapping transfer is best suited for semi-supervised domain adaptation where the target domain is unlabeled (Peng & Dredze, 2017). Adversarial transfer uses adversarial learning techniques to capture the shared features of different tasks independently (Liu et al., 2017). Like mapping transfer, adversarial transfer works best for unsupervised or self-supervised domain adaptation tasks (Liu et al., 2017). Network transfer reuses parts of a neural network trained on a source domain dataset to approximate the predictive function $f_T(\cdot)$ for a task in the target domain. Network transfer is powerful when the source and target domains contain labeled data. Past IS studies employing DTL approaches have used multi-layer transfer learning (MLTL) when the model in the source domain was initialized with specific features and had multiple layer types (Zhu et al., 2020). Pretraining tasks (e.g., pre-initialization) often improve MLTL performance (Liu et al., 2019). Other DTL methods include updating existing model weights with an ensemble method or using pretrained language models (Ruder et al., 2019). Prevailing approaches to evaluate DTL are comparisons against non-DTL models and/or different DTL types and ablation analyses on layer and parameter transfer (Houlsby et al., 2019).

Research Testbed and Design

We developed a novel research design based on our methodological foundation to help address the posed research questions. The proposed research design has four major components (Figure 3): (1) data collection, (2) preprocessing and dataset construction, (3) DTL-EL model, and (4) experiments and evaluations. We describe each component in the following subsections.

Data Collection

We collected three sources of exploits for our research: hacker forums, exploit DNMs, and public exploit repositories. The three sources contain varying metadata, which can fall into one of four categories. First, description metadata provides high-level information about the exploit, including title, exploit source, and date. Second, author metadata provides details about a user, including name and reputation score. Third, content metadata provides the exploit source code and the post and discussion describing the exploit. Finally, operation metadata pertains to how the exploit operates, including attack type and targeted platform. These categories are detailed in Table 4, along with features of each category, their descriptions, an example, and whether a feature is present in the three exploit collections (✓means the feature is present, X means it is not).

Exploit DNMs and public exploit repositories contain key operation metadata not found in hacker forums, such as attack type and platform. Source code and post content can be collected from all three platforms and can therefore serve as the basis for a DTL model. We collected nine hacker forums, one exploit DNM, and six public exploit repositories. Each platform was identified based on the input of cybersecurity domain experts, the popularity of the platform in the hacker community, and link-following techniques (Samtani et al., 2022). We summarize each collected platform's type, name, language, dates, posts, source code snippets, and authors in Table 5.

Our collection included 16 platforms across three languages, 258,739 source code snippets, and 999,012 unique authors. The hacker forum testbed contained 79,437 unlabeled source code snippets posted between 2002 and 2020. Hacker-forum source code snippets were identified through special code blocks used on each forum (Samtani et al., 2017). One significant exploit DNM containing 33,766 exploits made by 6,052 authors was collected. Six public exploit repositories with 145,536 professionally vetted exploits were collected. Taken together, our research testbeds far exceed the quantity presented in prevailing IS cybersecurity analytics literature (Benjamin et al., 2019; Samtani et al., 2022; Samtani et al., 2017).

Preprocessing and Dataset Construction

For exploit DNMs and public exploit repositories, the eight most popular exploit labels based on attack type were retained. These included web applications (43,475 exploits), denial of service (DoS) (12,121 exploits), remote (11,787 exploits), local (7,993 exploits), SQL injection (7,187 exploits), cross-site scripting (XSS) (7,025 exploits), file inclusion (3,412 exploits), and overflow (3,333 exploits). Source code was stripped of non-alphanumeric, lower-cased, lemmatized, and tokenized characters. Consistent with best practices in DL-based text analytics, the input sequence for DL models was padded with a special token to ensure proper lengths for all inputs (Yenter & Verma, 2017). For ML models, a fixed corpus was built from the training data vocabulary. Fixed-length vectors were created for each input via count vectorization and term frequency-inverse document frequency weighting.

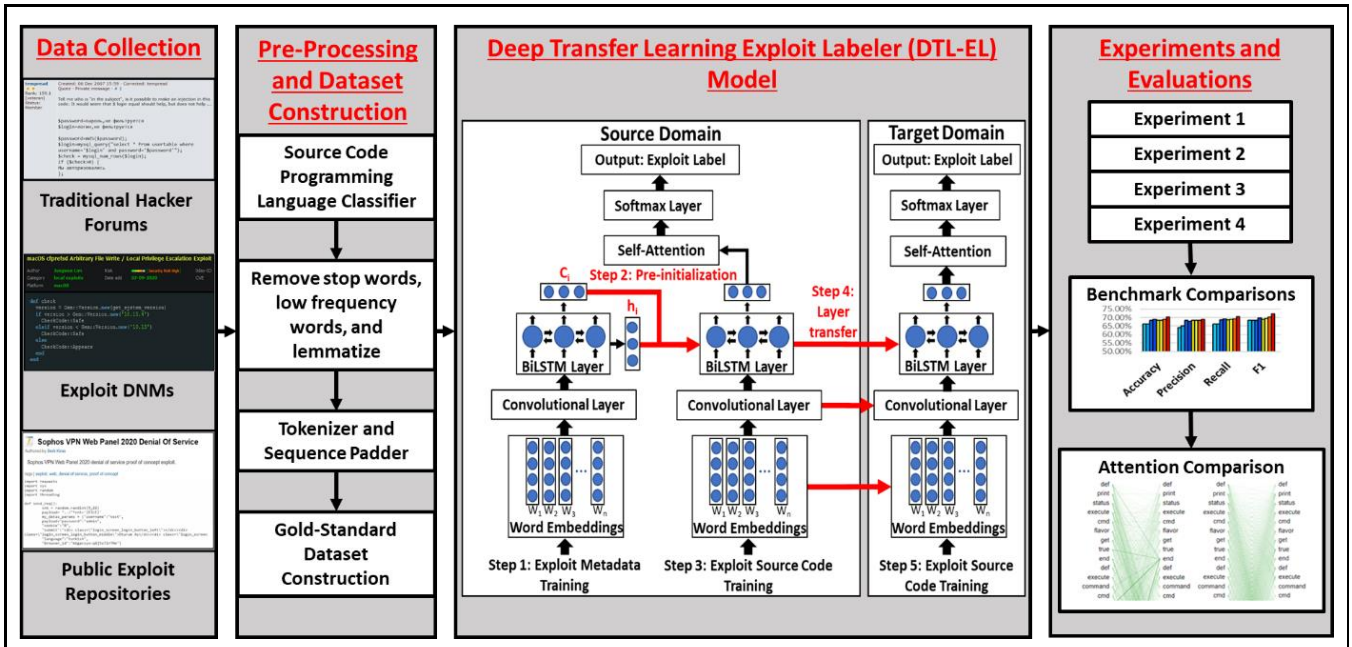

Figure 3. Proposed Research Design

Table 4. Summary of Key Metadata Available in Exploit-Specific Platforms

Category	Feature	Description	Example	Hacker Forum	Exploit DNM	Public repository
Description	ID	A unique post identifier	EDB-ID: 812	✓	✓	✓
	Title	Exploit header	inoERP 4.15 SQL injection	✓	✓	✓
	Exploit source	Where the exploit was collected from	exploitDB	X	X	✓
	Date	Postdate of exploit	26-Sep-19	✓	✓	✓
Author	Author	The person who posted the exploit	Alexandrovich Lyhin	✓	✓	✓
	Reputation score	Respect for the author in the community	3/5 stars	✓	✓	X
Content	Post	A short paragraph explaining what the code does	inoERP version 4.15 suffers from a remote SQL injection vulnerability.	✓	✓	✓
	Discussion	Comments that follow the posting of the code	This still worked for me on Adobe 13.03	✓	✓	X
	Source code	The code of the exploit itself	def generatePayload(query): b64_query	✓	✓	✓
Operation	Attack type	Categorizes the code based on its operations	Local, remote, SQL injection	X	✓	✓
	Platform	System exploit targets	Windows, Apple, Linux	X	✓	✓

Table 5. Summary of Research Testbeds

Platform type	Platform name	Language	Start date	End date	# of code snippets	# of unique authors
Hacker forums	0x00sec	English	4/13/2017	7/15/2020	397	1,004
	Altenens	English	3/22/2010	4/1/2020	1,403	580,220
	AntiChat	Russian	4/1/2004	7/15/2020	64,890	84,143
	AntiOnline	English	4/10/2002	7/15/2020	2,063	13,017
	Cipher	English	5/1/2015	7/15/2020	2,207	3,551
	Go4expert	English	12/25/2004	7/15/2020	5,800	15,213
	PersianTools	Persian	8/18/2015	4/1/2020	528	19,360
	WWHClub	Russian	2/6/2014	7/15/2020	53	133,598
	WildersSecurity	English	2/8/2002	7/15/2020	2,096	127,103
Summary	**9 Forums**	**3 Languages**	**2/8/2002 – 7/1/2020**		79,437	977,209
Exploit DNM	**0day.today**	**English**	**1/1/1996**	**4/1/2020**	33,766	6,052
Public exploit repository	Seebug	English	12/12/2001	4/1/2020	56,657	291
	ExploitDB	English	8/1/1988	4/1/2020	43,120	7,814
	PacketStorm	English	8/17/1999	4/1/2020	39,433	7,102
	Metasploit	English	10/12/2005	4/1/2020	4,040	1
	Vulnerlab	English	7/14/2009	4/1/2020	1,635	525
	Zeroscience	English	7/8/2008	4/1/2020	651	18
Summary:	**6 Repositories**	**English**	**1/1/1988 - 7/1/2020**		145,536	15,751
Total:	**16 Sources**	**3 Languages**	**1/1/1988 - 7/1/2020**		258,739	999,012

Training and evaluating a supervised DTL model requires a source and target domain dataset (Zhu et al., 2020). In this study, the source domain dataset was created from collected exploits in exploit DNMs and public exploit repositories that contained an exploit label. These data sources were chosen for the source domain, as they were carefully curated and reviewed by cybersecurity domain experts and contain rich metadata such as descriptive titles and exploit labels (Samtani et al., 2020). The target domain consists of hacker-forum exploit source code posts.

The ground-truth target domain dataset was constructed in three steps. First, we carefully defined keywords for each exploit label in the source domain to retrieve relevant exploits from hacker forums based on each post's thread title and content[2]. Second, exploit source code snippets of fewer than 100 characters in length were omitted, as these often contain irrelevant information (e.g., IPs for proxies). Third, we manually verified the remaining data and discarded irrelevant content. The source and target domain datasets are summarized by the exploit label in Table 6.

The source domain dataset contained 96,333 labeled exploits in eight exploit label categories. For our target domain dataset, the preprocessing steps reduced the 79,437 unlabeled hacker forum source code snippets in our research testbed to 4,842 labeled exploit source code snippets. The dataset reduction is attributable to the lack of related metadata in hacker forums that allows targeted keyword matching (thus further motivating our proposed approach). Keyword matching requires a predefined lexicon which is often time-consuming to develop and maintain. This time cost is pronounced in hacker community research, as terminology is constantly evolving and changing (Samtani et al., 2020). Moreover, direct keyword matching can often fail due to small content mismatches (e.g., term variations or misspellings) (Samtani et al., 2022). Even with the significant reduction in dataset size, the target domain dataset exceeds the size of the testbeds used in related IS studies (Benjamin et al., 2019; Samtani et al., 2022; Samtani et al., 2017). The most considerable disparity in our domains is in the web applications category, which is the most common exploit type in the source domain but the second least common in the target domain.

[2] We conducted preliminary analysis to assess keyword matching as a viable exploit labeling strategy. In this analysis, 100 exploits were separately labeled by two experts with over half a decade of experience in CTI, dark web analytics, and exploit analysis. Keywords were generated based on common tags provided to exploits by cybersecurity domain experts. The initial Cohen's kappa between the ratings was 0.88. The raters met after the first round of labeling to resolve differences and attained 100% agreement on exploit labels. Compared to DTL-EL, the keyword-matching approach correctly labeled fewer exploits (37 vs. 56), incorrectly labeled more exploits (8 vs. 2), and was unable to label most exploits (55 vs. 42). These results indicate that a keyword-based approach alone attains suboptimal exploit labeling performance.

