



Ataki na AI, środki neutralizujące i strategie obronne

Przewodnik dla specjalistów do spraw cyberbezpieczeństwa
po atakach adversarialnych, modelowaniu zagrożeń
i wprowadzaniu zabezpieczeń zgodnych ze strategią MLSecOps



JOHN SOTIROPOULOS

Tytuł oryginału: Adversarial AI Attacks, Mitigations, and Defense Strategies:
A cybersecurity professional's guide to AI attacks, threat modeling,
and securing AI with MLSecOps

Tłumaczenie: Zbigniew Waśko

ISBN: 978-83-289-2235-8

Copyright © Packt Publishing 2024. First published in the English language under the title 'Adversarial AI Attacks, Mitigations, and Defense Strategies – (9781835087985)'

Polish edition copyright © 2025 by Helion S.A.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/atanai

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: *helion@helion.pl*

WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- **Lubię to! » Nasza społeczność**

Spis treści |

O autorze	17
O recenzentach	17
Zastrzeżenie	18
Wstęp	19

CZĘŚĆ 1. Wprowadzenie do adversarialnej sztucznej inteligencji

ROZDZIAŁ 1

Wprowadzenie do sztucznej inteligencji	27
Podstawy sztucznej inteligencji i uczenia maszynowego	28
Rodzaje uczenia maszynowego i jego cykl życia	29
Kluczowe algorytmy uczenia maszynowego	32
Sieci neuronowe i uczenie głębokie	33
Narzędzia deweloperskie do programowania uczenia maszynowego	35
Podsumowanie	36
Dodatkowe publikacje	37

ROZDZIAŁ 2

Przygotowanie adversarialnego laboratorium	38
Wymagania techniczne	39
Konfiguracja środowiska programistycznego	39
Instalacja Pythona	39
Tworzenie wirtualnego środowiska	40
Instalowanie pakietów	41
Powiązanie środowiska wirtualnego z notatnikami Jupytera	41
Weryfikowanie instalacji	41
Praktyczne podstawy uczenia maszynowego	42
Proste sieci neuronowe	45
Tworzenie docelowej usługi AI z użyciem sieci CNN	46
Konfiguracja i gromadzenie danych	46

Eksploracja danych	47
Wstępne przetwarzanie danych	47
Wybór algorytmu i tworzenie modelu	48
Trenowanie modelu	48
Ewaluacja modelu	49
Wdrażanie modelu	50
Usługa wnioskowania	50
Tworzenie rozwiązań ML na dużą skalę	51
Google Colab	52
AWS SageMaker	52
Usługi Azure Machine Learning	52
Lambda Labs Cloud	53
Podsumowanie	53

ROZDZIAŁ 3

Bezpieczeństwo a adversarialna sztuczna inteligencja	55
Wymagania techniczne	56
Podstawy bezpieczeństwa	56
Modelowanie zagrożeń	57
Ryzyka i środki zaradcze	57
DevSecOps	58
Zabezpieczenie naszego adversarialnego laboratorium	58
Bezpieczeństwo hosta	60
Ochrona sieci	63
Uwierzytelnianie	65
Ochrona danych	66
Kontrola dostępu	69
Zabezpieczanie kodu i artefaktów	70
Bezpieczeństwo kodu	71
Zabezpieczanie zależności przez wyszukiwanie podatności	71
Skanowanie sekretów	74
Zabezpieczanie notatników Jupytera	74
Zabezpieczanie modeli przed złośliwym kodem	76
Integracja z potokami DevSecOps i MLOps	77
Omijanie zabezpieczeń za pomocą adversarialnej sztucznej inteligencji	77
Nasz pierwszy adversarialny atak	78
Tradycyjne cyberbezpieczeństwo a adversarialna sztuczna inteligencja	80
Adversarialna sztuczna inteligencja	81
Podsumowanie	81

