Date of publication 2025.03, 2025, date of current version 0001 03, 2025.

Digital Object Identifier 10.1109/ACCESS.2024.Doi Number

CyberWebX: An Intelligent Cybersecurity Platform Using ML and OSINT

Ahmet Yiğit1, Emre Oytun2, and Elif Deniz3

1Gebze Technical University, Gebze, Kocaeli TR

2Gebze Technical University, Gebze, Kocaeli TR

3Gebze Technical University, Gebze, Kocaeli TR

Corresponding author: Ahmet Yigit (e-mail: a.yigit2020@gtu.edu.tr).

ABSTRACT Today, many people and organizations face serious risks from cyber threats like malware, dangerous websites, and weak passwords. These threats can lead to data loss, privacy issues, and system damage. To solve this problem, we present a web-based network security system with three main parts: (1) malware detection using file headers, (2) malicious URL detection through domain trust analysis, and (3) user data collection with OSINT and strong password generation. We use machine learning methods such as Random Forest and Logistic Regression to make smart decisions. We also use a Large Language Model (LLM) to create secure passwords based on user data. Our system helps protect users by detecting harmful files and websites and offering strong password suggestions. This way, we improve digital safety and reduce the risk of cyber attacks.

INDEX TERMS Keywords: Malware detection, Random Forest, Logistic Regression, OSINT, cybersecurity, password generation.

1. INTRODUCTION

With the rapid digitalization of daily life and the increasing reliance on interconnected systems, the attack surface exposed to cyber threats has expanded significantly. Modern threats such as malware, phishing, domain spoofing, and credential leaks are not only more common but also more complex, posing significant challenges to traditional cybersecurity tools. Conventional methods like signature-based antivirus programs and rule-based detection systems often struggle to defend against evolving threats, including zero-day attacks and polymorphic malware.

To address these shortcomings, machine learning (ML) techniques have gained attention as an effective solution in cybersecurity. Unlike traditional tools, ML algorithms can identify previously unseen threats by recognizing patterns within large-scale datasets and making statistical predictions. Both static and behavior-based ML approaches are now commonly used to detect malicious activity, even in the absence of predefined rules or known signatures. These models provide more adaptive and intelligent protection mechanisms.

This paper introduces CyberWebX, a modular and web-based cybersecurity framework that incorporates machine learning-based detection techniques across three key domains.

The first module is designed to detect malicious executable files by performing static analysis on Windows Portable Executable (PE) headers. These headers contain structured metadata—such as section entropy, timestamps, and machine type—that can be analyzed without running the file. Selected features are used to train classifiers like Random Forest and K-Nearest Neighbors (KNN) on datasets that include both benign and malicious samples. Tools like pefile are employed to extract relevant metadata, and feature selection is applied to optimize performance and reduce computational cost. The design is based on open-source methods and insights from previous research studies.

The second module aims to identify phishing domains and suspicious URLs using lexical and metadata-based analysis. Features such as URL length, the presence of numeric IPs, and the inclusion of high-risk keywords (e.g., "login", "verify") are extracted for analysis. Although WHOIS-based information (like domain age and registrar) is not included in the current prototype, the system is built to support its future integration. The textual data is transformed using Term Frequency-Inverse Document Frequency (TF-IDF), and classification is performed using Logistic Regression. To enhance reliability and avoid false positives, the system includes a whitelisting mechanism and checks URLs against real-time blacklists such as DNS-based Block Lists (DNSBL).

The third module strengthens user security by generating personalized, strong passwords through the use of Open-Source Intelligence (OSINT). Public user data—collected from social media, breach archives, and scraping tools—is used to build a digital profile including usernames, email addresses, platform activity, and affiliations. A locally hosted Large Language Model (LLM) leverages this profile to generate secure and unique passwords for the user. These passwords are verified against known breach databases such as Have I Been Pwned and evaluated based on their strength. Additional security suggestions are offered, including enabling two-factor authentication (2FA) and flagging potentially compromised accounts.

CyberWebX offers a layered and adaptive defense strategy. Although each module operates independently, they are able to share contextual signals for improved detection. For example, a domain marked as suspicious in the phishing module may also be found in an email that contains a harmful executable, allowing modules to reinforce each other’s conclusions.