Table 6. Source and Target Domains in Ground-Truth Exploit Dataset		
Exploit label	**Source domain count**	**Target domain count**
Web Applications	43,475	57
DoS	12,121	714
Remote	11,787	672
Local	7,993	1,952
SQL Injection	7,187	702
XSS	7,025	485
File Inclusion	3,412	29
Overflow	3,333	231
Total	**96,333**	**4,842**

Table 7. Source Code Metrics by Domain in the Ground-Truth Dataset			
Domain	**ASCL**	**HAE**	**MACC**
Source (exploit DNMs and public repository)	24.93***	371.24***	6.67***
Target (hacker forums)	15.48	215.79	4.14

Note: $*: p < 0.05$ $**: p < 0.01$ $***: p < 0.001$

Extant literature suggests proving a systematic difference between source and target domain datasets to rule out other DTL types (e.g., mapping) or learning paradigms (e.g., incremental learning) (Zhuang et al., 2020). Consistent with best practices in source code analysis literature, we calculated the average source code length (ASCL), Halstead average effort (HAE), and McCabe average cyclomatic complexity (MACC) for each domain (Nuñez-Varela et al., 2017). ASCL measures the average lines of code. HAE measures the difficulty of developing a piece of source code based on the number of unique operands and operators in the source code. MACC measures the source code's average number of control flow statements (e.g., if, else, for). These measures were chosen as they are seminal and language-agnostic. The Radon Python package (Lacchia, 2020) was used to calculate each metric for each ground-truth domain. Consistent with source code analysis literature, a one-tailed *t*-test was conducted to measure statistically significant differences between domains (Kapllani et al., 2020). The results of our analysis appear in Table 7.

Code in the source domain has a longer ASCL (24.93 vs 15.48 average lines), a higher HAE (371.24 vs 215.79), and a higher MACC (6.67 vs 4.14 average control flow statements) than our target domain. These results are significant at $p < 0.001$ and suggest systematic differences in the coding practices of the source and target domain datasets. More specifically, the results suggest that our source domain dataset is longer, more difficult to code, and more complex on average than our target domain dataset. Since our dataset consists of significantly distinct and labeled source and target domains, we selected network-based DTL for our task of exploit labeling.

Deep Transfer Learning Exploit Labeler (DTL-EL) Model

The proposed DTL-EL is a supervised network-based DTL model that trains and transfers the layers of a BiLSTM model with pre-initialized hidden and cell states from professionally vetted exploits (source domain) to a BiLSTM with self-attention designed to label hacker-forum exploit source code in a target domain. The proposed DTL-EL model is presented in Figure 4.

The DTL-EL model follows a five-step procedure for labeling source code from hacker forums. A sketch of our proposed DTL-EL model is presented below:

Step 1 (exploit metadata training): An exploit title BiLSTM model is trained using professionally vetted exploit titles (metadata) from exploit DNMs and public exploit repositories as input. At $t = 0$, The hidden state $h_{t=0}$ and cell state $C_{t=0}$ memories start at 0 and output a concatenated $h_i^{meta} = \left[\overleftarrow{h_t}\ \overrightarrow{h_t}\right]^T$ and $C_i^{meta} = \left[\overleftarrow{C_t}\ \overrightarrow{C_t}\right]^T$.

Step 2 (pre-initialization design): An exploit source code BiLSTM model is pre-initialized at $t = 0$ with the hidden states and long-term memories obtained from the exploit title BiLSTM: $h_{t=0} = h_i^{meta}$ and $C_{t=0} = C_i^{meta}$. The hidden states and long-term memories are not static (i.e., are trainable) and are updated during the model training process.

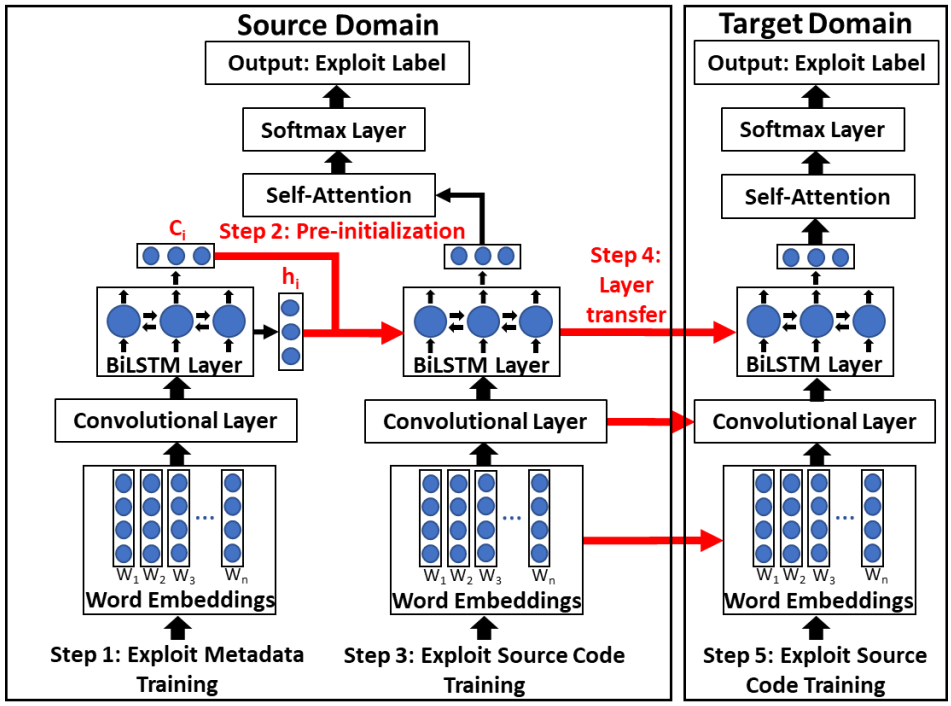

Figure 4. Illustration of the Proposed DTL-EL Model

Step 3 (source domain exploit source code training): The pre-initialized exploit source code BiLSTM model is trained using professionally vetted exploit DNM and public exploit repository exploits to learn a representation for the source domain task of labeling hacker forum exploits. Our classifier uses the softmax function, $\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$, where $\vec{z}$ is the input vector given from the self-attention mechanism, i is the i^{th} class, and K is the number of classes. The output of the model is a probability distribution where the input is labeled with the class having the highest probability in the distribution.

Step 4 (layer transfer): Consistent with recent IS literature, we implemented a multi-layer transfer learning (MLTL) design from $\mathcal{D}_S$ to $\mathcal{D}_T$ (Zhu et al., 2020). MLTL is chosen due to the heterogeneity of our layer weights: The pre-initialization design updates the BiLSTM and self-attention layer weights but does not directly update convolutional and embedding layers. A new exploit source code BiLSTM model in the target domain ($\mathcal{D}_T$) (i.e., for hacker-forum exploit source code) is constructed by reusing n layers from the trained source-domain exploit source code BiLSTM model. Following best practices in DTL literature, we fine-tune the weights for all of the reused layers since $\mathcal{D}_T$ contains labeled data (Mou et al., 2016).

Step 5 (target domain exploit source code training): The target domain model is trained using ground-truth hacker exploit source code as input to adapt the feature representation from $\mathcal{D}_S$ to $\mathcal{D}_T$. The training process is the same as Step 3.

The input word embeddings for each BiLSTM model were created with GloVe, a prevailing context-free embedding technique that can learn the global statistic information of input sequences and is robust to long sequences compared to other context-free models (e.g., Word2Vec) (Kowsari et al., 2019). GloVe was chosen over contextual embedding models (e.g., BERT) because the performance of contextual embeddings often degrades with noisy text (e.g., text in hacker forums) (Srivastava et al., 2020). Consistent with best practices in text classification literature, the embedding vectors produced by GloVe were inputted into a convolutional layer with a kernel size of 3 and a rectified linear unit (ReLU) activation function (Yenter & Verma, 2017). The convolutional layer can capture and engineer local features by focusing on word combinations in the size of the kernel (e.g., kernel size of 3 means the convolutional layer learns trigrams). The hybrid convolutional-BiLSTM model has significantly outperformed BiLSTM models on benchmark text classification tasks (e.g., sentiment analysis) by learning local and low-dimensional vectors for each input (Yenter & Verma, 2017).

Since hybrid DL models are prone to overfitting and can become unstable without proper tuning and construction (Liu & Guo, 2019), we implemented a dropout layer to improve generalizability and a batch normalization layer to stabilize the model by reducing internal covariate shifts (Ioffe & Szegedy, 2015). To help attain consistent performances, we combined the dropout and batch normalization layers with a nonadaptive optimizer (Chen et al., 2019). We also fine-tuned GloVe embeddings to stabilize our input embedding layer. Fine-tuned GloVe embeddings (learned across training) have significant benefits in text classification tasks when combined with convolutional, BiLSTM, and attention layers (Son et al., 2019). The full details of our parameter settings and embeddings are detailed in Appendix A.

The key novelty in our proposed DTL-EL is the (trainable) pre-initialization design. The DL models developed in past hacker forum analytics studies did not pre-initialize hidden and cell states for their cybersecurity tasks (Ebrahimi et al., 2022). Pre-initialization can boost classification performance and learn representations of the input data missed by non-initialized models (Peng & Dredze, 2017). Since concatenating the title and the exploit source code as input may cause the model to overfit due to overly descriptive titles, our pre-initialization design followed a multitask learning (MTL) approach, where the exploit source code BiLSTM learns from the exploit title BiLSTM. MTL approaches have been used in IS literature to improve classification performance (Lin et al., 2017). However, in contrast to the traditional MTL paradigm, we did not leverage the final output of the exploit title BiLSTM. We compared a standard LSTM cell within the BiLSTM for source code processing and our proposed pre-initialized LSTM cell in Figure 5.

In the pre-initialized LSTM cell (the right side of Figure 5), $h_i = \left[\overleftarrow{h_i}\ \overrightarrow{h_i}\right]^T$ and $C_i = \left[\overleftarrow{C_i}\ \overrightarrow{C_i}\right]^T$ represent the concatenated forward and backward hidden and cell state vectors of the exploit title BiLSTM model in $\mathcal{D}_S$. Then, h_i and C_i were used to pre-initialize $h_{t=0}$ and $C_{t=0}$ of a new and untrained exploit source code BiLSTM. The pre-initialized BiLSTM was then trained on the exploit DNM and public exploit repository exploit source code and label (e.g., SQL injection).

In addition to including the pre-initialized BiLSTM into DTL-EL, we incorporated a self-attention mechanism into each BiLSTM in the DTL-EL to process sequences that appear within exploit code data while considering the context of the code for each timestep. To the best of our knowledge, no IS study has implemented a self-attention mechanism that takes pre-initialized hidden states as input. However, past IS cybersecurity analytics literature has leveraged self-attention mechanisms on exploit content (specifically titles) to improve model performance (Samtani et al., 2022). A self-attention mechanism can help improve exploit labeling performance by capturing long-range semantic relationships (e.g., a function called lines after it is defined) while simultaneously differentiating between labels (Liu, 2020). Therefore, we implemented the BiLSTM with self-attention model, where all queries and key-value pairs are attained from the output and hidden states in the previous BiLSTM layer, respectively (Liu & Guo, 2019).

After source domain training, the embedding, convolutional, BiLSTM, and/or attention layers were transferred to the target domain model to label each hacker forum exploit. In our design, $\mathcal{D}_T$ did not contain a pre-initialization design in the BiLSTM layer, as hacker forum thread titles were not consistently available or indicative of an exploit's intended purpose (Samtani et al., 2017). However, the weights transferred from the BiLSTM and attention layers in $\mathcal{D}_S$ were calculated with our pre-initialization design. The exact layers that are transferred were determined through an ablation analysis, which is further detailed in the next subsection.