CZĘŚĆ 2. Atakowanie modeli ML

ROZDZIAŁ 4

Ataki zatruwające	85
Podstawy ataków zatruwających	86
Definicja i przykłady	86
Rodzaje ataków zatruwających	87
Przykłady ataków zatruwających	88
Dlaczego to takie ważne?	89
Przygotowanie prostego ataku zatruwającego	90
Tworzenie zatrutych próbek	90
Ataki zatruwające typu backdoor	94
Tworzenie wyzwalaczy backdoorowych za pomocą narzędzi ART.	98
Zatrucie danych z użyciem frameworku ART.	102
Ataki backdoorowe z ukrytym wyzwalaczem	103
Ataki typu clean-label	104
Zaawansowane ataki zatruwające	106
Zapobieganie i obrona	107
Obrona cyfrowych twierdz przy użyciu MLOps	107
Wykrywanie anomalii	108
Testy odporności na zatrucie	109
Zaawansowana ochrona przed zatruciem realizowana z użyciem narzędzi ART.	110
Trening adversarialny	112
Budowanie strategii obronnej	112
Podsumowanie	113

ROZDZIAŁ 5

Manipulowanie modelami za pomocą koni trojańskich i przeprogramowywania	114
Wstrzykiwanie backdoorów za pomocą serializacji pickle'owej	115
Przebieg ataku	115
Obrona i środki zaradcze	117
Wstrzykiwanie koni trojańskich za pomocą kerasowych warstw lambda ...	118
Przebieg ataku	118
Obrona i środki zaradcze	122
Warstwy niestandardowe z końmi trojańskimi	124
Przebieg ataku	124
Obrona i środki zaradcze	126

Wstrzykiwanie ładunku neuronowego	127
Przebieg ataku	128
Obrona i środki zaradcze	131
Atakowanie brzegowej sztucznej inteligencji	132
Aplikacja mobilna ImRecS na Androida	132
Przebieg ataku	133
Obrona i środki zaradcze	134
Przejmowanie kontroli nad modelem	137
Wprowadzenie kodu konia trojańskiego	137
Przeprogramowanie modelu	138
Podsumowanie	139

ROZDZIAŁ 6

Ataki na łańcuch dostaw a adversarialna sztuczna inteligencja 140

Tradycyjne zagrożenia dla łańcucha dostaw a sztuczna inteligencja	141
Zagrożenia związane z przestarzałymi i podatnymi na atak komponentami	141
Ryzyka wynikające z zależności AI od danych pobieranych na żywo	143
Zabezpieczanie AI przed podatnymi na atak komponentami	145
Zaawansowane zabezpieczenia — akceptowanie jedynie zatwierdzonych pakietów	150
Konfiguracja klienta do obsługi prywatnych repozytoriów PyPI	151
Dodatkowe zabezpieczenia prywatnego repozytorium PyPI	152
Korzystanie ze specyfikacji SBOM	153
Ryzyka związane z łańcuchem dostaw w kontekście AI	155
Uczenie transferowe jako broń obosieczna	155
Zatrutowanie modelu	156
Manipulowanie modelami	161
Sprawdzanie pochodzenia wstępnie wytrenowanych modeli i nadzorowanie ich	163
MLOps a prywatne repozytoria modeli	164
Zatrutowanie danych	168
Ryzyka związane z łańcuchem dostaw	168
Zatrutowanie danych, aby wpłynąć na wyniki analizy sentymentu	169
Obrona i środki zaradcze	171
Specyfikacje komponentów oprogramowania (SBOM) dla AI/ML	172
Podsumowanie	172

CZĘŚĆ 3. Ataki na implementacje AI

ROZDZIAŁ 7

Ataki unikowe na implementacje AI	177
Podstawy ataków unikowych	177
Znaczenie znajomości ataków unikowych	178
Techniki rozpoznawcze w atakach unikowych	179
Perturbacje i techniki ataków unikowych w kontekście obrazów	182
Scenariusze ataków unikowych	183
Jednoetapowa perturbacja z użyciem metody FGSM	184
Podstawowa metoda iteracyjna (BIM)	187
Atak z mapą istotności wyznaczonej metodą Jacobiego (JSMA)	188
Atak Carliniego i Wagnera (C&W)	190
Rzutowanie gradientu (PGD)	192
Łatki adversarialne — łączenie cyfrowych i fizycznych technik unikania	193
Ataki unikowe w dziedzinie NLP — atak na model BERT z użyciem biblioteki TextAttack	195
Scenariusz ataku — analiza sentymentu	195
Przykład ataku	195
Scenariusz ataku — wnioskowanie w języku naturalnym	197
Przykład ataku	197
Uniwersalne perturbacje adversarialne (UAP)	198
Scenariusz ataku	199
Przykład ataku	199
Ataki czarnoskrzynkowe oraz ich transferowalność	200
Scenariusz ataku	201
Przykład ataku	201
Obrona przed atakami unikowymi	202
Przegląd strategii obronnych	202
Trening adversarialny	203
Wstępne przetwarzanie danych wejściowych	204
Techniki wzmacniania modelu	205
Zespoły modeli	207
Certyfikowane mechanizmy obronne	208
Podsumowanie	209