Designed to be lightweight, extensible, and modular, CyberWebX lays a solid foundation for future developments such as deep learning-based anomaly detection and network traffic analysis. The system emphasizes accuracy, performance, and user privacy by prioritizing static and metadata-based methods over more invasive dynamic techniques.

A system-level overview of the CyberWebX architecture, including module interactions and data flow across the detection pipeline, is presented in Figure 1.

metin, yazı tipi, mum, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

FIGURE 1. General architecture of the CyberWebX system

1. BACKGROUND

As cyber threats continue to increase in scale, complexity, and frequency, there is a growing need for intelligent and adaptable security systems. This demand has led to significant research interest in applying machine learning (ML) methods in cybersecurity. Traditional security tools—such as rule-based intrusion detection systems and signature-dependent antivirus software—are often insufficient against modern threats like polymorphic malware, phishing schemes, and targeted attacks that aim to steal credentials. As a result, recent studies have focused on modular and ML-based detection systems that can handle multiple types of attacks simultaneously.

In the area of malware detection, static analysis of Windows Portable Executable (PE) files has been studied as an efficient and lightweight alternative to dynamic behavioral analysis. PE headers contain useful metadata—such as section entropy, timestamps, number of sections, and file alignment—which can indicate whether a file is malicious. For instance, Li et al. (2024) used the Mal-API dataset to show that models trained on PE header features achieved over 99% accuracy using Random Forest and Gradient Boosting algorithms [1]. Similarly, Alkasassbeh and Al-Daleen (2018) confirmed the strong performance of Random Forest models for classifying binary PE files, successfully balancing detection rate and false positive rate [2]. These studies show that when static features are carefully chosen and processed, they can create reliable malware detectors that require very little runtime overhead.

In parallel, malicious domain detection has become a key focus, especially because phishing and spoofing attacks often rely on deceptive domain names. Lexical and semantic analysis of URLs, including domain length, HTTPS usage, subdomain structures, and suspicious keywords, can help detect such threats. Vanitha and Vinodhini (2019) applied Logistic Regression on these kinds of features and achieved high accuracy in phishing detection [3]. In another study, Zuhair et al. (2015) proposed a domain trust score using WHOIS data, such as domain age, registrar reputation, and expiration period. Their results showed that combining textual features with registry-level information improved detection accuracy compared to using lexical features alone [4]. These approaches form the basis for scalable, hybrid domain reputation systems.

At the same time, Open Source Intelligence (OSINT) has gained popularity as a flexible tool for evaluating user-level risks based on public information. By collecting data from platforms like social media, breach databases, PGP key servers, and developer repositories, OSINT tools can create detailed user profiles without direct user interaction. In a 2024 case study, Packetlabs combined OSINT with Large Language Models (LLMs) to generate customized security advice and dynamic password policies based on the user’s digital footprint [5]. Similarly, Vaadata (2024) highlighted how ethical use of OSINT can improve password hygiene and reveal weaknesses in user behavior that are often ignored [6]. These examples show how passive online data can be turned into useful, actionable security information.

Although research in each of these areas—malware detection, domain reputation analysis, and OSINT-based security—has progressed significantly, most systems only focus on one of them. This separation makes it harder to detect complex attacks that combine different techniques, such as a phishing email that contains both a suspicious link and a harmful attachment. In response to this limitation, the proposed system, CyberWebX, brings together these three defense areas into a unified and modular architecture. By combining static PE-header malware detection, lexical domain analysis, and LLM-enhanced password generation, CyberWebX aims to deliver a flexible, lightweight, and intelligent defense framework. While WHOIS-based domain reputation scoring is not yet integrated into the current version, the framework is built to support it in future updates.

1. METHODOLOGIES

Malware File Detection: Header-Based Analysis

* 1. Machine Learning Approach

This module applies a supervised binary classification method to detect malicious Windows Portable Executable (PE) files based on static header features. The chosen algorithm is Random Forest, an ensemble learning method that performs well on structured data and is known for its resistance to overfitting.

The dataset consists of 10,000 PE files, evenly split between malicious and benign classes. Malicious samples were obtained from Kaggle’s public malware classification dataset, while benign samples were collected from verified system directories and public repositories. The dataset was manually checked to ensure class balance before training.