Experiments and Evaluations

Consistent with the computational design science paradigm (Rai, 2017), we rigorously evaluated our proposed DTL-EL artifact with a series of technical benchmark experiments. Each experiment's goal, model types, benchmark models, and evaluation metrics appear in Table 8.

Experiment 1 aimed to identify if our pre-initialization design and added self-attention mechanism in the source domain DTL-EL improved exploit labeling performance over benchmark models. Additionally, IS literature (Zhu et al., 2020) and fundamental DTL principles (Zhuang et al., 2020) recommend evaluating a source domain model to find the best-performing model to transfer features to a target domain. Therefore, we evaluated whether transferring the knowledge learned from exploit DNMs and public exploit repositories (DTL-EL) improved exploit labeling when compared to non-DTL approaches in Experiment 2. Two sets of benchmark models were used in Experiments 1 and 2: (1) classical ML models that included naive Bayes, logistic regression, decision tree, SVM, extreme gradient boosting (XGBoost), and light gradient boosting machine (LightGBM) and (2) DL-based models that included RNN, GRU, LSTM, BiLSTM without self-attention, a pre-initialized BiLSTM, and BiLSTM with self-attention. These models are commonly used in past hacker forum analytics literature (Ebrahimi et al., 2018; Goyal et al., 2018; Samtani et al., 2017). In Experiment 2, we trained three variations of each model: one with the target domain dataset, one with the source domain dataset, and a concatenated dataset of the source and target domains. We kept the same target domain validation dataset for each variation.

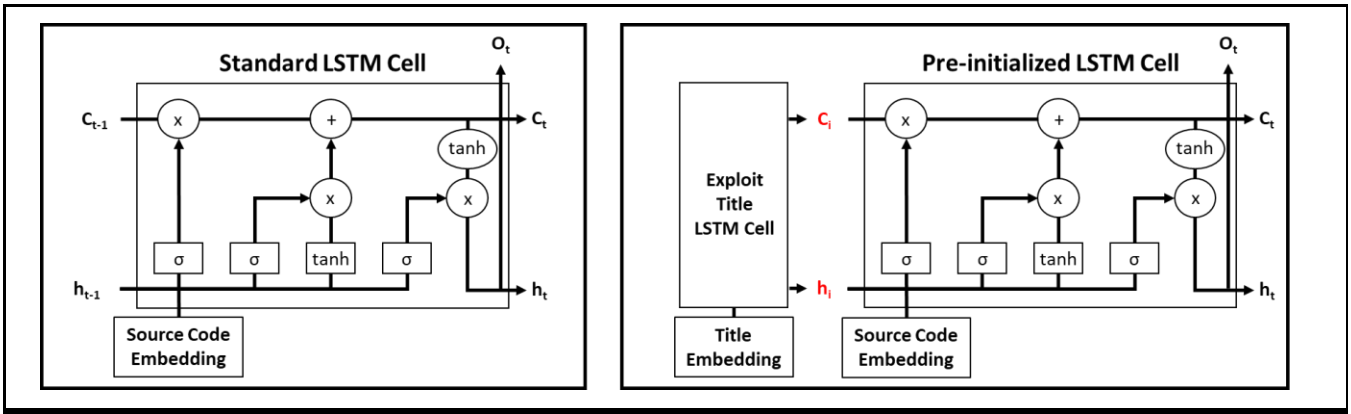

Figure 5. Standard LSTM Cell for Text Processing (Adapted from Thangaraj & Sivakami, 2018) (left) and Our Proposed Pre-Initialized LSTM Cell (right)

Table 8. Summary of Experiment Designs

#	Experiment	Goal	Type	Benchmarks	Metrics	References
1	DTL-EL against prevailing classification methods on the source domain	An evaluation that our pre-initialization design and self-attention mechanism in the source domain.	Classical machine learning	Naive Bayes, logistic regression, decision tree, SVM, XGBoost, LightGBM	Accuracy, precision, recall, F1-score	Ebrahimi et al., 2018; Goyal et al., 2018; Samtani et al., 2017; Tan et al., 2018; Zhuang et al., 2020
			Deep learning	RNN, GRU, LSTM, BiLSTM, BiLSTM with self-attention		
2	DTL-EL against non-transfer learning approaches on the target domain	An evaluation to find differences between DTL, prevailing classical machine learning, non-DTL approaches, and dataset variations.	Classical machine learning	Naive Bayes, logistic regression, decision tree, SVM, XGBoost, LightGBM		
			Deep learning	RNN, GRU, LSTM, BiLSTM, BiLSTM with self-attention		
3	DTL-EL against alternate transfer learning approaches	An evaluation to rule out superior design within DTL literature.	Transfer learning	Adaptive SVM, parameter sharing, adversarial, BERT		Houlsby et al., 2019 Liu et al., 2017; Peng & Dredze 2017; Peng et al., 2008
4	DTL-EL against transfer learning layer selection on the target domain	Ablation analysis to identify the value of transferring different layers for DTL-EL with and without pre-initialization.	Layer selection	DTL-EL: Embedding, CNN, LSTM, attention layers		

Note: BiLSTM = bidirectional long-short term memory; CNN = convolutional neural network; DTL-EL = deep transfer learning exploit labeler; GRU = gated recurrent unit; LightGBM = light gradient boosting machine; LSTM = long-short term memory RNN = recurrent neural network; SVM = support vector machine; XGBoost = eXtreme gradient boosting.

Our proposed DTL-EL model incorporates a BiLSTM with self-attention mechanism in the target domain. Consequently, the weights assigned to the input features from the self-attention during the labeling process can be visualized to explain how the model reached its output prediction. Explainability in our task of exploit labeling is defined as how well our model identifies tokens that consistently make up each exploit label (Wiegreffe & Pinter, 2019). However, there is debate on whether self-attention mechanisms truly provide explainability in NLP tasks (Jain & Wallace, 2019). Therefore, we performed an adversarial test on the DTL-EL for both Experiments 1 and 2 to find out whether the self-attention mechanism found meaningful tokens in our inputs for each output. To perform this task, we implemented an adversarial experiment where we compared the self-attention weights of DTL-EL with weights learned from an adversarial model (adversarial DTL-EL). The goal of the adversarial model is to obtain similar prediction scores as DTL-EL with a

different attention weight distribution. Adversarial attention weights were learned using the loss function proposed by Wiegreffe and Pinter (2019):

$$\mathcal{L}(\mathcal{M}_a, \mathcal{M}_b)^{(i)} = TVD\left(\hat{y}_a^{(i)}, \hat{y}_b^{(i)}\right) - \lambda KL\left(\alpha_a^{(i)} \middle|\middle| \alpha_b^{(i)}\right),$$

where $\mathcal{M}_b$ is the DTL-EL base model, $\mathcal{M}_a$ is the adversarial model, $\hat{y}^{(i)}$ are the predictions, and $\alpha^{(i)}$ are the attention distributions. A model with good explainability would perform better in their specified tasks (e.g., exploit labeling) than their adversarial variation.

Although a network-based DTL approach was ideal for our context, we were interested in how seminal DTL approaches compared to the proposed DTL-EL. Therefore, in Experiments 3 and 4, we explored the boundaries of our network-based implementation of DTL. In Experiment 3, we evaluated transfer learning with the popular classical ML model, SVM (Peng et al., 2008), MTL (Peng & Dredze, 2017), adversarial learning (Liu et al., 2017), and an adapted bidirectional encoder representations from transformers (BERT) model (Houlsby et al., 2019). In Experiment 4, we performed an ablation analysis on the effect of layer transfer from a pre-initialized and non-pre-initialized source domain model. We performed this analysis from a single- and multi-layer transfer perspective.

Since the source domain dataset is imbalanced (45.13% of the dataset belongs to web applications), accuracy alone is not a viable performance measure (Ebrahimi et al., 2022). Therefore, we included precision, recall, and F1-score (harmonic mean of precision and recall) as metrics to evaluate each model's exploit labeling performance in each experiment. Each metric was computed using true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The formulas for each metric are as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}, F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Among the four metrics, scholars conducting IS cybersecurity research suggest that the F1-score is the best metric for comparing models, as it is not sensitive to data imbalance (Ebrahimi et al., 2022). The reported metrics for each model are a weighted average across each class label based on the support (i.e., count) of each class. This weighted average formula is:

$$Weighted\ Average = \frac{\sum_i S_i M_i}{N},$$

where S_i is the support for a class label, M_i is the calculated metric for the class label (e.g., accuracy), and N is the total number of samples. One-tailed paired *t*-tests were used to evaluate statistically significant differences between the proposed approach and benchmarks. Our source and target domain datasets were split into a training and testing dataset wherein all exploits in the testing set are newer than those in the training datasets in both domains. Exploits posted before 2019 were placed in the training dataset, while exploits posted in 2019 or later were placed in the testing dataset. We chose this split to provide enough data to our testing set in both domains and test whether the DTL-EL model could label new and unseen exploits. In the source domain, 80,582 exploits were used for training (83.63% of the dataset) and 15,751 exploits were used for testing (16.35% of the dataset). In the target domain, 3,902 exploits were used for training (80.58% of the dataset) and 940 exploits were used for testing (19.42% of the dataset). We implemented a stratified 10-fold cross-validation split to conduct model training.

Results and Discussion

Experiment 1: DTL-EL against Prevailing Classification Methods on the Source Domain

Experiment 1 compared our proposed DTL-EL (a pre-initialized BiLSTM with attention) against classical ML and DL benchmarks on the source domain dataset. We also compared BiLSTM models with and without a self-attention mechanism. The accuracy, precision, recall, and F1-score for each model are presented in Table 9.

Our proposed DTL-EL, which combines a BiLSTM, our pre-initialization design, and a self-attention mechanism, outperformed all classical ML and DL models in accuracy (90.75%), precision (91.12%), recall (90.83%), and F1-score (90.91%). The F1-score for the classical ML methods ranged from 57.39% for naive Bayes to 78.82% for LightGBM. However, the LightGBM and other ML models were quickly overfit on the training data, suggesting that ML models are not complex enough to fully capture exploit source code representations. All DL methods reached higher F1-scores than LightGBM. RNN outperformed LightGBM by 1.30% (from 78.82% to 80.12%); however, the F1-score increased when the RNN was replaced with a GRU (85.80%), LSTM (86.20%), or BiLSTM (86.62%) layer. One possible explanation for this improvement is that GRU, LSTM, and BiLSTM include a gating mechanism to fix the vanishing gradient problem that RNNs suffer from when processing long sequences (e.g., exploit code) (Liu & Guo, 2019).

Table 9. Experiment 1: DTL-EL against Prevailing Classification Methods on the Source Domain

Model Type	Model	Initial State	Accuracy	Precision	Recall	F1-score
Classical Machine Learning	Naive Bayes	N/A	58.12%***	57.97%***	56.82%***	57.39%***
	Logistic regression	N/A	66.01%***	66.23%***	70.86%***	68.12%***
	Decision tree	N/A	70.55%***	70.05%***	67.14%***	68.87%***
	SVM	N/A	78.21%***	79.86%***	77.02%***	78.43%***
	XGBoost	N/A	83.54%***	78.96%***	78.29%***	78.53%***
	LightGBM	N/A	83.83%***	78.98%***	78.82%***	78.82%***
Deep Learning	RNN	Zero	80.12%***	82.01%***	78.31%***	80.12%***
	GRU	Zero	86.54%**	86.81%**	84.31%**	85.80%**
	LSTM	Zero	86.22%***	86.48%***	85.93%***	86.20%***
	BiLSTM	Zero	87.14%**	86.76%**	86.37%**	86.62%**
	BiLSTM with self-attention	Zero	87.95%**	87.29%**	86.74%**	87.23%**
	Pre-Initialized BiLSTM	Pre-Initialized	88.21%**	88.98%*	87.62%**	88.29%**
Proposed DTL-EL		Pre-Initialized	**90.75%**	**91.12%**	**90.83%**	**90.91%**

Note: $*: p < 0.05$ $**: p < 0.01$ $***: p < 0.001$. Top scores are highlighted in boldface.