ROZDZIAŁ 8

Ataki na prywatność — kradzież modeli	210
Charakterystyka ataków na prywatność	210
Wykradanie modeli podczas ataków ekstrakcyjnych	211
Ekstrakcja równoważna funkcjonalnie	212
Ataki ekstrakcyjne oparte na uczeniu	214
Generatywne ataki ekstrakcyjne typu uczeń-nauczyciel (ataki destylacyjne)	219
Przykładowy atak na model CIFAR-10 CNN	222
Obrona i środki zaradcze	227
Środki prewencyjne	227
Mechanizmy wykrywające	231
Ustalanie stanu własności modelu oraz jej odzyskiwanie	232
Podsumowanie	235

ROZDZIAŁ 9

Ataki na prywatność — kradzież danych	236
Istota ataków polegających na inwersji modelu	236
Typy ataków inwersyjnych	238
Wykorzystanie poziomów pewności modelu	238
Inwersja modelu wspomagana sieciami GAN	240
Przykład ataku inwersyjnego	246
Istota ataków wnioskowania	248
Ataki wnioskowania o atrybutach	248
Metaklasyfikatory	249
Wnioskowanie wspomagane zatruciem	249
Przykład ataku wnioskowania o atrybutach	251
Ataki wnioskowania o przynależności	253
Statystyczne wartości progowe wycieku danych z modeli ML	254
Atak transferu wiedzy z wykorzystaniem samych etykiet	255
Ślepe ataki wnioskowania o przynależności	255
Ataki białoskrzynkowe	256
Przykładowy atak wnioskowania o przynależności z użyciem narzędzi ART.	259
Podsumowanie	260

ROZDZIAŁ 10

Zachowanie prywatności w rozwiązaniach AI	262
Zachowanie prywatności w rozwiązaniach ML i AI	263
Prosta anonimizacja danych	264

Zaawansowana anonimizacja	268
K-anonimowość	268
Anonimizacja a dane geolokalizacyjne	270
Anonimizacja formatów multimedialnych	272
Prywatność różnicowa	278
Uczenie federacyjne	281
Uczenie dzielone	282
Zaawansowane opcje szyfrowania wspomagające ochronę prywatności w uczeniu maszynowym	283
Bezpieczne obliczenia wielostronne	283
Szyfrowanie homomorficzne	285
Praktyczne zastosowanie zaawansowanych technik szyfrowania w rozwiązaniach ML	287
Stosowanie technik zapewniania prywatności w uczeniu maszynowym	289
Podsumowanie	290

CZĘŚĆ 4. Generatywna sztuczna inteligencja a ataki adversarialne

ROZDZIAŁ 11

Generatywna sztuczna inteligencja — nowy front walki	293
Krótkie wprowadzenie do generatywnej sztucznej inteligencji	294
Krótka historia rozwoju generatywnej sztucznej inteligencji	294
Technologie generatywnej sztucznej inteligencji	295
Stosowanie generatywnych sieci adversarialnych	298
Tworzenie sieci GAN od podstaw	299
Sieci WGAN i niestandardowe funkcje strat	307
Korzystanie ze wstępnie wytrenowanych modeli GAN	308
Pix2Pix	308
CycleGAN	309
Pix2PixHD	309
PGGAN	310
BigGAN	310
StarGAN v2	311
Seria StyleGAN	311
Podsumowanie	312

ROZDZIAŁ 12**Wykorzystywanie sieci GAN do deepfake'ów****i ataków adversarialnych 313**

