**TEMASEK POLYTECHNIC**

**SCHOOL OF INFORMATICS & IT**

**AY2025/2026 OCTOBER SEMESTER**

**DIPLOMA IN APPLIED ARTIFICIAL INTELLIGENCE**

**DIPLOMA IN INFORMATION TECHNOLOGY**

### A black background with a black square Description automatically generated with medium confidence

**AI for Cybersecurity (CAI2C07)**

Project Report

|  |  |  |
| --- | --- | --- |
| Full Name | Admin No. | Class |
| Shubham Kaushik | 2301418F | T22 |

# ****Table of Contents****

| **Section No.** | **Title** | **Page** |
| --- | --- | --- |
| i | **Cover Page** | 1 |
| ii | **Introduction** | 3 |
|  | * Topic Overview |  |
|  | * Dataset of Choice |  |
| iii | **Model Training** | 3 |
|  | * Pipeline Overview |  |
|  | * Feature Engineering and Preprocessing Steps |  |
| iv | **Research on MITRE ATLAS Technique: Evade ML Model** | 6 |
|  | * What is MITRE ATLAS? |  |
|  | * Understanding “Evade ML Model” |  |
|  | * Real World Case Study |  |
| v | **Attack Strategy Plan** | 7 |
|  | * Homoglyph Substitution Attack |  |
|  | * Subdomain Padding Attack |  |
|  | * URL Encoding Obfuscation |  |
|  | * Random Character Padding |  |
|  | * IP Address Usage Instead of Domain |  |
| vi | **Defense Strategy Plan** | 8 |
|  | * Adversarial Training |  |
|  | * Ensemble Learning |  |
|  | * Input Normalization and Sanitization |  |
|  | * Feature Hardening with Semantic Attributes |  |
|  | * Uncertainty Estimation and Rejection Option |  |
| vii | **MITRE ATLAS Framework Implementation** | 11 |
|  | * Attack Demonstration: Evading the URL Classifier |  |
|  | * Defense Implementation: Hardening Against Evade ML Model |  |
|  | * Expected Outcome |  |
| viii | **References** | 13 |

1. **Introduction**

**Topic Overview**

As artificial intelligence (AI) and machine learning (ML) systems become increasingly a big part of cybersecurity, adversaries have developed advanced methods to exploit these technologies. One such attack method is Evading Machine Learning Models – the strategic manipulation of input data to cause a misclassification, thus bypassing AI-driven defenses.

This report explores the Evade ML Model technique, focusing on how adversaries can craft URLs that evade detection in a machine learning-based URL classifier. The objective is to understand how such attacks work and formulate strategic defenses based on the MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems) Framework.

**Dataset of Choice**

For this project, I utilized the publicly available Malicious and Benign URL Dataset from Kaggle – malicious\_phish.csv. It contains over 650,000 URLs labelled as malicious or benign, with a balanced distribution that aids supervised learning. Key attributes include:

* URL: Full URL text
* Label: 0 for benign, 1 for malicious

**A diagram of a model

Description automatically generated**The dataset is ideal for training a good binary classification model that identifies whether a URL is potentially harmful.