Adding a self-attention mechanism to the BiLSTM marginally increased the F1-score (87.23%) over the best-performing DL algorithm without an attention mechanism, BiLSTM (86.62%). The self-attention mechanism looks at the hidden states of each BiLSTM cell, capturing important aspects of the input sequence. Incorporating the proposed pre-initialization design into the BiLSTM with no self-attention mechanism further increased BiLSTM's performance from 87.41% to 88.29%. The difference is statistically significant at $p < 0.05$. These results suggest that using the exploit title to pre-initialize the BiLSTM's hidden and cell states can improve exploit source code categorization. The DTL-EL attained higher F1-scores than the BiLSTM with a self-attention mechanism (from 87.23% to 90.91%) and the pre-initialized BiLSTM without a self-attention mechanism (from 88.29% to 90.91%). The differences in both cases were statistically significant at $p < 0.01$. These results indicate that the self-attention mechanism can identify important exploit features from the pre-initialized hidden states.

In addition to evaluating the performance of DTL-EL against benchmark methods in the source domain, we implemented the proposed adversarial test to identify if the self-attention mechanism identified (i.e., weighed) tokens that contributed to the model's output (Wiegreffe & Pinter, 2019). The adversarial DTL-EL model obtained an F1-score of 76.34%. This score is 14.62% lower than the F1-score of the DTL-EL model (90.91%). The steep decline in the F1-score suggests the adversarial weights lose essential information needed to label exploit source code. These results also indicate that our self-attention mechanism focused on the most valuable tokens for each exploit label and attained the best labeling performance.

Experiment 2: DTL-EL against Non-Transfer Learning Approaches on the Target Domain

Experiment 2 evaluated whether the features extracted from the source domain improve the classification performance of DTL-EL in the target domain. We evaluated our proposed DTL-EL model against state-of-the-art classical ML and DL benchmarks on the target domain ground-truth dataset (hacker forum exploits). We compared training models using the source domain, the target domain, and both training datasets concatenated (i.e., combined). All models used the same target domain validation dataset. The accuracy, precision, recall, and F1-score for each model are summarized in Table 10. The performances for each model in each of the eight exploit categories are presented in Appendix B.

For each model, the performance for all tracked metrics was the highest on the target domain dataset, followed by source + target and source. The performance differences may be due to the fundamental coding differences (e.g., ASCL) between professionally vetted exploits (source domain) and hacker forum exploits (target domain). These coding differences in the source domain dataset may have prevented each model from generalizing to the target domain evaluation dataset. Given these results, we discuss the results of the models trained on the target domain only.

The four classical ML methods attained an F1-score between 13.45% (naive Bayes) and 46.32% (LightGBM). This F1-score range for classical ML models is lower than the range seen in Experiment 1 on the source domain because hacker-forum exploit source code is often less structured than professionally vetted exploits (shown previously in Table 7). Our proposed

DTL-EL improved on the best-performing classical ML model (LightGBM) in the F1-score by 24.02% (from 46.32% to 70.34%), and this difference was statistically significant at $p < 0.001$. These results suggest that the complex DL models can capture latent and more representative features of exploit content better than the ML models (which were overfit in the imbalanced classification setting).

All DL methods outperformed the classical ML methods in the F1-score. RNN achieved the lowest F1-score at 55.12%. Similar to the results from Experiment 1, the F1-score increased when the RNN layer was replaced with a GRU (60.69%), LSTM (60.02%), or BiLSTM (60.71%) layer. Adding a self-attention mechanism to the BiLSTM marginally increased the F1-score from 60.71% to 62.52%. However, DTL-EL outperformed the BiLSTM with attention and no DTL layers (from 62.52% to 70.34%, F1-score), and the difference was statistically significant at $p < 0.001$. These results indicate that identifying and transferring layers from a metadata-rich hacker exploit source domain to a target domain significantly outperforms the single dataset-based model approaches prevalent in extant literature (Ebrahimi et al., 2018; Williams et al., 2018). Transferred pretrained layers leading to stronger performances than random initialization in similar tasks is consistent with the seminal literature (Yosinski et al., 2014). The performance gain may be attributable to the source domain model learning generalized information and inductive bias (i.e., model assumptions when making a prediction) that is being transferred to our DTL-EL model (Li et al., 2018).

Similar to Experiment 1, we performed an adversarial experiment for DTL-EL. The adversarial DTL-EL model obtained an F1-score of 43.76%—26.58% lower than the DTL-EL (70.34%). These results suggest that our self-attention mechanism finds the most valuable tokens for each exploit label to improve classification performance. In Figure 6, we illustrate sample exploits that DTL-EL correctly identified but the best competing approach (BiLSTM with self-attention) missed. Specifically, we visualized the semantic relationships between input tokens for DTL-EL and the BiLSTM with self-attention on a remote exploit (operates over a network without direct machine access) and local exploit (requires machine access). These lines demonstrate a simplified control flow of how the code operates. We present an excerpt (for space considerations) of each exploit's code in each category at the top of Figure 6. Thicker lines indicate stronger (i.e., higher weighted) semantic relationships between two tokens in the input exploit source code.

As shown in Figure 6, DTL-EL developed higher-weighted (thicker lines) semantic relationships (i.e., dependencies) between specific tokens for remote and local exploits. In contrast, the BiLSTM with the self-attention model weighted most of the relationships between tokens nearly identically. In the remote exploit, DTL-EL found long-term dependencies between "def" and "end," which are the beginning and end of the exploit function, respectively. Additionally, DTL-EL found strong dependencies on the "authenticate" token, which is vital for accessing machines in remote exploit attacks. The BiLSTM with the self-attention model did not find strong dependencies between tokens, leading to it incorrectly assigning a label of "denial of service" to the code instead of the correct "remote" label. In the local exploit, DTL-EL found strong semantic relationships between tokens such as "process" and "executable," "cmd" and "grep," and "def" and "config." Using the grep Unix command in the command line (cmd) is a common local privilege escalation technique. As with the previous example, the BiLSTM with the self-attention model did not find these dependencies; instead, it classified the exploit as a "web application" instead of a "local" exploit.

We conducted a centered kernel alignment (CKA) analysis to further identify the internal differences between the source and target domain model and identify the specific layers that were affected as a result of transfer learning (Kornblith et al., 2019)[3]. The results of our analysis (presented in Appendix C) suggest that the lower-level layers of the DTL-EL model (embedding, convolution, pooling, batch norm, and dropout) are more similar to higher-level layers (BiLSTM, self-attention, dropout, dense) than the non-DTL BiLSTM with self-attention. These differences in similarities demonstrate that the transfer and fine-tuning of source domain layers creates a more closely linked internal feature representation than a non-DTL approach and therefore potentially improves performance and results in differences in attention weighting.

Experiment 3: DTL-EL against Alternate Transfer Learning Approaches

For Experiment 3, we explored the results of four types of transfer learning. We implemented an adaptive SVM that uses the hinge loss function and an L2 regularization term to adapt a classifier from a source domain for the target domain (Peng et al., 2008). We then implemented two types of MTL: hard and soft. In hard MTL, we merged both domain inputs after the embedding layer and had the same model layers from the convolutional layer through the BiLSTM layer with separate

[3] CKA measures the similarities between internal feature representations of model layers. Comparing similarities across models trained on the same datasets can help explain differences in model performance.

output layers. In soft MTL, we implemented two separate BiLSTM models with a custom loss function that used the sum of categorical cross-entropy, mean-squared error, and cosine proximity between the true exploit label and predicted exploit label to minimize the distance between weights of the two models. Our adversarial DTL used the same design as Liu et al. (2017), wherein a shared-private model with domain discriminators for each feature was defined and a custom adversarial loss function was implemented. Our implementation of adapter-based transfer learning followed the fine-tuning of the BERT model in Houlsby et al. (2019). The accuracy, precision, recall, and F1-score of each transfer learning design are presented in Table 11.

Overall, DTL-EL had the highest performance in accuracy (72.11%), precision (70.57%), recall (70.15%), and F1-score (70.34%). The difference in performance was statistically significant against all benchmark models. MTL approaches outperformed classical ML models on all four metrics. Within the MTL paradigm, soft parameter sharing (64.02%) outperformed hard parameter sharing (61.09%) in terms of the F1-score. Hard parameter sharing works well when the source and target tasks are similar. However, our source and target domains were dissimilar enough to cause decreases in performance. Soft parameter sharing can alleviate this issue through feature sharing. Adversarial DTL (62.72%) outperformed hard parameter sharing, possibly because it maximizes the training error with a reversed gradient. However, the adversarial approach performed worse than the soft parameter sharing model. Finally, BERT attained a higher F1-score (65.31%) than MTL approaches, the adversarial approach, and classical ML but underperformed compared to DTL-EL on the target hacker forum dataset. This suggests that pretrained contextual models like BERT may be too general for our dataset, and a more targeted approach is needed (Srivastava et al., 2020).

Experiment 4: DTL-EL against Transfer Learning Layer Selection on the Target Domain

In Experiment 4, we explored combinations of transferred layers from the source domain to the target domain. The embedding, convolutional, BiLSTM, and self-attention layers had transferable weights and features in the source domain DTL-EL model. We first evaluated transferring each layer individually. This included trained embeddings from Word2Vec, GloVe, and BERT to evaluate the best transfer performance of unsupervised embeddings. We then evaluated an MLTL approach, which is common in IS literature (Zhu et al., 2020) and recommended when performing homogeneous (e.g., source and target domain datasets share attributes) transfer learning (Yosinski et al., 2014). Since our pre-initialization design is a core novelty of DTL-EL, we also compared layer transfer from the source domain BiLSTM with the self-attention model without pre-initialized hidden and cell states. Each model was trained on the target domain dataset used in Experiment 2. The accuracy, precision, recall, and F1-score are summarized in Table 12.

For each transfer type and layer, the transferred pre-initialized layers always performed better than the non-pre-initialized layers. Single-layer transfer with pre-initialization attained F1-scores between 64.56% to 68.16%, with the GloVe embedding layer performing the best. The GloVe embedding holds global information about the word vectors of professionally vetted exploits that Word2Vec and BERT may have missed. These results suggest that our dataset was too noisy for the embeddings from large contextual models to adequately generalize (Srivastava et al., 2020). Therefore, we only considered the GloVe embedding in our MLTL design.

Within MLTL, we found that adding additional layers to the GloVe layer provided higher F1-scores. The GloVe layer with a convolutional layer (68.69%) or BiLSTM layer (69.31%) attained better scores than all single-layer models and the model with a transferred convolutional, BiLSTM, and attention layer (68.41%). However, using the GloVe, convolutional, BiLSTM, and attention layers from the source domain DTL-EL model led to the highest accuracy (72.11%), precision (70.57%), recall (70.15%), and F1-score (70.34%). The differences in the F1-score over the second-best layer transfer technique (GloVe, convolutional, and attention) were significant at $p < 0.05$.

Practical Implications and Contributions to the IS Knowledge Base

Practical Implications

Recent IS cybersecurity studies have identified three cybersecurity stakeholders that can benefit from IT artifacts equipped with advanced cybersecurity analytics: (1) cybersecurity managers, (2) educators, and (3) analysts (Yue et al., 2019). Past IS studies have frequently integrated a novel algorithm into a system with a user interface to help stakeholders access the algorithm and results (Samtani et al., 2017). Although not the focus of our study, we implemented a system with DTL-EL to illustrate an example of such an implementation. Details about this system are presented in Appendix D. We further elaborate on the practical implications of the proposed DTL-EL framework for each stakeholder below.