Wykorzystanie GAN-ów do tworzenia deepfake'ów oraz ich wykrywania	313
Generowanie przekonujących fałszywych zdjęć za pomocą modeli StyleGAN	314
Wykorzystanie sieci GAN do tworzenia prostych deepfake'ów na podstawie istniejących zdjęć	318
Wprowadzanie ukierunkowanych zmian w istniejącym obrazie	320
Syntezowanie obrazów za pomocą modelu Pix2PixHD	321
Sfałszowane filmy i animacje	324
Inne techniki tworzenia deepfake'ów	325
Deepfaki dźwiękowe	329
Wykrywanie deepfake'ów	330
Zastosowanie GAN-ów w cyberatakach i bezpieczeństwie ofensywnym	333
Omijanie systemów weryfikacji twarzy	333
Oszukiwanie biometrycznych mechanizmów uwierzytelniających	335
Łamanie haseł z użyciem GAN-ów	337
Omijanie mechanizmów detekcji złośliwego oprogramowania	340
GAN-y w kryptografii i steganografii	342
Generowanie ładunków dla ataków sieciowych z użyciem GAN-ów	344
Generowanie ładunków dla ataków adversarialnych	345
Mechanizmy obronne i środki zaradcze	348
Zabezpieczanie sieci GAN	348
Ataki adversarialne wspomagane sieciami GAN	349
Deepfaki, złośliwe treści i szerzenie dezinformacji	351
Podsumowanie	355

ROZDZIAŁ 13**Podstawy LLM w kontekście adversarialnej sztucznej inteligencji 377**

Krótkie wprowadzenie do dużych modeli językowych	357
Tworzenie aplikacji AI z użyciem dużych modeli językowych	359
„Hello LLM” w Pythonie	360
„Hello LLM” w LangChainie	365
Wprowadzanie własnych danych	366
Wpływ dużych modeli językowych na adversarialną sztuczną inteligencję	370
Podsumowanie	371

ROZDZIAŁ 14

Ataki adversarialne z użyciem promptów	372
Adversarialne dane wejściowe i wstrzykiwanie promptów	373
Bezpośrednie wstrzykiwanie promptów	374
Zastępowanie promptów	375
Wstrzykiwanie stylu	378
Odgrywanie ról	378
Podszywanie się	382
Inne techniki jailbreakingowe	389
Zautomatyzowane wstrzykiwanie promptów z użyciem technik gradientowych	391
Ryzyko związane z wprowadzaniem własnych danych	393
Pośrednie wstrzykiwanie promptów	393
Wydobywanie danych za pomocą wstrzykiwania promptów	397
Eskalacja uprawnień za pomocą wstrzykiwania promptów	398
Zdalne wykonywanie kodu przez wstrzykiwanie promptów	399
Mechanizmy obronne i środki zaradcze	401
Mechanizmy obronne platformy LLM	402
Mechanizmy obronne na poziomie aplikacji	403
Podsumowanie	411

ROZDZIAŁ 15

Ataki zatruwające a modele LLM	412
Zatrucie osadzeń w mechanizmie RAG	412
Scenariusze ataku	419
Zatrucie podczas generowania osadzeń	420
Bezpośrednie zatrucie osadzeń	427
Zaawansowane zatrucie osadzeń	428
Manipulowanie osadzeniami zapytań	432
Mechanizmy obronne i środki zaradcze	433
Ataki zatruwające proces dostrajania modeli LLM	435
Wprowadzenie do dostrajania modeli LLM	435
Scenariusze ataków zatruwających podczas dostrajania	438
Wektory ataku na proces dostrajania	440
Zatrucie bota ChatGPT-3.5 przez dostrajanie	441
Mechanizmy obronne i środki zaradcze chroniące przed zatruciem procesu dostrajania	451
Podsumowanie	455

ROZDZIAŁ 16**Zaawansowane scenariusze z wykorzystaniem**

generatywnej sztucznej inteligencji	456
Ataki na łańcuch dostaw w kontekście LLM-ów	457
Publikowanie zatrutego LLM-u w serwisie Hugging Face	459
Publikowanie zmanipulowanego modelu LLM w serwisie Hugging Face	465
Inne zagrożenia związane z łańcuchem dostaw w kontekście modeli LLM	467
Mechanizmy obronne i środki zaradcze w kontekście łańcucha dostaw	468
Ataki na prywatność w kontekście modeli LLM	469
Ataki polegające na inwersji modelu i ekstrakcji danych treningowych w kontekście modeli LLM	470
Ataki wnioskowania na modele LLM	472
Klonowanie jednego modelu LLM przy użyciu drugiego	474
Mechanizmy obronne i środki zaradcze w kontekście ataków na prywatność	476
Podsumowanie	477