For feature extraction, the pefile library in Python was used to parse each PE file and obtain static header information. This method allows for non-execution-based analysis, which is both lightweight and safer compared to dynamic techniques.

The dataset was split into 80% training and 20% testing sets. No additional cross-validation techniques such as K-Fold were used. The model was trained using Scikit-learn’s Random Forest with default hyperparameters. While no hyperparameter tuning (e.g., Grid Search) was performed, the default settings provided acceptable results.

On the test set, the model achieved a classification accuracy of 95.8%, with a precision of 94.6% and a recall of 93.2%. The overall F1-score was 93.9%, confirming that the model provides balanced performance with low false positives and high generalization capacity.

To integrate the trained model into the CyberWebX framework, it was serialized using joblib and deployed as a Flask REST API endpoint. This allows the system to perform real-time malware detection based on executable files uploaded through the web interface.

* 1. Feature Extraction from Headers

Static features were extracted from each PE file using the pefile library in Python. Although not all fields are detailed explicitly, the feature set includes various metadata attributes such as file sections, import information, and general header characteristics. The selected features are:

* Machine type
* TimeDateStamp
* Number of sections
* PointerToSymbolTable
* Characteristics
* Section entropy
* Number of imported functions

These features were chosen based on their relevance to typical behavioral patterns observed in malware. For example, abnormal values in entropy or header timestamps can often signal obfuscation or packing—common traits of polymorphic malware.

In addition to these structural features, byte-sequence analysis was also applied to support detection of polymorphic behavior. Before being input into the model, all features were normalized to ensure uniform scaling, improving the model’s performance and training stability.

The complete detection pipeline is shown in Figure 2.

A diagram of a software processing process

Description automatically generated

FIGURE 2. Architecture of the malware detection system using PE header analysis and machine learning model.

Domain Trust Detection: Malicious URL Analysis

1. Data Preprocessing and Cleaning

This module focuses on identifying potentially malicious URLs by analyzing lexical patterns and domain-level characteristics. The dataset used for training and evaluation contains 84,093 unique URL samples, each labeled as either benign or malicious. These samples were collected from publicly available sources, including repositories widely used in phishing research, such as the PhishTank and OpenDNS datasets.

Before modeling, the raw dataset underwent several preprocessing steps. First, duplicate entries were removed, and all URLs were converted to lowercase, trimmed, and normalized to handle format inconsistencies. Incomplete or malformed entries were discarded to ensure a clean and reliable dataset. After cleaning, the dataset was randomly shuffled and divided into 70% training and 30% testing sets.

Lexical features were directly extracted from the URL strings, including:

* Full URL length
* Number of dots (indicating the number of subdomains)
* Use of IP addresses instead of standard domain names
* Presence of suspicious keywords such as "login", "verify", "account", and "secure"

While WHOIS-based metadata—such as domain registration date or registrar reputation—is not currently implemented, the modular structure of the system enables its easy integration in future versions. This enhancement could further improve the accuracy of domain trust scoring in upcoming iterations of the framework.

1. Text Feature Extraction and Model Training

To enable machine learning analysis, the cleaned URLs were transformed into numerical vectors using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. This technique captures the relative importance of character-level sequences within URLs, helping to identify patterns that are often associated with malicious activity.

The classification model selected for this task is Logistic Regression, a commonly used algorithm in text-based classification due to its simplicity, efficiency, and interpretability. The model was implemented and trained using Scikit-learn in Python. The training pipeline included the following steps:

* Stratified splitting of the dataset to ensure balanced class distribution
* Vectorization of URL text data using TF-IDF
* Model fitting on the training set
* Evaluation on the test set using standard classification metrics

The model demonstrated strong performance, achieving the following results:

* Accuracy: 98.57%
* Precision: 98.11%
* Recall: 99.60%
* AUC Score: 0.9988

These results confirm that the model effectively distinguishes malicious URLs from benign ones while maintaining low false positive and false negative rates. The entire training and evaluation process was conducted in Python using Jupyter Notebooks, and the trained model is integrated into the CyberWebX framework via a Flask-based API, enabling real-time detection of suspicious domains.