Cybersecurity managers: Cybersecurity managers often require automatically generated and easily digestible reports and visualizations to determine the best course of action for their organization's cyber policy (Samtani et al., 2020). Chief information security officers (CISOs) are typically responsible for allocating security investments and resources. However, CISOs can often be overwhelmed and make suboptimal decisions due to the large quantities of unstructured information available from external resources (Alomar et al., 2020). The results produced by the proposed DTL-EL can be carefully synthesized to create dynamic visualizations of exploit trends and summary statistics to reduce the strain on a CISO. For example, suppose a CISO sees a sharp increase in web application exploits posted in hacker communities. In that case, they can invest resources internally (e.g., assign cyber analysts to focus on web application exploits) and externally (purchase software and additional protections for their web application servers). Furthermore, cybersecurity managers can customize their organization's DTL-EL implementation based on their cyber-risk profile. For example, DTL-EL can be tuned to minimize cost instead of error using a cyber-risk cost matrix (Kim et al., 2012). In Appendix E, we provide an illustration of how applying a cost-sensitive classifier and a MetaCost wrapper to our proposed DTL-EL model leads to a trade-off between total cost, average misclassification cost, F1-score, and mislabeled exploits.

Cyber analysts: Common tasks that many security analysts in cybersecurity operations centers (CSOCs) often conduct include monitoring, identifying, and ranking threats to their cyber infrastructure (Samtani et al., 2020). However, there are significant difficulties in collecting and sifting through large cyber threat data sources (Agyepong et al., 2020). To conduct common tasks, cyber analysts require a framework that can facilitate red teaming (Alomar et al., 2020). CSOC analysts can benefit from the automatic and incremental collection and labeling features provided by the DTL-EL framework through custom alerts for new exploits. The new exploits and their visualized semantic relations could help CSOC analysts gain insight into protections against the code via operationalized penetration testing, professional domain knowledge, or consultation.

Cybersecurity educators: Training cyber analysts with data mining skills is an essential component of the National Institute of Standards and Technology (NIST) Initiative for Cybersecurity Education (Shoemaker et al., 2018). However, this often means that educators typically require up-to-date and labeled datasets to keep their curriculums current. The large, international hacker community datasets in this study can be leveraged in emerging cybersecurity analytics curricula. For example, the labeled exploit datasets from our three collected sources and auto-generated summary statistics labeled testbed can be incorporated into massively open online courses (MOOCs) for dissemination to domestic and international institutions. Since the collection is updated weekly, cybersecurity educators can provide up-to-date content to their students.

Contributions to the IS Knowledge Base

IS scholars have stressed the importance of contributing prescriptive knowledge to the IS knowledge base with a novel IT artifact (Rai, 2017). Our proposed framework is situated within the growing body of IS cybersecurity analytics research. This stream of literature has primarily relied on a single dataset type and rarely included an explanation (attention visualization) for end users to interact with (limiting the potential practical utility of the analytics). Considering these issues, this work aims to contribute a novel multimodal cybersecurity analytics approach to the IS knowledge base. The large international hacker community testbeds and DTL-EL can help future IS scholars and cybersecurity stakeholders pursue advanced cybersecurity analytics research on exploit labeling.

Our study also follows the guidelines of Type I ML research and contributes a BiLSTM model IT artifact with a carefully combined feature-based model pretraining, expert-knowledge layer transfer, and long sequential text classification to the IS knowledge base (Padmanabhan et al., 2022). While our IT artifact has been built for exploit labeling (rich textual source domain, noisy textual target domain), DTL-EL could be adapted for other classification tasks of interest to the IS discipline. If an IS researcher has a source domain with two textual features, they can pre-initialize with short textual features that are fully populated in the dataset and have some explanatory value to warm up the model. The source domain model should then be trained with long-sequential textual features. The layers of the source domain model should then be highly transferrable to a similar but noisier domain. This transferability can be tested via an ablation analysis and a CKA similarity measurement to identify the specific model components that DTL improves. Additionally, transfer learning can be conducted at the embedding layer. The model training results indicated that transferring pretrained embeddings (e.g., GloVe) consistently improved performance. While the GloVe embedding outperformed other options for our task, contextual embeddings (e.g., BERT) could perform better than context-free embeddings for a DTL task than with a semantically consistent textual dataset (e.g., product descriptions). We provided two examples of domains of interest to the IS community that can benefit from DTL-EL to further demonstrate its applicability and generalizability.

Healthcare informatics: Recent IS literature has focused on improving the quality of electronic health records (EHRs) (Kohli & Tan, 2016). EHRs are unstructured text documents detailing a patient's medical history. DTL-EL could be adapted to perform EHR classification tasks (e.g., medical diagnosis) by constructing a ground-truth dataset of EHRs and related diagnoses, pre-initializing the high-level summaries, and training the model with the full EHR (containing long-term dependencies of the patient's medical history). This model could be leveraged via transfer to a target domain without high-level summaries or detailed EHRs.

Social media analytics: Online social networking platforms contain a wealth of text that can be used for important tasks such as product review analysis, conversation disentanglement, and more (Chen et al., 2012). However, discussions on social media often use long sentences with uncommon phrases and semantic issues. Scholars can consider adapting DTL-EL to process long social media texts. For example, annotated Twitter datasets can be used as a source domain where the model is pre-initialized with the annotations and trained with the tweet content. Additionally, embeddings trained on Twitter data (e.g., BERTweet) can be applied to the model to boost performance. Then, a target domain model can be trained on a similar social network (e.g., Reddit) with transferred features from the Twitter dataset to improve model performance.

Conclusion And Future Directions

The rapid proliferation of complex IS systems has been met by new exploits designed to circumvent vulnerabilities and cause irreparable cyber breaches. Recently, practitioners and academics have placed significant focus on proactively identifying and labeling exploits from hacker forums to mitigate these cyber threats. However, prevailing approaches for labeling hacker exploits do not leverage knowledge from exploit DNMs or public exploit repositories to enhance hacker exploit labeling performance. Consequently, executing critical CTI tasks that rely on labels remains a significant challenge.

In this study, we adopted the computational design science paradigm to develop a novel deep transfer learning exploit labeler (DTL-EL) framework for labeling exploits from hacker forums. DTL-EL incorporates a novel approach for pre-initializing the BiLSTM with a self-attention mechanism in the source domain based on the rich metadata (e.g., exploit titles) found in exploit DNMs and public exploit repositories. We demonstrated through a series of benchmark experiments that DTL-EL outperformed state-of-the-art non-DTL ML and DL techniques in labeling hacker exploit source code in hacker forums. The results indicated that the pre-initialized BiLSTM with an attention mechanism better identified and weighted key features than their non-pre-initialized counterparts. DTL-EL offers proactive CTI capabilities at the tactical, operational, and strategic levels to help companies improve their security posture against cyberattacks.

We identified three promising directions for future work. First, DTL-EL can be adapted and extended for cybersecurity tasks such as identifying, collecting, and labeling personally identifiable information, malicious pastes, and DNM postings. Second, linking our labeled exploits to prevailing cybersecurity risk management frameworks (e.g., MITRE ATT&CK) can allow for a more fine-grained analysis of exploit types, risk assessments, and targeted mitigation strategies. These new insights can provide information about exploit types outside this project's scope (e.g., multi-label exploits). Third, social network analysis and named entity resolution can be performed to link hackers, forums, and exploits with specific target users or organizations. Each direction can significantly improve proactive CTI collection and dissemination efforts and ultimately contribute to a safer cyberspace for organizations, individuals, and governments.

Acknowledgments

We are grateful to the senior editor, the associate editor, and the three anonymous reviewers for their constructive comments and feedback. This material is based on work supported by the National Science Foundation under Grant DUE-1303362, OAC-1917117, DGE-1946537, and CNS-1850362.

References

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, *6*, 52138-52160. https://doi.org/10.1109/ACCESS.2018.2870052

Agyepong, E., Cherdantseva, Y., Reinecke, P., & Burnap, P. (2020). Challenges and performance metrics for security operations center analysts: A systematic review. *Journal of Cyber Security Technology*, *4*(3), 125-152. https://doi.org/10.1080/23742917.2019.1698178

Alomar, N., Wijesekera, P., Qiu, E., & Egelman, S. (2020). "You've got your nice list of bugs, now what?" Vulnerability discovery and management processes in the wild. In *Proceedings of the 16th Symposium on Usable Privacy and Security* (pp. 319-339). https://www.usenix.org/conference/soups2020/presentation/alomar

Ampel, B. M., Samtani, S., Zhu, H., Ullman, S., & Chen, H. (2020). Labeling hacker exploits for proactive cyber threat intelligence: A deep transfer learning approach. In *Proceedings of the IEEE Conference on Intelligence and Security Informatics*. https://doi.org/10.1109/ISI49825.2020.9280548

Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate.* arXiv. http://arxiv.org/abs/1409.0473

Benaroch, M. (2018). Real options models for proactive uncertainty-reducing mitigations and applications in cybersecurity investment decision making. *Information Systems Research*, *29*(2), 315-340. https://doi.org/10.1287/isre.2017.0714

Benjamin, V., Valacich, J. S., & Chen, H. (2019). DICE-E: A Framework for conducting darknet identification, collection, evaluation with ethics. *MIS Quarterly*, *43*(1), 1-22. https://doi.org/10.25300/MISQ/2019/13808

Brown, R., & Lee, R. M. (2021). *2021 SANS Cyber Threat Intelligence (CTI) Survey*. SANS Institute. https://www.sans.org/white-papers/40080/

Chen, G., Chen, P., Shi, Y., Hsieh, C.-Y., Liao, B., & Zhang, S. (2019). *Rethinking the usage of batch normalization and dropout in the training of deep neural networks.* Arxiv. http://arxiv.org/abs/1905.05928

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, *36*(4), 1165-1188. https://doi.org/10.2307/41703503

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). https://doi.org/10.1145/2939672.2939785

Chollet, F. (2015). *Keras.* GitHub. https://github.com/keras-team/keras

Deliu, I., Leichter, C., & Franke, K. (2018). Collecting cyber threat intelligence from hacker forums via a two-stage, hybrid process using support vector machines and latent Dirichlet allocation. In *Proceedings of the IEEE International Conference on Big Data* (pp. 5008-5013). https://doi.org/10.1109/BigData.2018.8622469

Ebrahimi, M., Chai, Y., Samtani, S., & Chen, H. (2022). Cross-lingual cybersecurity analytics in the international dark web with adversarial deep representation learning. *MIS Quarterly*, *46*(2), 1209-1226. https://doi.org/10.25300/MISQ/2022/16618

Ebrahimi, M., Nunamaker, J. F., & Chen, H. (2020). Semi-supervised cyber threat identification in dark net markets: A transductive and deep learning approach. *Journal of Management Information Systems*, *37*(3), 694-722. https://doi.org/10.1080/07421222.2020.1790186

Ebrahimi, M., Surdeanu, M., Samtani, S., & Chen, H. (2018). Detecting cyber threats in non-English dark net markets: A cross-lingual transfer learning approach. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics* (pp. 85-90). https://doi.org/10.1109/ISI.2018.8587404

Elsheikh, A., Yacout, S., & Ouali, M.-S. (2019). Bidirectional handshaking LSTM for remaining useful life prediction. *Neurocomputing*, *323*, 148-156. https://doi.org/10.1016/j.neucom.2018.09.076

Goyal, P., Hossain, K. T., Deb, A., Tavabi, N., Bartley, N., Abeliuk, A., Ferrara, E., & Lerman, K. (2018). *Discovering signals from web sources to predict cyber attacks.* arXiv. http://arxiv.org/abs/1806.03342

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75-105. https://doi.org/10.2307/25148625

Houlsby, N., Giurgiu, A., Jastrzçbski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 4944-4953).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning* (pp. 448-456). https://doi.org/10.5555/3045118.3045167

Jain, S., & Wallace, B. C. (2019). Attention is not explanation. In *Proceedings of the 2019 Conference of the North* (pp. 3543-3556). https://doi.org/10.18653/v1/N19-1357

Kapllani, G., Khomyakov, I., Mirgalimova, R., & Sillitti, A. (2020). An empirical analysis of the maintainability evolution of open source systems. *Open Source Systems*, 78-86. https://doi.org/10.1007/978-3-030-47240-5_8

Karhu, K., Gustafsson, R., & Lyytinen, K. (2018). Exploiting and defending open digital platforms with boundary resources: Android's five platform forks. *Information Systems Research*, *29*(2), 479-497. https://doi.org/10.1287/isre.2018.0786

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 3149-3157). https://dl.acm.org/doi/10.5555/3294996.3295074

Kim, J., Choi, K., Kim, G., & Suh, Y. (2012). Classification cost: An empirical comparison among traditional classifier, cost-sensitive classifier, and MetaCost. *Expert Systems with Applications*, *39*(4), 4013-4019. https://doi.org/10.1016/j.eswa.2011.09.071

Kohli, R., & Tan, S. S.-L. (2016). Electronic health records: How can IS researchers contribute to transforming healthcare? *MIS Quarterly*, *40*(3), 553-574. https://www.jstor.org/stable/26629027

Kornblith, S., Norouzi, M., Lee, H., & Hinton, G. (2019). Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 3519-3529).