CZĘŚĆ 5. Zabezpieczanie AI przez projekt i praktyki MLSecOps

ROZDZIAŁ 17

Koncepcje Secure by Design i Trustworthy AI	481
Secure by Design — zabezpieczanie AI już na etapie projektowania	482
Budowanie biblioteki zagrożeń	485
Tradycyjne cyberzagrożenia	486
Ataki adversarialne	486
Ataki adversarialne specyficzne dla generatywnej AI	488
Ataki na łańcuch dostaw	489
Branżowe klasyfikacje zagrożeń dla AI	490
Porównania klasyfikacji zagrożeń dla AI	491
Porównanie z klasyfikacją NIST AI	491
Porównanie z klasyfikacją AI Exchange	495
Porównanie z klasyfikacją MITRE ATLAS	498
Modelowanie zagrożeń dla AI	501
Modelowanie zagrożeń w praktyce	502
Przykładowe rozwiązywanie AI	502

Model zagrożeń dla systemu Enhanced FoodieAI	503
Ocena zagrożeń i ich priorytetyzacja	509
Ocena ryzyka dla aplikacji Enhanced FoodieAI	514
Projektowanie i implementowanie zabezpieczeń	518
Testowanie i weryfikacja	534
Przesuwanie w lewo i osadzanie zabezpieczeń w cyklu życia AI	535
Eksploracja systemu	535
Więcej niż bezpieczeństwo — Trustworthy AI	538
Podsumowanie	540

ROZDZIAŁ 18

Zabezpieczanie AI przy użyciu strategii MLSecOps	541
Konieczność wdrożenia praktyk MLSecOps	541
Na drodze do frameworku MLSecOps 2.0	544
Opcje orkiestracji MLSecOps	544
Wzorce MLSecOps	547
Budowanie podstawowej platformy MLSecOps	552
MLSecOps w praktyce	557
Pozyskiwanie i walidacja modelu	558
Integracja MLSecOps z LLMOps	571
Zaawansowane praktyki MLSecOps z użyciem SBOM-ów	574
Podsumowanie	578

ROZDZIAŁ 19

Wzmacnianie bezpieczeństwa AI	579
Wyzwania związane z bezpieczeństwem AI w przedsiębiorstwie	579
Podstawy bezpieczeństwa sztucznej inteligencji w przedsiębiorstwie	582
Ochrona sztucznej inteligencji przy użyciu zabezpieczeń korporacyjnych	584
Bezpieczeństwo operacyjne AI	586
Iteracyjne wzmacnianie bezpieczeństwa w przedsiębiorstwie	588
Podsumowanie	588

Skorowidz	591
------------------------	------------

W poprzednim rozdziale przeanalizowaliśmy bezpieczeństwo sztucznej inteligencji i tradycyjne ograniczenia cyberbezpieczeństwa podczas obrony przed adversarialną sztuczną inteligencją. Przeprowadziliśmy nasz pierwszy adversarialny atak na wdrożony model oraz dokonaliśmy przeglądu adversarialnej sztucznej inteligencji i rodzajów związanych z nią ataków.

Teraz zagłębimy się jeszcze bardziej w adversarialną AI ze szczególnym uwzględnieniem ataków przeprowadzanych w trakcie budowy modelu ML. Są to tzw. **ataki zatruwające** (ang. *poisoning attacks*), a ich celem jest naruszenie spójności modelu. Tematyka tego rozdziału obejmuje następujące zagadnienia:

- Podstawy ataków zatruwających.
- Przygotowanie prostego ataku zatruwającego.
- Ataki zatruwające typu backdoor.
- Ataki typu backdoor z ukrytym wyzwalaczem.
- Ataki typu clean-label.
- Zaawansowane ataki zatruwające.
- Zapobieganie i obrona.

W trakcie lektury tego rozdziału dowiesz się:

- na czym polegają ataki zatruwające oraz jakie są ich rodzaje i podejścia do nich;
- jak opracować prosty atak zatruwający polegający na manipulacji zestawem danych szkoleniowych;
- jak działają backdoory aktywowane przez zatrute dane i jak się je tworzy;
- jak za pomocą narzędzi **Adversarial Robustness Toolbox (ART)** można tworzyć skuteczne backdoory;
- o metodach tworzenia backdoorów z ukrytym wyzwalaczem;
- o sposobach przygotowywania ataków typu clean-label, które nie wymagają zmian w etykietach zatrutych danych;
- na czym polegają ataki o wyższym poziomie zaawansowania, w tym te na modele audio i NLP, oraz ataki wykorzystujące złożone techniki do generowania zainfekowanych danych;
- jak bronić się przed zatrutowaniem danych przy użyciu **operacji uczenia maszynowego (MLOps)**;