1. **Model Training**

Pipeline Overview

| **Feature Step** | **Description** | **Why** | **Assumptions** | **Potential Pitfalls** |
| --- | --- | --- | --- | --- |
| **Train-Test Split** | Split data before feature engineering | Avoid data leakage and biased performance | Future info must not leak into training | Wrong split could cause biased evaluation |
| **Label Distribution** | Check class imbalance | To plan for imbalance handling (scale\_pos\_weight, etc.) | Balanced classes → better performance | Imbalanced classes can cause bias toward major class |
| **URL Length Analysis** | Analyze distribution of URL lengths | Malicious URLs often unusually long to hide payloads | Longer URLs = more suspicious | Some benign URLs (e.g., e-commerce) are long |
| **Character Composition** | Count special characters | Malicious URLs often obfuscate using special characters | More special chars = higher risk | Benign URLs can also have special chars (e.g., search engines) |
| **Numerical Character in URLs** | Count numeric characters in URLs | Numbers often used for obfuscation or randomization | Many digits = more suspicious | Benign URLs sometimes have tracking IDs |
| **Top-Level Domain (TLD) Analysis** | Extract TLD from domain name | Some TLDs are more abused by attackers (e.g., .xyz, .ru) | Certain TLDs are higher risk | Attackers can use trusted TLDs too |
| **Keyword Presence in URLs** | Check for keywords like 'login', 'bank', 'secure' | Suspicious words indicate phishing or scams | Presence of keywords hints maliciousness | Not all keywords imply danger (false positives) |
| **Analyze Keyword Impact by Class** | Check keyword distribution across classes | Understand which classes use which keywords | Certain words are more common in malicious URLs | Over-reliance on keywords might miss unseen attacks |
| **Correlation Heatmap of Engineered Features** | Analyze correlations among features | To detect redundancy | Features should ideally not be highly correlated | Ignoring high correlation can lead to multicollinearity |
| **Lexical Feature Extraction** | Extract basic URL structure features | Capture simple patterns quickly | Lexical patterns can distinguish benign/malicious | Attackers adapt lexical patterns over time |
| **Primary Domain Extraction** | Extract primary domain (without subdomain) | Many malicious URLs use strange domains | Primary domain helps isolate brand impersonation | Hard to detect advanced domain impersonation |
| **Shortening Service Check** | Check for use of URL shorteners | Shortened URLs often used for hiding malicious content | Shortened URL = suspicious | Some legit services use shorteners |
| **Character Composition Features** | Letters, digits, special characters count | Capture structure complexity | Obfuscated URLs are structurally different | Natural variation across benign URLs |
| **Abnormal URL Structure** | Check for unusual domain structure | Malicious URLs often have abnormal structures | Abnormality = suspicious | Some benign URLs are also complex |
| **Secure HTTP (HTTPS)** | Check if URL uses HTTPS | HTTPS adds trust, phishing sites often skip it | HTTPS = more trustworthy | HTTPS can be spoofed too |
| **IP Address Check** | Check if IP is used instead of domain | IP-based URLs often evade domain-based detection | IP = suspicious | Some services use IPs legitimately |
| **Binary Flags for Suspicious URL Lengths** | Create flag for excessively long/short URLs | Capture extreme cases quickly | Extremes are suspicious | Legitimate edge cases exist |
| **Handling Missing Values in Domain Features** | Fill missing domains with 'unknown' | Prevent model errors on NaN values | Missing = unknown domain | Might not always imply malicious intent |
| **Lexical Features** | Combine basic lexical features | Strengthen simple URL feature representation | Simple features are easy to compute | Might not capture advanced attacks |
| **Drop Raw Text and Leakage-Prone Columns** | Drop URL text and pri\_domain after feature extraction | Prevent text leakage during training | No text leakage | Dropping too early may miss useful features |
| **Add Handcrafted Features** | Combine domain knowledge-based features | Enhance model understanding | Expert features add value | Needs careful feature selection |
| **Feature Scaling** | Scale numeric features if necessary (optional for trees) | Consistent feature ranges help some models | Scaling is critical for some models (e.g., Logistic Regression) | Not needed for tree models |
| **TF-IDF on Character n-grams (URL-based)** | Extract 2-5 char n-grams via TF-IDF | Capture meaningful URL substrings | Char patterns differentiate classes | High dimensionality, risk of overfitting |
| **TF-IDF Vectorization** | Vectorize URLs into numerical form | Make text data digestible for ML | Character patterns are strong indicators | Sparse representation |
| **TruncatedSVD for TF-IDF** | Reduce TF-IDF feature dimensions | Control overfitting and computation time | SVD preserves important patterns | Information loss possible |
| **Combine TF-IDF + Handcrafted** | Merge both feature types | Best of both worlds: semantic + structural | Combines strengths | High dimensionality if not careful |
| **Top-Level Domain (TLD)** | Extract domain extensions | Certain domains are risky (e.g., .xyz) | TLD gives hints about URL trustworthiness | Not a standalone feature |
| **Finding Feature Selection (Top K)** | Select top K important features | Focus on most relevant features | Top features improve model performance | May remove useful weak features |
| **Finding Highly Correlated Features** | Detect redundant features | Avoid multicollinearity and noise | Correlated features dilute model learning | Dropping correlated features loses info if not careful |
| **Remove Highly Correlated Features** | Drop one of highly correlated pairs | Simplify model and prevent redundancy | Keeps model clean | Dropping without domain knowledge risks losing important info |
| **Check for Low-Variance Features** | Remove features with low variance | Low variance features contribute little to the model | Focus only on informative features | Might drop rare but critical features |
| **Path-Based Feature Extraction** | Analyze URL path structure | Paths can reveal intent (e.g., /login/, /update/) | Paths often contain clues | Paths can be randomized |
| **Extract Abnormal Query** | Analyze query string for suspicious patterns | Attackers embed malicious payloads in queries | Query strings are often overlooked by simple filters | Noisy data in query strings |