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information. An International Interdisciplinary Journal*, *10*(4), 150. https://doi.org/10.3390/info10040150

Lacchia, M. (2020). *Radon 4.1.0 documentation*. https://radon.readthedocs.io/en/latest/

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., ... & He, L. (2022). A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology*, *13*(2), 1-41. https://doi.org/10.1145/3495162

Li, X., Grandvalet, Y., & Davoine, F. (2018). Explicit inductive bias for transfer learning with convolutional networks. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 2825-2834).

Lin, Y.-K., Chen, H., Brown, R. A., Li, S.-H., & Yang, H.-J. (2017). Healthcare predictive analytics for risk profiling in chronic care: A Bayesian multitask learning approach. *MIS Quarterly*, *41*(2), 473-495. https://doi.org/10.25300/MISQ/2017/41.2.07

Liu, G., & Guo, J. (2019). Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing*, *337*, 325-338. https://doi.org/10.1016/j.neucom.2019.01.078

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., & Smith, N. A. (2019). Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 1073-1094). https://doi.org/10.18653/v1/N19-1112

Liu, P., Qiu, X., & Huang, X. (2017). Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P17-1001

Liu, S. (2020). A unified framework to learn program semantics with graph neural networks. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (pp. 1364-1366). https://doi.org/10.1145/3324884.3418924

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1412-1421).

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., & Jin, Z. (2016). How transferable are neural networks in NLP applications? *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 479-489). https://doi.org/10.18653/v1/D16-1046

Newman, L. (2020). Russia's FireEye hack is a statement—but not a catastrophe. *Wired*. https://www.wired.com/story/russia-fireeye-hack-statement-not-catastrophe/

Nuñez-Varela, A. S., Pérez-Gonzalez, H. G., Martínez-Perez, F. E., & Soubervielle-Montalvo, C. (2017). Source code metrics: A systematic mapping study. *The Journal of Systems and Software*, *128*, 164-197. https://doi.org/10.1016/j.jss.2017.03.044

Padmanabhan, B., Fang, X., Sahoo, N., & Burton-Jones, A. (2022). Editor's comments: Machine learning in information systems research. *MIS Quarterly*, *46*(1), iii-xix.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345-1359. https://doi.org/10.1109/TKDE.2009.191

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825-2830. http://jmlr.org/papers/v12/pedregosa11a.html

Peng, N., & Dredze, M. (2017). Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP* (pp. 91-100). https://doi.org/10.18653/v1/W17-2612

Peng, T., Zuo, W., & He, F. (2008). SVM based adaptive learning method for text classification from positive and unlabeled documents. *Knowledge and Information Systems*, *16*(3), 281-301. https://doi.org/10.1007/s10115-007-0107-1

Rai, A. (2017). Editor's comments: Diversity of design science research. *MIS Quarterly*, *41*(1), iii-xviii.

Ruder, S., Peters, M. E., Swayamdipta, S., & Wolf, T. (2019). Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/N19-5004

Samtani, S., Chai, Y., & Chen, H. (2022). 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), 911-946. https://doi.org/10.25300/MISQ/2022/15392

Samtani, S, Abate, M., Benjamin, V., & Li, W. (2020). *Cybersecurity as an industry: A cyber threat intelligence perspective*. Springer. https://doi.org/10.1007/978-3-319-90307-1

Samtani, S., Chinn, R., Chen, H., & Nunamaker, J. F. (2017). Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence. *Journal of Management Information Systems*, *34*(4), 1023-1053. https://doi.org/10.1080/07421222.2017.1394049

Sen, R., Verma, A., & Heim, G. R. (2020). Impact of cyberattacks by malicious hackers on the competition in software markets. *Journal of Management Information Systems*, *37*(1), 191-216. https://doi.org/10.1080/07421222.2019.1705511

Shen, T., Zhou, T., Long, G., Jiang, J., & Zhang, C. (2018). Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *Proceedings of the 6th International Conference on Learning Representations.*

Shin, B., & Lowry, P. B. (2020). A review and theoretical explanation of the "cyberthreat-intelligence (CTI) capability" that needs to be fostered in information security practitioners and how this can be accomplished. *Computers & Security*, *92*, Article 101761. https://www.sciencedirect.com/science/article/pii/S0167404820300456

Shoemaker, D., Kohnke, A., & Sigler, K. (2018). *A guide to the National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework (2.0)*. Auerbach Publications.

Silic, M., & Lowry, P. B. (2020). Using design-science based gamification to improve organizational security training and compliance. *Journal of Management Information Systems*, *37*(1), 129-161. https://doi.org/10.1080/07421222.2019.1705512

Son, L. H., Kumar, A., Sangwan, S. R., Arora, A., Nayyar, A., & Abdel-Basset, M. (2019). Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. *IEEE Access*, *7*, 23319-23328. https://doi.org/10.1109/ACCESS.2019.2899260

Srivastava, A., Makhija, P., & Gupta, A. (2020). Noisy text data: Achilles' heel of BERT. In *Proceedings of the Sixth Workshop on Noisy User-Generated Text* (pp. 16-21). https://doi.org/10.18653/v1/2020.wnut-1.3

Sun, H., Xu, M., & Zhao, P. (2020). Modeling malicious hacking data breach risks. *North American Actuarial Journal*, *25*(4), 484-502. https://doi.org/10.1080/10920277.2020.1752255

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A Survey on deep transfer learning. In *Proceedings of the International Conference on Artificial Neural Networks* (pp. 270-279). https://doi.org/10.1007/978-3-030-01424-7_27

Thangaraj, M., & Sivakami, M. (2018). Text classification techniques: A literature review. *Interdisciplinary Journal of Information, Knowledge, and Management*, *13*, 117-135. https://doi.org/10.28945/4066

Tounsi, W., & Rais, H. (2018). A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, *72*, 212-233. https://doi.org/10.1016/j.cose.2017.09.001

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems* (pp. 6000-6010). https://dl.acm.org/doi/10.5555/3295222.3295349

Wagner, T. D., Mahbub, K., Palomar, E., & Abdallah, A. E. (2019). Cyber threat intelligence sharing: Survey and research directions. *Computers & Security*, *87*(11), 1-13. https://doi.org/10.1016/j.cose.2019.101589

Wiegreffe, S., & Pinter, Y. (2019). Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (pp. 11-20). https://doi.org/10.18653/v1/D19-1002

Williams, R., Samtani, S., Patton, M., & Chen, H. (2018). Incremental hacker forum exploit collection and classification for proactive cyber threat intelligence: An exploratory study. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics* (pp. 94-99). https://doi.org/10.1109/ISI.2018.8587336

Yenter, A., & Verma, A. (2017). Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In *Proceedings of the IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference* (pp. 540-546). https://doi.org/10.1109/UEMCON.2017.8249013

Yin, H. H. S., Langenheldt, K., Harlev, M., Mukkamala, R. R., & Vatrapu, R. (2019). Regulating cryptocurrencies: A supervised machine learning approach to de-anonymizing the Bitcoin blockchain. *Journal of Management Information Systems*, *36*(1), 37-73. https://doi.org/10.1080/07421222.2018.1550550

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems* (vol. 2, pp. 3320-3328) https://dl.acm.org/doi/10.5555/2969033.2969197

Yue, W. T., Wang, Q.-H., & Hui, K.-L. (2019). See no evil, hear no evil? Dissecting the impact of online hacker forums. *MIS Quarterly*, *43*(1), 73-95. https://doi.org/10.25300/MISQ/2019/13042

Zhu, H., Samtani, S., Chen, H., & Nunamaker, J. F. (2020). Human identification for activities of daily living: A deep transfer learning approach. *Journal of Management Information Systems*, *37*(2), 457-483. https://doi.org/10.1080/07421222.2020.1759961

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1) 43-76. https://doi.org/10.1109/JPROC.2020.3004555

About the Authors

Benjamin M. Ampel is a Ph.D. student in the Department of Management Information Systems at the Eller College of Management, University of Arizona. Benjamin serves as a CyberCorps Scholarship-for-Service Fellow in the UArizona Artificial Intelligence Laboratory under the guidance of Dr. Hsinchun Chen. His research primarily focuses on AI-enabled Cybersecurity. Benjamin has published peer reviewed articles that have appeared in journals such as *AIS Transactions on Replication Research* and *ACM Digital Threats: Research and Practice* and in conference proceedings such as *IEEE ISI*, *AMCIS*, and *ICIS*. He has also contributed to a variety of projects supported by the National Science Foundation (NSF) relating to secure and trustworthy computing (SaTC) and cybersecurity innovation for cyber infrastructure (CICI).

Sagar Samtani is an assistant professor and Grant Thornton Scholar in the Department of Operations and Decision Technologies and the Founding Director of the Data Science and Artificial Intelligence Lab at the Kelley School of Business at Indiana University (IU). He received his Ph.D. from the Artificial Intelligence (AI) Lab at the University of Arizona. Dr. Samtani's research focuses on developing AI-enabled algorithms and systems for cybersecurity and mental health applications. He has published over 65 journal, conference, and workshop papers in *MIS Quarterly*, *Information Systems Research*, *Journal of MIS*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Dependable and Secure Computing*, and others. His research has received funding from the NSF and other agencies. He has won several Best Paper awards for his research. Dr. Samtani has won the IU Outstanding Junior Faculty Award, the IEEE Big Data Security Junior Research Award, the AIS Early Career Award, and the IU Trustees Teaching Award. He was inducted into the NSF/CISA CyberCorps SFS Hall of Fame and was named by Poets and Quants as a Top 50 Undergraduate Business School Professor in 2022. Dr. Samtani's work has received media attention from the Associated Press, *WIRED*, *Forbes*, *Miami Herald*, Fox, *Science Magazine*, and AAAS.

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 (UTSA). He received his Ph.D. in management information systems from the University of Arizona (UA). Dr. Zhu's research focuses on developing advanced analytics (e.g., deep learning) 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. He is a member of the IEEE, ACM, AIS, and INFORMS.

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 Ph.D. in information systems from New York University. He is the author/editor of over 20 books, 25 book chapters, 320 SCI journal articles, and 160 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 in 1989, which has received significant research funding ($60M+) from the NSF, NIH, DOD, DOJ, CIA, DHS, and other agencies. He has served as editor-in-chief, senior editor, or associate editor for major ACM/IEEE (*ACM TMIS*, *ACM TOIS*, *IEEE IS*, *IEEE SMC*), MIS (*MISQ*, *DSS*), and Springer (*JASIST*) journals and has served as conference/program chair of major ACM/IEEE/MIS conferences in digital libraries (ACM/IEEE JCDL, ICADL), information systems (ICIS), security informatics (IEEE ISI), and health informatics (ICSH). Dr. Chen is the director of the UA AZSecure Cybersecurity Program, with $10M+ funding from NSF SFS, SaTC, and CICI programs and CAE-CD/CAE-R cybersecurity designations from NSA/DHS. He is a fellow of ACM, IEEE, and AAAS.