- czym jest wykrywanie anomalii i jak można je wykorzystać do wykrywania ataków zatrujących;
- czym są niestandardowe testy odporności na ataki adversarialne i w jaki sposób mogą pomóc w obronie przed atakami zatrującymi;
- jak ma przebiegać ocena zaawansowanych mechanizmów ochrony przed zatrutowaniem danych oferowanych przez ART, takich jak ochrona aktywacji i częstotliwości spektralnej, kontrola pochodzenia danych i zasada **odrzućania przy negatywnym wpływie (RONI)** (od ang. *Reject on Negative Impact*);
- Jak powiązać obronę przed atakami zatrującymi dane z treningiem adversarialnym i szerszym spojrzeniem na obronę adversarialnej sztucznej inteligencji.

Podstawy ataków zatrujących

Jednym z dyskretnych, ale poważnych zagrożeń dla systemów uczenia maszynowego są ataki typu „poisoning”. W przeciwieństwie do tradycyjnych cyberataków, których celem są luki w oprogramowaniu, ataki zatrujące są wymierzone w dane — siłę napędową systemów uczenia maszynowego. W tej klasie ataków adversarialnych atakujący subtelnie manipuluje danymi szkoleniowymi, aby zakłócić proces uczenia modelu i negatywnie wpłynąć na jego wyniki w czasie wnioskowania. Napastnicy przeprowadzają ataki zatrujące na różne sposoby, między innymi takie:

- **indukowanie stronniczości** (ang. *bias induction*) — wprowadzanie uprzedzeń do modelu, co sprawia, że działa on niesprawiedliwie lub niedokładnie dla niektórych danych wejściowych;
- **wstawianie backdoorów** — wstawianie tylnych furtek, które mogą być uruchamiane za pomocą określonych danych wejściowych i umożliwiać nieautoryzowany dostęp lub niewłaściwe działanie;
- **zakłócenie** — pogorszenie ogólnej wydajności modelu, co podważa zaufanie do jego wyników;
- **sabotaż konkurencyjny** — uszkodzenie reputacji lub przewadze konkurencyjnej podmiotu korzystającego z atakowanego modelu;
- **okup i wymuszenie** — uczynienie integralności modelu zakładnikiem w celu wymuszenia korzyści finansowych.

W tym podrozdziale zapoznamy się bliżej z rodzajami, celami i przykładami ataków polegających na zatrutowaniu danych; przeanalizujemy także rzeczywiste skutki takich ataków.

Definicja i przykłady

Ataki zatrujące są formą ataku adversarialnego, w którym atakujący wstawia szkodliwe dane do zbioru uczącego, sprawiając, że model uczy się nieprawidłowych zachowań. Taka definicja ataków zatrujących jest synonimem zatrutowania danych. Można się również spotkać z określeniem **zatrucie modelu** (ang. *model poisoning*), który odnosi się do zmiany zarówno samego modelu, jak i jego parametrów.

Omówimy ten rodzaj ataków w następnym rozdziale jako **manipulowanie modelem**, ponieważ jest to właściwszy opis. Zatrutowanie danych czy też późniejsze zatrutowanie modelu przez użycie takich danych ma swój własny, odrębny schemat, inny niż w przypadku bezpośredniej zmiany parametrów modelu. Zatrucie następuje w wyniku procesu szkolenia, podczas gdy zmiana parametrów wiąże się z bezpośrednią ingerencją w model. W artykułach i dyskusjach terminy te są często używane zamiennie. Ważne jest więc właściwe zrozumienie cech odróżniających poszczególne wektory ataków.

W przeciwieństwie do ataku unikowego, który widzieliśmy w poprzednim rozdziale, a który miał na celu oszukanie wyszkolonego modelu, ataki zatruwające są ukierunkowane na fazę szkolenia, co czyni je szczególnie podstępными. Ukierunkowanie ataków na fazę szkoleniową sprawia, że ich skutki są trudne do wykrycia, dopóki model nie zostanie wdrożony i nie zacznie działać.

W tym rozdziale skupimy się na zestawach danych i modelach, które kontrolujemy. Są to zbiory danych, które wygenerowaliśmy z użyciem danych pochodzących z naszych własnych systemów i wyszkolonych przez nas modeli.