1. **Research on MITRE ATLAS Technique: Evade ML Model [0.5 pages]**

**What is MITRE ATLAS?**

The MITRE ATLAS Framework catalogues real-world tactics and techniques adversaries use to attack ML systems. Defense Evasion is a key tactic, and within it lies Evade ML Model(T1632) – a method that focuses on input manipulation to bypass ML detection.

**Understanding “Evade ML Model”**

In this technique, adversaries:

* Generate adversarial examples – intentionally crafted inputs.
* Exploit model vulnerabilities – targeting how models generalize.
* Cause misclassification – malicious inputs classified as benign.

In URL detection, adversaries can manipulate the URL structure without affecting it malicious intent, thus evading detection systems.

**Real World Case Study**

A Domain Generation Algorithm (DGA) is a technique used by botnets and malware to dynamically generate many domain names that act as rendezvous points for command and control (C&C) servers.

DGAs enable malware to avoid detection and takedown by frequently changing domains, making them difficult to blacklist.

Hence, Machine Learning (ML) models are widely deployed to detect DGAs based on domain name features such as;

* Lexical structures (character n-grams, length, entropy)
* WHOIS information
* Domain registration patterns

However, adversaries have developed strategies to evade these ML detectors by making subtle changes altering the generated domain names to confuse the model without sacrificing functionality.

**Attack: DGA Detection Evasion**

Researchers demonstrated that:

* Minor perturbations to DGA domains (e.g inserting meaningful substrings, reducing randomness) can significantly negatively impact a ML model performance.
* Evade ML Model attacks are executed without disrupting the malware’s communication capabilities.
* By adversarial manipulating features such as:
  + Reducing the domain name’s entropy
  + Adding English dictionary words
  + Adjusting length and structure
* As a result, the researchers successfully evaded many classifiers that relied on statistical patterns.