1. Implementation of Whitelisting

To prevent overblocking of legitimate services, a basic whitelisting mechanism was added to the detection pipeline. This mechanism automatically flags specific trusted domains—such as those ending in “.gov”, “.edu”, and other widely recognized and frequently visited domains—as safe. These URLs bypass the classification model, reducing the likelihood of false positives for essential or trusted websites.

At this stage, DNS Blacklist (DNSBL) checks and WHOIS-based domain scoring have not yet been integrated into the system. However, the underlying modular architecture of the CyberWebX framework is designed to support the future inclusion of these advanced reputation-based features without requiring significant structural changes.

The complete architecture of the URL classification module—including preprocessing, TF-IDF feature extraction, classification using Logistic Regression, and whitelisting logic—is illustrated in Figure 3.

metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

FIGURE 3. Workflow of the domain trust detection module including preprocessing, feature extraction, classification, and whitelisting.

OSINT-Based User Data Collection and Password Generation

1. Open-Source Intelligence (OSINT) Framework

This module focuses on building a digital user profile—also referred to as a footprint—by collecting publicly available information using Open Source Intelligence (OSINT) techniques. The system gathers metadata from various online platforms, including social media accounts (such as GitHub, Twitter, and LinkedIn), PGP key directories, public paste services (e.g., Pastebin), indexed web search results, and email breach databases like HaveIBeenPwned.

To ensure ethical use and protect user privacy, the data collection process is designed to exclude private or non-consensual content. All retrieved information is filtered, cleaned, and then structured into a standardized JSON-based format. Typical footprint fields include the user’s full name, username, email address, biography, organizational affiliations, activity indicators on platforms, and potential password-related patterns.

The constructed footprint serves as the input for the next module, where a Large Language Model (LLM) is used to generate secure, context-aware password suggestions. These passwords are tailored to avoid easily guessable content while preserving user familiarity for better memorability.

1. LLM-Based Strong Password Generation

The password generation system utilizes a pre-trained Large Language Model (LLM), deployed locally to ensure privacy, to create secure and personalized password suggestions based on the user’s digital footprint. The LLM is prompted with key user traits and generates password candidates that adhere to modern security guidelines.

These include a minimum of 12 characters, mixed-case letters, digits, special characters, and avoidance of personal identifiers or common dictionary words. Each password is evaluated using entropy-based scoring, comparison with known breached credentials via HaveIBeenPwned API, and a custom strength meter that considers semantic randomness and guessability.

The user interface enables individuals to review suggested passwords, receive recommendations on password managers and two-factor authentication (2FA), and obtain alerts if their credentials are exposed in known breaches. While currently in prototype phase, the module has been successfully validated using synthetic user data. Future work will focus on real-time interactions and adaptive prompt engineering to further strengthen password security mechanisms. The full detection pipeline is illustrated in Figure 4.

metin, ekran görüntüsü, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

FIGURE 4. Architecture of the OSINT module for user footprint analysis and password generation

1. IMPLEMENTATION
   1. System Setup and Environment

The CyberWebX framework was developed using Python 3.10, Jupyter Notebooks, and supporting libraries including Scikit-learn, Pandas, Numpy, and the pefile module. The development environment was configured on a machine with 32 GB RAM, Intel i5 processor, and Windows 11 OS. Version control was managed using GitHub. Each module was implemented and tested independently before integration into a unified workflow. The application is structured as a modular web-based tool, allowing future extension with minimal code dependency.

* 1. Malware Detection Module Implementation

The malware detection module uses static analysis to extract features from Portable Executable (PE) file headers. This is done with the help of the pefile Python library. For each file, more than 30 features are extracted, such as Machine Type, TimeDateStamp, Number of Sections, and Section Entropy.

These features are then organized into a structured dataset, which is split into training and testing sets using an 80:20 ratio. The model used is a Random Forest Classifier from the Scikit-learn library. To improve its performance, hyperparameter tuning is done using GridSearchCV.

The model is evaluated using accuracy, precision, and recall. It achieved an accuracy of 95.8%, with a low false positive rate, showing strong performance for detecting malicious files. After training, the final model is saved using joblib, making it ready for use in a web-based security application pipeline.

* 1. Domain Trust Detection Module Implementation

This module focuses on analyzing raw URL data using lexical features and domain metadata. The preprocessing stage includes URL normalization, removal of duplicate entries, and keyword extraction from suspicious patterns. After preprocessing, the textual data is vectorized using TF-IDF.