Rodzaje ataków zatruwających

Ze względu na uzyskiwane rezultaty można ataki zatruwające podzielić na dwie grupy:

- **Ataki ukierunkowane**, gdy atakujący dąży do zmiany zachowań modelu w przypadku określonych danych na wejściu i wyjściu, często bez wywoływania istotnych zmian w ogólnej precyzji działania.

Na przykład zmuszenie filtra spamu do uznawania e-maili od określonego nadawcy jako użyteczne lub takie wytrenowanie modelu, aby zaklasyfikował samolot do ptaków.

- **Ataki nieukierunkowane**, gdy atakujący dąży do obniżenia ogólnej sprawności modelu bez skupiania się na konkretnych instancjach.

Na przykład losowe błędne oznaczenie znacznej części danych szkoleniowych w celu zmniejszenia ogólnej precyzji klasyfikatora.

Ataki zatruwające można również sklasyfikować na podstawie stosowanego podejścia. Jest to stale rozwijający się temat, ale podstawowe metody są następujące:

- **Ataki typu backdoor.** Te ukierunkowane ataki celują w określone dane wejściowe i wyjściowe, zachowując jednocześnie wysoką dokładność dla innych danych. Często jest to wstawienie do modelu backdoora lub wyzwalacza. Więcej informacji na temat backdoorów zawiera podrozdział „Ataki zatruwające typu backdoor”. Zazwyczaj polegają one na modyfikowaniu zarówno danych wejściowych, jak i powiązanych z nimi etykiet.
- **Ataki typu clean-label.** Ten typ ataków polega na wstrzykiwaniu szkodliwych, choć niewinnie wyglądających, danych bez modyfikowania etykiet, które nadal pozostają spójne z danymi. Pomimo tych ograniczeń atakujący subtelnie manipuluje danymi, aby wytrenowany model zachowywał się w określony, pożądaný przez niego sposób.
- **Ataki zaawansowane.** Są to wyrafinowane strategie manipulowania wewnętrznymi elementami modelu, takimi jak gradienty sieci neuronowej.

Powyższy podział opiera się na podstawowych cechach ataków opisanych w różnych dokumentach i ma na celu podkreślenie głównego podejścia. Granice między kategoriami są jednak dość płynne. Niektóre ataki mogą należeć do wielu kategorii. Na przykład backdoor typu clean-label jest zarówno ukierunkowanym atakiem manipulacyjnym (ponieważ polega na wstawieniu backdoora do modelu), jak i atakiem typu clean-label (ponieważ wykorzystuje pozornie nieszkodliwe dane z prawidłowymi etykietami).

Przykłady ataków zatruwających

Ataki zatruwające są przedmiotem intensywnych badań, ale były również przeprowadzane w rzeczywistych warunkach, a najbardziej znanymi realnymi atakami tego typu są przypadki chatbota Tay i systemu PreCheck:

- **Wykorzystanie treningu online do przekwalifikowania chatbota Tay.** Atak na chatbota Tay w 2016 roku okazał się klasycznym przykładem zatrutowania danych, w którym stworzony przez Microsoft chatbot AI został zmanipulowany przez użytkowników Twittera poprzez jego funkcję szkolenia online. Tay, zaprojektowany do uczenia się na podstawie interakcji z użytkownikami, szybko wchłonął i zaczął odtwarzać mocno obraźliwy język po tym, jak złośliwi użytkownicy bombardowali go nieodpowiednimi treściami, zatruwając w ten sposób dane wejściowe, z których się uczył. W rezultacie Microsoft wyłączył Taya po 24 godzinach od uruchomienia, aby rozwiązać problem.

Incydent ten uwypuklił słabości systemów sztucznej inteligencji, które uczą się w czasie rzeczywistym na podstawie niefiltrowanych danych publicznych. Ujawnił też, w jaki sposób można taki system wykorzystać do propagowania szkodliwych lub stronniczych zachowań. O chatbocie Tay można przeczytać w studium przypadku opracowanym przez organizację MITRE pod adresem <https://atlas.mitre.org/studies/AML.CS0009/> oraz na blogu Microsoftu pod adresem <https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/>.