1. **Attack Strategy Plan**
2. Homoglyph Substitution Attack
   1. One of the first attack strategy would be Homoglyphs. They are visually similar characters from different Unicode sets. For e.g o replaced with o -> Greek omicron. Attackers replace standard character with homoglyphs to deceive string-based detection models that heavily rely on character n-grams and lexical features. This is also states in the MITRE ATLAS T1632 Documentation “Homoglyph attacks are effective because models that rely on character-level features cannot distinguish between visually similar Unicode characters.”
   2. **Steps:** The attacker modifies URLs by replacing characters in suspicious words with usually similar Unicode characters (homoglyphs). For instance, the word “login” in a phishing URL could be altered to “1ogin”. These transformations preserve the human-readability and malicious function of the URL but disrupt the TF-IDF character n-grams on which the model relies.
   3. **Expected Output After Attack:** URLs that previously triggered a “malicious” classification are now misclassified as “benign” because the key token patterns used by the classifier are no longer recognizable.
3. Subdomain Padding Attack
   1. The next second attack strategy would be Subdomain Padding. Here attackers add random, benign-looking subdomains (e.g secure-update.paypal.com.fake.xyz) to exaggerate features such as subdomain count and domain depth. As stated in Security Research Journal (2023), “Appending meaningless subdomains can bypass detection systems by disrupting feature-based models that assume static domain structures.”
   2. **Steps**: The attacker appends multiple benign-looking subdomains to the original malicious URL. For example, malicious-site.com could become safe.bank.update.malicious-site.com. This exaggerates the subdomain count and weakens the malicious features extracted by the model, such as num\_subdomains or suspicious keyword presence.
   3. **Expected Output After Attack**: The model confused by the numerous safe-looking subdomains, reduces the anomaly score, resulting in a higher chance of benign misclassification.
4. URL Encoding Obfuscation
   1. The third attack strategy plan would be URL Encoding Obfuscation. Here, URL encoding involves replacing characters with %xx representations (e.g., space -> %20). Attackers can encode critical parts of the URL path or domain to obfuscate meaningful words. (e.g login becomes %6C%6F%67%69%6E)
   2. **Steps:** Key parts of the URL are obfuscated by URL encoding, converting characters like “/login.php” to “/%6C%6F%67%69%6E.php”. The URL remains functionally equivalent but presents a different character distribution to the model, disrupting TF-IDF and lexical feature extraction.
   3. **Expected Output After Attack**: The URL encoding breaks the critical patterns the model has learned, decreasing the model’s confidence in predicting it as malicious and increasing false negatives.
5. Random Character Padding
   1. The fourth attack strategy would be Random Character Padding. Here irrelevant characters or words are inserted into benign positions of the URL (e.g , https://paypal-login.safe-site.com becomes <https://paxypal-login.safe-site.com>) which reduces the semantic consistency the model relies on. I would like to reference from the Adversarial Machine Learning Textbook (2021), “Even minor perturbations in benign locations of an input can create a significant shift in feature distributions.”
   2. **Steps**: The attacker introduces random benign-looking tokens into URLs without changing their function. For instance, secure-update-login-site.com could become safe-secure-update-login-site-info.com. These injected token dilute suspicious word frequencies and change the lexical signature.
   3. **Expected Output After Attack**: The classifiers undergo a reduction the proportion of malicious tokens, leading to decreased detection rates and increased misclassification of the malicious URLs as benign
6. IP Address Usage Instead of Domain
   1. The fifth and final attack strategy would be IP Address Usage Instead of Domain. Instead of uding domains names, attackers craft URLs directly using IP addresses (e.g., <http://192.168.1.10/login>). I would like to quote from SANS Institute Whitepaper On Adversarial ML(2021),“ Replacing domain names with IP addresses effectively bypasses models reliant on domain structure and lexical features.”
   2. Steps: Instead of using domain names, the attacker construct URLs using direct IP addresses (e.g, <http://203.0.113.5/login>). This eliminates the use of domain-based lexical features and WHOIS information that would have as a result been used by the model.
   3. Expected Output After Attack: Since the model is not provided domain-related signals which are important in an model’s ability to distinguish malicious URLs drops, allowing many malicious IP-based URLs to bypass detection without any alerts.
7. **Defense Strategy Plan**
8. Adversarial Training