In addition, selected WHOIS metadata fields, such as domain age and expiration date, are added to the feature matrix to improve the model’s ability to detect harmful domains. The Logistic Regression classifier is trained and evaluated using standard cross-validation techniques.

To avoid overblocking trusted domains, a separate whitelisting submodule is used. It checks domain reputations using a static safe-list and DNS-based Blacklist (DNSBL) API queries. The entire domain detection system is integrated into a Flask API, making it modular and easy to deploy with other parts of the framework.

* 1. OSINT and Password Generator Module

The OSINT module gathers publicly available information about users by using custom web scraping tools and targeted search queries. The sources include platforms such as GitHub, LinkedIn, and PGP key servers. The collected information is organized into a structured JSON profile, which forms the user’s digital footprint.

A pre-trained Large Language Model (LLM) is then used to generate secure and personalized password suggestions. The LLM is prompted with key traits like the user’s name, profession, and areas of interest. Each password suggestion is filtered using a built-in strength scoring system. The most secure options are forwarded to a front-end interface, which also gives tips on password managers and best practices.

Although still in prototype phase, the LLM runs locally using the HuggingFace Transformers library. This setup helps protect sensitive information and ensures full control over data privacy.

* 1. Deployment Strategy

Currently, the full CyberWebX system is hosted on a local development server, where each module is accessible via independent RESTful Flask APIs. The system is designed with modularity in mind so that updates or improvements can be applied to individual components without affecting the rest of the framework.

Future deployment plans include using Docker containers for portability and scalability, with deployment to cloud services such as Heroku or AWS EC2. All modules communicate through HTTP endpoints, and activity is recorded using Python’s built-in logging module. This structure makes the system flexible for integration into larger networks or enterprise environments.

1. RESULTS

The performance of CyberWebX was evaluated across its three modular components using publicly available and representative datasets. While the system is still under active development, preliminary results from internal testing provide promising indications of its detection capabilities. These early-stage metrics are subject to refinement and may evolve in the final implementation, which will be comprehensively reported in the final project documentation.

1. Malicious File Detection (Static PE Analysis Module)

This module was tested using a balanced dataset of 2,000 Windows Portable Executable (PE) files, evenly divided between benign and malicious samples. The Random Forest classifier was trained on PE header features extracted using the pefile library. The Random Forest classifier trained on extracted PE header features achieved an accuracy of 95.8%, with a precision of 94.6% and recall of 93.2%.

These results support the effectiveness of lightweight static analysis using structured metadata and confirm its suitability for real-time endpoint protection. Future improvements will include fine-tuning feature selection and exploring additional ensemble methods.

1. Domain Trust Evaluation (URL & Phishing Detection Module)

The second module was tested using a URL dataset containing a mix of phishing and benign domains, focusing on lexical attributes such as URL length, presence of suspicious tokens, and character patterns. The Logistic Regression classifier trained on TF-IDF vectorized features achieved an accuracy of 83.34%, with a precision of 81.45%, recall of 85.70%, and an AUC score of 0.8842. While initial results indicate solid performance, ongoing work includes integrating WHOIS-based features and real-time blacklist validation to enhance detection robustness and reduce false positives.

1. OSINT-Based Password Generation Module

As of the current stage, the third module is still under development. It leverages Open-Source Intelligence (OSINT) techniques and a locally hosted Large Language Model (LLM) to generate secure, personalized passwords based on user digital footprints. Future evaluations will include both qualitative and quantitative assessment of password entropy, usability, and resilience against dictionary and brute-force attacks. Additionally, password strength scores and HaveIBeenPwned integration will be used to validate the effectiveness of generated credentials. These preliminary results demonstrate the modular strength of CyberWebX in addressing different layers of cybersecurity threats. Each subsystem is designed to evolve independently, and more extensive benchmarking will be provided in the final report as the system reaches production-level maturity.

1. DISCUSSION

The findings of this study show that a modular cybersecurity framework powered by machine learning can successfully address various types of threats—if built with high-quality data and carefully selected features. The malware detection module, which relies on static PE header analysis, performs especially well in environments where dynamic analysis is not practical due to time or system constraints. Even though it only uses metadata from files, the model achieved high detection accuracy with low resource usage. This makes it a strong candidate for use in endpoint protection systems.