Appendix A

Benchmark Model Specifications

The naive Bayes, logistic regression, decision tree, and support vector machine (SVM) models were implemented using the Scikit-Learn library (Pedregosa et al., 2011). The XGBoost model was implemented using the XGBoost Python library (Chen & Guestrin, 2016). The LightGBM was implemented using the LightGBM Python library (Ke et al., 2017). Our input was transformed using a term frequency-inverse document frequency (TF-IDF) calculation that was applied to turn the co-occurrence counts into vector representations. The output of the TF-IDF transformation was then inputted into each classification model. The GridSearchCV module in Scikit-Learn was used for each model to search for the best parameters for our exploit code labeling task. We summarize the parameters determined by grid-search for each of the models below:

- **Naive Bayes:** Additive smoothing parameter (alpha in Scikit-Learn) of 1.
- **Logistic Regression:** L2 penalty term, the liblinear solver, and a C parameter of 0.01.
- **Decision Tree:** had a max depth of 5.
- **SVM:** Linear support vector classification (linearSVC in Scikit-Learn).
- **XGBoost:** Learning rate of 0.01, max tree depth of 10, minimum child weight of 6, multi-softmax objective function, sub-sample of 0.8.
- **LightGBM:** Learning rate of 0.01, no max depth, minimum child weight of 0, 1,000 estimators, sub-sample of 0.85.

Each deep learning model was implemented with the Keras Python library (Chollet, 2015). Model parameters were adjusted based on best practices in related literature (Li et al., 2020). When we evaluated the changes in performance by swapping out specific layers, such as gated recurrent unit (GRU), long-short term memory (LSTM), or bidirectional LSTM (BiLSTM), we kept the embedding, convolutional, batch normalization, and dropout layers constant. The convolutional layer was one-dimensional, had a kernel size of 3, and a rectified linear unit (ReLU) activation function. Each dropout layer was set to 0.5. All subsequent benchmark models followed the same structure (unless there was an attention layer or pre-initialization design). The self-attention mechanism uses the Keras L2 regularizer set to $1e-4$ for the kernel, and the L1 regularizer set to $1e-4$ for the bias term. To implement our pre-initialization design, the BiLSTM layer in the exploit title model had the return state set to true with no concatenation. We manually concatenated the returned hidden and cell states. The concatenated output was fed into the exploit code model at the BiLSTM layer by setting the initial state equal to the concatenated output. The remainder of the model was the same as the BiLSTM.

Appendix B

Per Exploit Label Analysis for Experiments 1 and 2

In the main text, we presented the results of the proposed DTL-EL and all benchmark methods across all eight exploit labels in Experiment 1 and Experiment 2. However, we were also interested in identifying how each approach performed at the exploit category level for each domain. In Experiment 1, our proposed DTL-EL did not always produce the highest precision or recall score for each exploit label, but always produced the highest F1-score. DTL-EL's precision was lower than the pre-initialized BiLSTM without attention on cross-site scripting (XSS; 93.31% to 96.14%) and web applications (94.71% to 95.88%). Each exploit's properties can partially explain these results. XSS attacks and web application exploits are targeted (e.g., written for a specific website) and contain named entities that do not frequently appear in other exploits and may adversely affect our self-attention mechanism. DTL-EL attained a lower F1-score than the pre-initialized BiLSTM for SQL injections (88.98% to 90.04%), remote exploits (83.55% to 80.91%), and file inclusion exploits (88.45% to 86.76%), which are three of the four most infrequent exploits in the source domain (possibly causing lower recall scores). Since Experiment 2 examined DTL-EL's labeling performance (our core objective), we provide the precision, recall, and F1-score for each exploit target label in Table B1. Our proposed DTL-EL outperformed all models in each exploit label in terms of recall and F1-score in the target domain. DTL-EL performed best in precision for five exploit labels but underperformed LSTM in SQL injection (91.27% to 91.45%) and XSS exploits (75.67% to 77.11%). DTL-EL's precision underperformed LightGBM for file inclusion exploits (50.77% to 58.68%). In the target domain, the code for XSS, file inclusion, and SQL injection are often much shorter than exploits in the other five labels, possibly explaining DTL-EL's precision scores.

Table B1. Per Label Results for Experiment 2: DTL-EL against Non-Transfer Learning Approaches on the Target Domain

Exploit label	Model type	Model	Precision	Recall	F1-Score
Web applications	**Classical machine learning**	Naive Bayes	76.92%***	08.93%***	16.00%***
		Logistic regression	65.32%***	08.04%***	14.75%***
		Decision tree	39.18%***	33.93%***	36.36%***
		SVM	54.71%***	19.64%***	30.14%***
		XGBoost	52.83%***	25.00%***	33.94%***
		LightGBM	55.56%***	26.79%***	36.14%***
	Deep learning	RNN	56.48%***	25.11%***	35.87%***
		GRU	59.39%***	31.61%***	37.93%***
		LSTM	59.80%***	40.18%*	44.22%***
		BiLSTM	60.48%***	40.46%*	45.47%***
		BiLSTM with attention	62.31%***	41.14%	47.67%***
	Proposed DTL-EL		**65.97%**	**41.43%**	**49.17%**
Denial of service (DoS)	**Classical machine learning**	Naive Bayes	64.81%***	43.54%***	57.54%***
		Logistic regression	67.14%***	50.32%***	60.91%***
		Decision tree	67.16%***	56.91%***	61.61%***
		SVM	64.46%***	53.19%***	62.05%***
		XGBoost	67.89%***	56.27%***	65.33%***
		LightGBM	75.35%***	57.80%***	65.42%***
	Deep learning	RNN	65.12%***	52.18%***	59.31%***
		GRU	70.03%***	59.87%***	64.55%***
		LSTM	72.48%***	70.59%***	71.52%***
		BiLSTM	74.75%***	67.46%***	70.92%***
		BiLSTM with attention	75.15%***	69.42%***	72.67%***
	Proposed DTL-EL		**78.48%**	**72.42%**	**75.57%**
Remote	**Classical machine learning**	Naive Bayes	64.29%***	09.00%***	15.79%***
		Logistic regression	68.60%***	23.60%***	35.12%***
		Decision tree	49.88%***	42.00%***	45.60%***
		SVM	62.55%***	30.40%***	40.92%***
		XGBoost	67.14%***	37.60%***	48.21%***
		LightGBM	62.17%***	37.80%***	47.01%***
	Deep learning	RNN	61.08%***	39.15%***	50.11%***
		GRU	61.01%***	41.20%***	51.66%***
		LSTM	63.74%***	45.99%***	53.16%***
		BiLSTM	67.13%***	48.41%***	56.21%***
		BiLSTM with attention	68.02%***	48.88%***	58.03%***
	Proposed DTL-EL		**75.88%**	**51.20%**	**63.41%**

Local	**Classical machine learning**	Naive Bayes	59.92%***	37.61%***	46.21%***
		Logistic regression	58.77%***	51.16%***	54.70%***
		Decision tree	57.50%***	53.85%***	55.61%***
		SVM	61.72%***	53.36%***	57.24%***
		XGBoost	65.40%***	56.78%***	60.78%***
		LightGBM	64.43%***	58.61%***	61.38%***
	Deep learning	RNN	65.87%***	57.12%***	61.62%***
		GRU	68.14%***	62.97%***	65.35%***
		LSTM	70.63%***	65.20%***	67.81%***
		BiLSTM	70.08%***	66.91%***	68.46%***
		BiLSTM with attention	71.13%***	67.22%***	69.71%***
	Proposed DTL-EL		**75.73%**	**71.43%**	**73.68%**
SQL Injection	**Classical machine learning**	Naive Bayes	77.01%***	82.45%***	79.64%***
		Logistic regression	83.03%***	81.47%***	82.24%***
		Decision tree	80.38%***	80.20%***	80.29%***
		SVM	83.22%***	81.10%***	82.14%***
		XGBoost	84.58%***	85.15%***	84.86%***
		LightGBM	83.60%***	84.92%***	84.26%***
	Deep learning	RNN	83.29%***	83.75%***	83.54%***
		GRU	81.47%***	85.45%***	83.41%***
		LSTM	**91.45%**	77.79%***	84.07%***
		BiLSTM	87.81%***	84.85%***	86.30%***
		BiLSTM with attention	88.14%***	85.06%***	86.61%***
	Proposed DTL-EL		91.27%	**92.12%**	**91.84%**
Cross-Site Scripting (XSS)	**Classical machine learning**	Naive Bayes	60.50%***	54.05%***	57.09%***
		Logistic regression	60.46%***	62.96%***	61.68%***
		Decision tree	58.65%***	58.19%***	58.42%***
		SVM	60.70%***	65.82%***	63.16%***
		XGBoost	68.39%***	65.34%***	66.83%***
		LightGBM	68.07%***	66.77%***	67.42%***
	Deep learning	RNN	69.78%***	67.08%***	68.29%***
		GRU	69.08%***	66.49%***	67.83%***
		LSTM	**77.11%**	65.72%***	70.96%***
		BiLSTM	74.40%*	71.02%***	72.67%***
		BiLSTM with attention	75.06%*	71.69%***	73.36%***
	Proposed DTL-EL		75.67%	**74.99%**	**75.43%**
File Inclusion	**Classical machine learning**	Naive Bayes	55.71%***	25.00%***	38.71%***
		Logistic regression	33.56%***	12.50%***	22.22%***
		Decision tree	44.44%***	33.33%***	38.10%***
		SVM	57.78%	29.17%***	42.42%***
		XGBoost	57.02%	29.83%***	41.81%***
		LightGBM	**58.68%**	33.15%***	47.06%***
	Deep learning	RNN	43.32%***	35.57%***	39.88%***
		GRU	45.10%***	36.12%***	41.87%***
		LSTM	47.50%***	39.03%***	42.73%***
		BiLSTM	43.33%***	39.17%***	41.11%***
		BiLSTM with attention	45.46%***	41.08%***	43.24%***
	Proposed DTL-EL		50.77%	**46.67%**	**48.72%**
Overflow	**Classical machine learning**	Naive Bayes	69.15%***	25.30%***	38.35%***
		Logistic regression	65.00%***	49.24%***	56.03%***
		Decision tree	61.07%***	53.48%***	57.03%***
		SVM	63.19%***	54.09%***	58.29%***
		XGBoost	69.38%***	59.39%***	64.00%***
		LightGBM	69.23%***	61.36%***	65.06%***
	Deep learning	RNN	71.45%***	61.92%***	66.21%***
		GRU	71.62%***	61.85%***	66.30%***
		LSTM	69.02%***	63.79%***	66.30%***
		BiLSTM	72.76%***	68.79%***	70.78%***
		BiLSTM with attention	73.04%***	69.46%***	71.87%***
	Proposed DTL-EL		**75.76%**	**72.94%**	**74.52%**

Note: $*\ p < 0.05$, $**\ p < 0.01$, $***\ p < 0.001$. Top scores appear in boldface.

Appendix C

Model Similarity Analysis

As indicated in the main text, we implemented centered kernel alignment (CKA) to visualize the internal representations of our target domain model (BiLSTM with self-attention) and DTL-EL to identify if the transfer learning process transfers layers that improve exploit labeling. CKA outputs a heatmap of similarities (from 0-1, 1 being highest) between layers (Kornblith et al., 2019). If the internal representations of the two models are similar, it may demonstrate that transfer learning is not providing significant model benefits, i.e., not transferring layers that improve performance. We present the CKA analysis results in Figure C1.

The results of our CKA analysis show that the lower-level layers of the DTL-EL model (0: embedding, 1: convolution, 2: pooling, 3: batch norm, 4: dropout) are more similar to higher-level layers (5: BiLSTM, 6: self-attention, 7: dropout, 8: dense) than the non-DTL model. Additionally, the higher-level layers are more similar to each other in the DTL model compared to the non-DTL model. Conversely, the DTL lower-level layers are less similar than the lower-level layers in the non-DTL model. These internal differences suggest that the transferred layers are learning lower-level feature representations of the exploits, potentially explaining the difference in performance over the randomly initialized weights of the target domain model. The differences also may suggest that the model is learning longer-range dependencies (BiLSTM and attention layers) at the cost of local-range dependencies (lower-level layers).