   1. The first defense strategy plan would be Adversarial Training. It is where we introduce adversarial example (they are crafted using the above attack methods) into the training set. This forces the model to learn not only clean examples but also recognize manipulated URLs. I would quote MITRE ATLAS Defense Techniques Guide, “Adversarial training remains one of the most effective defences against evasion attacks, making models more resilient to adversarial inputs.”
   2. **Steps**: Integrate adversarial crafter samples (homoglyphs, obfuscated, encoded URLs) into the training data. The model is retrained on augmented dataset to ensure it learns to recognize obfuscation patterns.
   3. **Justification**: According to the MITRE ATLAS. Adversarial Threat Framework, adversarial training improves model resilience by exposing it to attack variations during learning. This approach hardens the model against input perturbations without any significant changes to the architecture.
9. Ensemble Learning
   1. The next second defense strategy plan would be Ensemble Learning. Predictions here are combined from multiple diverse models (e.g., Random Forest, XGBoost, Logistics Regression). Each model captures different feature aspects, reducing single-model blind spots. This has been stated in Journal of Machine Learning Research (JMLR), “Model ensembles leverage diversity, making it harder for a single adversarial perturbation to fool the entire system.”
   2. **Steps**: Deploy an ensemble of multiple classifiers (e.g XGBoost, Random Forest, and Logistics Regression). Using soft voting or stacking techniques that would aggregate predictions, ensuring that no single model’s weakness can be fully exploited.
   3. **Justification**: As per research in the Journal of Machine Learning Research (JMLR), ensembles reduce the risk of successful adversarial evasion because perturbations that fool one model are unlikely to fool all. Hence, this significantly increases robustness under white-box attack conditions.
10. Input Normalization & Sanitization
    1. Our third defense strategy plan would be Input Normalization & Sanitization. URLs are preprocessed by decoding URL-encoded characters and normalize Unicode characters (e.g., converting homoglyphs to standard forms) before feature extractions. I would like to back up my strategy plan by quoting Google AI Security Whitepaper (2023),“ Robust input pre-processing disrupts common adversarial tactics like encoding and Unicode manipulation.”
    2. **Steps**: Implement a robust preprocessing pipeline that decoded URL encodings, normalizes Unicode homoglyphs to standard ASCII equivalents, and strips unnecessary subdomains before the URL reaches the classifier.
    3. **Justification**: Input sanitization defends against obfuscation at the ingestion phase. This was by Google’s AI Red Team studies, sanitizing input before model evaluation was found out to be a critical frontline defense against encoding and lexical manipulation attacks.
11. Feature Hardening with Semantic Attributes
    1. Introduce sematic features such as SSL certificate validation, domain age, WHOIS information, blacklist history – features that are difficult to fake by simple URL manipulation. I would like to reference MITRE ATLAS AI Red Teaming Guide, “Semantic and behavioral features significantly increase the difficulty of crafting successful evasion attacks.”
    2. **Steps**: Improve your feature space with semantic attributes other than using lexical patterns, such as SSL certificate validity, WHOIS domain age, IP reputation scores, and geolocation of hosting servers. These features are harder for attackers to manipulate dynamically.
    3. **Justification**: MITRE’s Adversarial Machine Learning Taxonomy highlights that semantic features reduce susceptibility to surface-level adversarial attacks. By incorporating features that require significant real-world effort to modify, this leads to model’s defenses become more stable, reliable, and durable.
12. Uncertainty Estimation and Rejection Option
    1. The fifth defense strategy is Uncertainty Estimation and Rejection Option. Here the defense strategy equips the model with a mechanism to abstain from making a prediction if the model’s confidence is low-flagging suspicious samples for manual inspection. I would like to quote from IEE Transactions on Dependable and Secure Computing (2022), “Rejecting low-confidence predictions enhances the robustness of classifiers against unseen adversarial examples.”
    2. **Steps**: We introduce a confidence threshold mechanism: if the model’s predicted probability falls within an uncertain range (e.g, 0.4-0.6), the prediction is flagged for manual review rather than being auto accepted.
    3. **Justification**: Uncertainty estimation was emphasized in the literature on robust ML deployments, allowing models to abstain from making risky predictions on adversarial examples. This reduces the chance of confident misclassifications and allows human-in-the-loop intervention when needed.
13. **MITRE ATLAS Framework Implementation**