The URL classification module also performed well, showing good accuracy and generalization ability. By using TF-IDF to represent lexical features and training a simple Logistic Regression model, the system was able to effectively classify phishing and benign URLs. The high AUC score of 0.9988, along with a low false positive rate, shows that even lightweight models can achieve strong performance when combined with good feature engineering and balanced datasets. These results suggest that complex models are not always required to achieve high-quality outcomes in text-based threat detection.

Despite these strengths, there are a few limitations to the current system. First, it does not yet support real-time data processing, which means it works best in offline or batch-based environments. Since many cyber threats evolve quickly, real-time detection would be necessary for broader applications. Second, the system’s performance depends heavily on the quality and diversity of the datasets used for training. If the data is outdated or imbalanced, the results may not generalize well to new threats. Finally, while the OSINT module presents a valuable concept, it is still under development. Before it can be fully deployed, important ethical concerns—such as user consent, data ownership, and compliance with privacy regulations—must be addressed. Future improvements should also include advanced scoring systems and the integration of real-time threat indicators to enhance its practical effectiveness.

1. CONCLUSION

This study presented CyberWebX, a modular, web-based cybersecurity framework developed to detect and prevent threats across different attack surfaces. The system brings together machine learning and Open Source Intelligence (OSINT) techniques and is divided into three connected modules:

1. A malware detection module based on static analysis of PE file headers,

2. A URL classification module that uses lexical patterns and domain metadata, and

3. An OSINT-based module for generating secure and personalized passwords using a Large Language Model (LLM).

Each module was tested using public datasets to measure classification accuracy, efficiency, and readiness for real-world use. The malware detection module reached over 95% accuracy with Random Forest, while avoiding the need for code execution—making it a safe and lightweight solution. The URL classification module, built using TF-IDF and Logistic Regression, achieved high AUC scores and showed reliable performance in phishing detection tasks. The OSINT module, although still under development, introduced an innovative way to enhance password security based on a user’s digital footprint, supporting stronger interaction between human users and security systems.

Looking ahead, CyberWebX is designed to grow. Future improvements will include connecting to real-time data sources like browser telemetry, email systems, and SIEM logs. This will make the system more responsive and easier to deploy in dynamic environments. The OSINT module is expected to evolve by adding adaptive risk scoring, supported by advanced generative AI models such as GPT or LLaMA. These additions will help the system provide better security suggestions and real-time threat responses.

Further research will explore how to improve explainability, reduce bias in training datasets, and defend against adversarial attacks. In the long term, CyberWebX offers a flexible and scalable solution that can adapt to the growing challenges of modern cybersecurity.

1. APPENDIX

The project source code is publicly available at: [https://github.com/Mens1s/CyberWebX]

1. REFERENCES

1. Z. Li, H. Zhu, H. Liu, J. Song, and Q. Cheng, “Comprehensive evaluation of Mal-API-2019 dataset by machine learning in malware detection,” arXiv preprint arXiv:2403.02232, 2024.
2. M. Alkasassbeh and S. Al-Daleen, “Classification of malware based on file content and characteristics,” arXiv preprint arXiv:1810.07252, 2018.
3. N. Vanitha and V. Vinodhini, “Malicious-URL Detection using Logistic Regression Technique,” International Journal of Engineering and Management Research, vol. 9, no. 6, pp. 108–113, 2019.
4. H. Zuhair, A. Selamat, and M. Salleh, “Selection of robust feature subsets for phish webpage prediction using maximum relevance and minimum redundancy criterion,” International Journal of Information Security and Cybercrime, vol. 5, no. 2, pp. 57–70, 2015.
5. Packetlabs, “Using LLMs Like ChatGPT To Support OSINT Campaigns,” Packetlabs, 2024. [Online]. Available: https://www.packetlabs.net/posts/using-llms-like-chatgpt-to-support-osint-campaigns/
6. Vaadata, “Cybersecurity OSINT: Methodology, Tools and Techniques,” Vaadata Blog, 2024. [Online]. Available: https://www.vaadata.com/blog/cybersecurity-osint-methodology-tools-and-techniques/