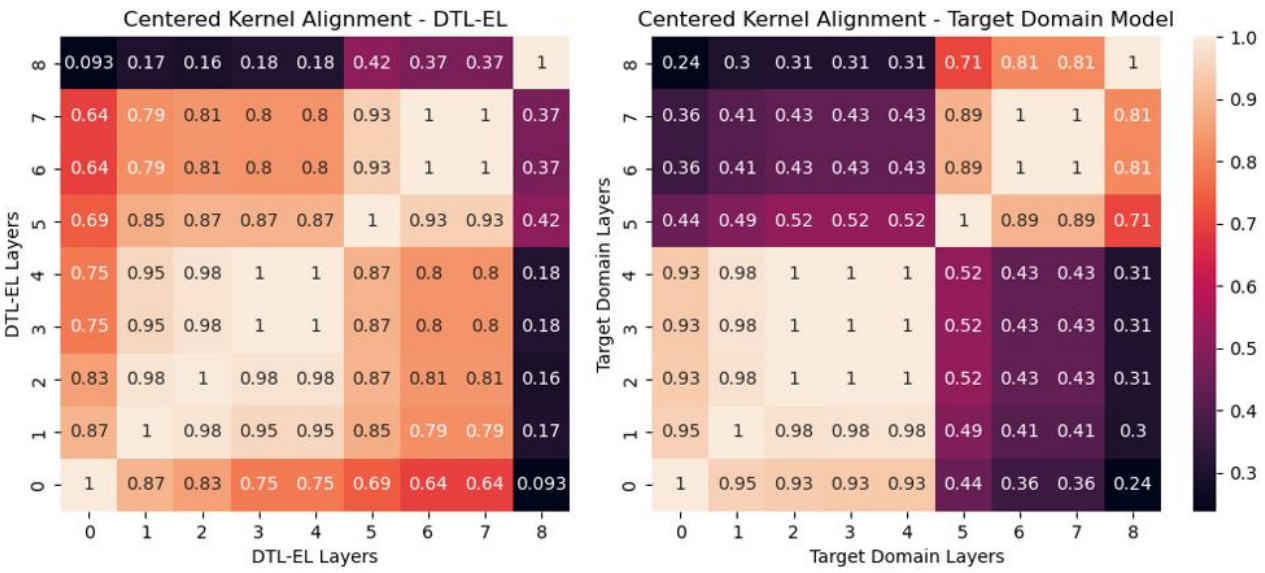

Figure C1. Results of DTL-EL's CKA (left) and Target Domain Model's CKA (right)

Appendix D

Implementation of DTL-EL Into a Web-Based User Interface

We applied DTL-EL to the unlabeled source code from our hacker forum testbed (74,605 source code snippets). DTL-EL only applied a label to the source code if the softmax function probability was greater than 80% for a given label[4]. Overall, the DTL-EL labeling process yielded 27,143 exploits from eight international hacker forums from 2002 to 2020. DoS (6,726 exploits), SQL injection (6,685 exploits), and local (4,098 exploits) were the most common exploits among all forums, while file inclusion (118 exploits) was the least common. Past IS literature has indicated that a system with a user interface (UI) can help CTI stakeholders effectively interact with a novel algorithm and its results (Samtani et al. 2017). However, most past IS cybersecurity analytics studies have not incorporated their proposed IT artifact into a system or UI. Our collection of labeled hacker forum exploits was incorporated into a UI to highlight the value of our artifact for organizational use. Our UI aims to facilitate the needs of cybersecurity stakeholders based on a targeted analysis of relevant literature. Figure D1 illustrates how our proposed DTL-EL framework is integrated into a UI for organizational use.

The UI offers several functions that help address relevant cybersecurity stakeholders' requirements when interacting with systems with advanced cybersecurity analytics. First, cyber analysts using the UI can explore our content, input a single exploit, or upload a comma separated value (CSV) file or JavaScript object notation (JSON) object with multiple exploits to return a list of exploit labels (top left of Figure D1) using the proposed DTL-EL. Second, cybersecurity managers can explore automated trends in exploits from our crawlers or their CSV at a total or per-label level (bottom left of Figure D1). The dates and exploit labels can be adjusted to generate custom visualizations for targeted analysis. For example, we see in Figure D1 that local exploits saw a steep increase in posted source code from October 2019 to February 2020. Third, cyber analysts and cybersecurity educators can closely look at the labeled exploits once a particular trend has been identified for more in-depth analysis (top middle of Figure D1). These tables can be filtered based on year, hacker name, exploit type, or forum to facilitate strategic CTI. Additionally, cyber analysts and educators can download filtered content of interest. Finally, cyber analysts can click on any exploit to see visualized semantic dependencies of the exploit. These dependencies show how the source code operates from a token importance standpoint. After identifying recent local exploits, a cyber analyst could compare the semantic dependencies between many local exploits and further determine specific trends within the coding practices of each exploit for tactical CTI. In the example, a cyber analyst using the UI could identify the increase in local attacks and investigate the most recent source code further. This will illuminate characteristics of how the code operates that could allow an analyst to operationalize the exploits and create countermeasures against them.

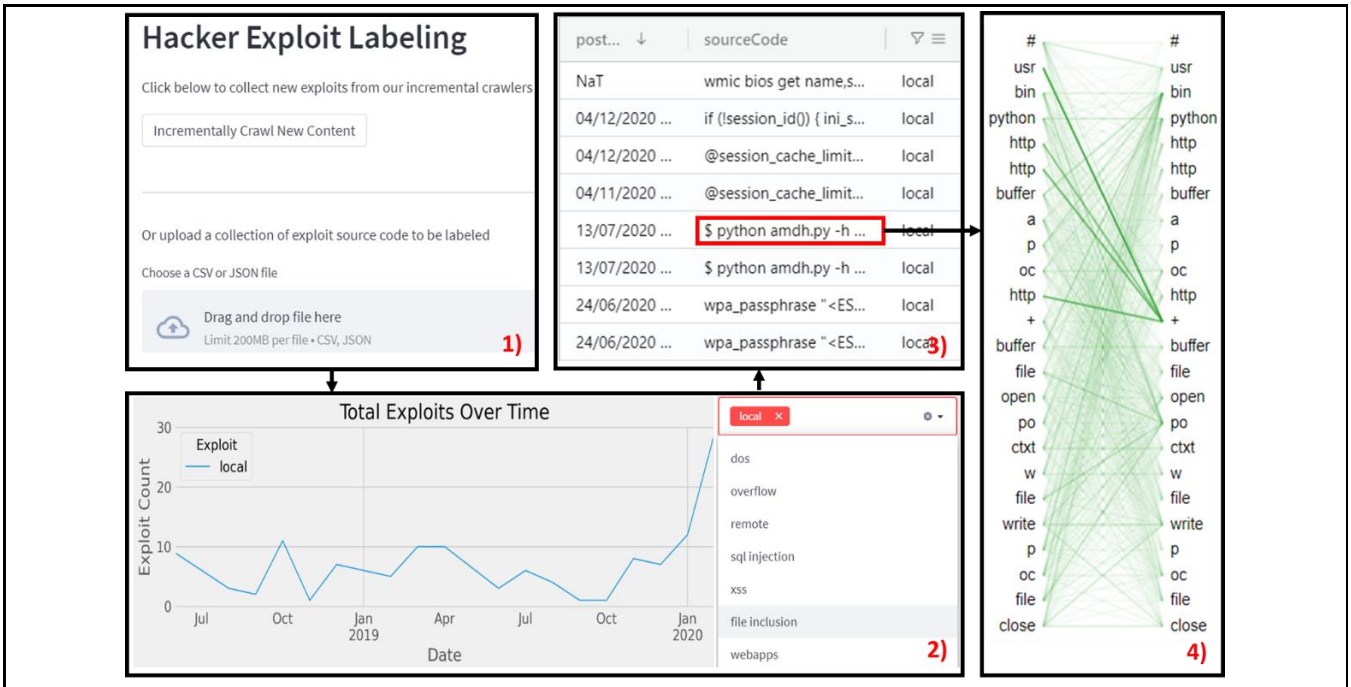

Figure D1. Screenshots Showing DTL-EL Integration into a UI. Users Can: (1) Upload or Crawl New Code, (2) Investigate Exploit Label Trends, (3) Study Recent or Relevant Exploits, and (4) Generate Semantic Dependencies for Any Exploit

[4] Within our gold-standard dataset, exploits labeled at 80% softmax achieved 94.87% accuracy and a 93.26% F1-score. Raising softmax higher than 80% did not lead to a statistically significant increase in accuracy/F1-score.

Appendix E

Cost-Sensitive Analysis for Exploit Labeling

Error minimization is commonly the goal of classification models. However, organizations will have different goals based on their internal cyber-risk tolerance, cyberinfrastructure, threat surfaces, etc. Therefore, DTL-EL should offer some flexibility to meet the goals of different organizations. Cost-sensitive analysis provides organizations with the flexibility to minimize cost instead of error by prioritizing the high-cost exploits specific to the organization (Kim et al., 2012). Therefore, we provide an example of cost-sensitive analysis using estimated costs of successful cyberattacks from reputable sources. In our cost matrix, a misclassified denial-of-service is $1,100,000,[5] local is $600,000,[2] remote is $2,500,000,[6] SQL injection is $196,000,[7] and web applications is $1,400,000.[3] Overflow is $100 as they often do not incur costs. Information on file inclusion and XSS exploits could not be found, so they were set to the closest exploit (local, web applications, respectively). Consistent with best practices in past literature, we compared error minimization to a cost-sensitive classifier and a MetaCost wrapper (Kim et al., 2012). A cost-sensitive classifier estimates class probabilities and uses them to minimize the expected cost at each prediction. MetaCost relabels training instances to estimate more accurate probabilities while predicting an exploits label. The results of each strategy by total cost (sum of mislabeled exploits of each type multiplied by the cost of that type), average misclassification cost (AMC, total cost divided by predictions), F1-score, and count of mislabeled exploits appear in Table E1.

The error minimization strategy had the highest F1-score (70.34%) compared to the cost-sensitive (66.14%) and MetaCost (69.16%) strategies. However, error minimization also led to the highest total cost ($276.2 million) and AMC ($57,042.54) among the three models. The cost-sensitive classifier strategy led to the lowest total cost ($245.8 Million) and AMC ($50,764.15) despite achieving the lowest F1-score (66.14%) and the highest number of mislabeled exploits (329). This difference suggests that the cost-sensitive strategy classifies high-cost exploits (e.g., remote) at the expense of misclassifying many lower-cost exploits (e.g., SQL injection). A cost-sensitive classifier therefore may be ideal for organizations prioritizing AMC and not mislabeled exploits. MetaCost achieved a close F1-score with error minimization (69.16%) while reducing total cost ($263.6 million) and AMC ($54,440.31). Organizations may choose MetaCost when the underlying predictive model produces inaccurate probabilities.

Table E1. Cost Comparison Between DTL-EL Model Strategies

Model strategy	Total cost	AMC	F1-score	Mislabeled exploits
Error minimization (DTL-EL)	$276.2 Million	$57,042.54	**70.34%**	**262**
Cost-sensitive classifier	**$245.8 Million**	**$50,764.15**	66.14%***	329
MetaCost	$263.6 Million	$54,440.31	69.16%	276

Note: $* p < 0.05,\ ** p < 0.01,\ *** p < 0.001$

[5] https://www.accenture.com/us-en/insights/security/invest-cyber-resilience
[6] https://purplesec.us/resources/cyber-security-statistics/
[7] https://www.helpnetsecurity.com/2014/03/28/analysis-of-three-billion-attacks-reveals-sql-injections-cost-196000