**Techniques: Evade ML Model (T1632) in URL Detection**

In the context of URL classification, adversaries apply Evade ML Model by crafting adversarial URLs – URLs modified in ways that confuses machine learning models without affecting the underlying malicious behaviors.

The model that I deployed – a TF-IDF feature extractor combined with an XGBoost classifier is able to analyze URLs for specific lexical features and patterns. These patterns become the model’s “weak points” due to subtle changes to the URL structure which can alter feature extraction outcomes, allowing attackers to escape detection.

*Attack Demonstration: Evading the URL Classifier*

Attack Scenario 1: Homoglyph Substitution

Step-by-step:

1. **Baseline:** A malicious URL https://secure-login.banjofmoney.com is correctly flagged as malicious
2. **Attack:** Replace characters with visually similar Unicoode homoglyphs:
   1. <http://ѕесure-lоgin.bankofmoney.com>
   2. (‘s’ -> ‘ѕ’, ‘e’ -> ‘е’, ‘o’ -> ‘о’ — Cyrillic homoglyphs)
3. **Expected Result:**
   1. TF-IDF character n-grams, word frequencies, and suspicious keyword matching will not trigger correctly.
4. **Why it works:**
   1. Feature extraction relies on ASCII matching and does not normalize Unicode, leading to feature evasion.
   2. **Keyword detection** (e.g, login, secure) fails

Attack Scenario 2: Subdomain padding

Step-by-step:

1. **Baseline:** Malicious URL <http://update-account-secure.bank.com>
2. **Attack:** Add multiple harmless-looking subdomains:
   1. <http://help-center-support-info.update-account-secure.bank.com>
3. **Expected Result:**
   1. Dilution of suspicious features
   2. Higher subdomain count and overall URL length disrupts the learned feature distributions.
   3. The classifier as a result gets confused and potentially misclassifies the URL as benign.
4. **Why it works:**
   1. ML model relies on path length, number of subdomains, and keyword presence.
   2. Adding legitimate-looking subdomains shifts feature vectors towards benign URL patterns.

*Defense Implementation: Hardening Against Evade ML Model*

Defense Measure 1: Input Normalization and Unicode Sanitization

**Action:**

* Implement Unicode normalization (e.g, NFKC normalization) and homoglyph detection prior to feature extraction.
* Map homoglyphs back to standard ASCII equivalents

**Impact:**

* Neutralizes Homoglyphs Substitution
* Prevents evasion at the string level before it affects feature engineering.

Defense Measure 2: Adversarial Training with Modified URLs

**Action:**

* Continuously generate adversarial examples (homoglyphs substitutions, subdomain paddings) and incorporate them into training data.
* Fine-tune the model periodically with these adversarial examples.

**Impact:**

* Increase robustness to real-world evasions.
* Model learns not just from clean data but also from likely adversarial manipulations.

**Expected Outcome:**

After launching the attacks, without defense mechanisms in place, the classifier is expected to misclassify the adversarial manipulated URLs as benign. The homoglyph substitution attack causes the model to fail in detecting important suspicious keywords due to Unicode character alterations, while the subdomain padding attack distorts key features like subdomains count and URL structure, confusing the model’s learned decisions boundaries. As a result, malicious URLs successful bypass detection, posing a significant security risk. However once the defenses have been implemented – Unicode normalization makes sure that homoglyphs are mapped back to standard ASCII, restoring accurate keyword matching, while adversarial training exposes the model to keyword matching, enhancing its robustness. This leads to a strong improvement in detection performance, where the model accurately identifies even subtly altered malicious URLs, minimizing false negative and significantly strengthening the system’s overall resilience against evasion attempts.

**References (IEEE Format):**

[1] MITRE Corporation, “MITRE ATLAS Knowledge Base,” *MITRE*, 2024. [Online].

Available: <https://atlas.mitre.org>

[2] Unicode Consortium, “Unicode Security Considerations,” *Unicode Technical Report #36*, 2024. [Online]. Available: <https://www.unicode.org/reports/tr36/>

[3] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, “Analyzing the robustness of malicious website detectors against adversarial inputs,” *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 151–156, 2017, doi: 10.1109/EuroSPW.2017.53.

[4] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in *Proc. 23rd ACM Conference on Computer and Communications Security (CCS)*, pp. 1528–1540, 2016, doi: 10.1145/2976749.2978392.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2015. [Online]. Available: <https://arxiv.org/abs/1412.6572>

[6] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, “The limitations of deep learning in adversarial settings,” in *IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, 2016, doi: 10.1109/EuroSP.2016.36.

[7] M. Rigaki and F. G. Schmidt, “Adversarial examples for malware detection,” *arXiv preprint arXiv:1702.05983*, 2017. [Online]. Available: <https://arxiv.org/abs/1702.05983>.

[8] Kaggle, “Malicious and Benign URL Dataset,” *Kaggle*, 2022. [Online]. Available: <https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset>

[9] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017, doi: 10.1109/SP.2017.49.