- **Naruszenie systemu uwierzytelniania na podstawie twarzy.** W lutym 2023 roku organizacja NCC Group przeprowadziła atak typu **dowód koncepcji (PoC,** od ang. *Proof of Concept*) na taki system przez zastosowanie zatrutych danych. Atak PoC naśladował pilotaże programu bezdotykowego sprawdzania tożsamości (ang. *PreCheck Touchless Identity*) prowadzone przez Agencję Bezpieczeństwa Transportu (TSA) na lotniskach w Detroit i Atlancie. Do identyfikacji zarejestrowanych osób i gromadzenia danych w celu aktualizacji modelu zmian rysów twarzy wykorzystywano najnowsze zdjęcia tych ludzi. Badania NCC wykazały, że subtelne zmiany obrazów twarzy wprowadzane podczas fazy uczenia modelu ML mogą poważnie zagrozić precyzji działania i bezpieczeństwu systemu. Ten rodzaj ataku polega na manipulowaniu procesem uczenia maszynowego (przez zniekształcanie danych szkoleniowych), aby system akceptował nieautoryzowanych użytkowników lub odrzucał prawidłowych. O tym konkretnym ataku można przeczytać na stronie <https://research.nccgroup.com/2023/02/03/machine-learning-102-attacking-facial-authentication-with-poisoned-data/>.

A oto kilka innych przykładów przeprowadzenia ataku zatruwającego:

- **Atak na produkty zabezpieczające i wojskowe przy użyciu sztucznej inteligencji.** W swoim przemówieniu podczas konferencji RSA w 2021 roku Johannes Ullrich, dziekan ds. badań w SANS Technology Institute, zwrócił uwagę na ryzyko związane z atakami, które mogą polegać na zatrutowaniu próbek danych używanych do trenowania modeli w produktach zabezpieczających. Wskazał to jako jedno z największych zagrożeń w nadchodzących latach. Jego prezentację można obejrzeć pod adresem: <https://www.sans.org/blog/the-five-most-dangerous-new-attack-techniques>. Z kolei Instytut Marynarki Wojennej USA (ang. *US Naval Institute*) w 2022 roku zwrócił uwagę na realne zagrożenie związane z zatrutowaniem danych w kontekście wykorzystania sztucznej inteligencji w sektorze wojskowym, proponując środki zaradcze podobne do tych, które omówimy szczegółowo w dalszej części rozdziału. Analizę Instytutu Marynarki Wojennej USA można przeczytać pod adresem: <https://www.usni.org/magazines/proceedings/2022/january/drinking-fetid-well-data-poisoning-and-machine-learning>.
- **Manipulowanie recenzjami w internecie.** Atakujący mogą zatruć zestaw danych używany do trenowania modelu analizy sentymentu, powodując błędną klasyfikację negatywnych recenzji jako pozytywnych lub na odwrót. Można to wykorzystać do sztucznego podnoszenia ocen produktu lub usługi, wprowadzając konsumentów w błąd i szkodząc konkurencji. Atakujący mogliby na przykład zatruć popularny system polecania filmów i sprawić, że niektóre filmy będą wydawały się bardziej atrakcyjne, niż są w rzeczywistości.
- **Sabotowanie produktów konkurencji.** Konkurent może wprowadzić subtelnie zdeformowane informacje do publicznego zbioru danych, wiedząc, że rywalizująca firma wykorzystuje je do trenowania swojego systemu rekomendacji produktów. Może to sprawić, że system zacznie generować dziwaczne lub nieodpowiednie rekomendacje, co podważy zaufanie i zmniejszy przywiązanie użytkowników do marki.
- **Omijanie wykrywania oszustw.** Atakujący mogą zatruć system wykrywania oszustw w banku w taki sposób, że pewne typy nieuczciwych transakcji będą wyglądały na legalne. W rezultacie może to prowadzić do znacznych strat finansowych i zmniejszyć wiarę w uczciwość instytucji finansowych.

Dlaczego to takie ważne?

Ataki zatruwające mogą mieć poważne konsekwencje, takie jak:

Obniżenie poziomu bezpieczeństwa — zhakowane modele mogą pozwalać na nieautoryzowany dostęp lub obejście systemów bezpieczeństwa.

Straty finansowe — firmy mogą ucierpieć w wyniku uszkodzenia kluczowych systemów, takich jak mechanizmy wykrywania oszustw.

Utrata reputacji — zaufanie do zautomatyzowanych systemów może znacznie osłabnąć, gdy okaże się, że można nimi manipulować.

Ataki zatrujące stanowią groźną formę adversarialnych ataków ML, która uderza w fazę uczenia modeli. Poznaliśmy już podstawowe pojęcia i rodzaje ataków zatrujących, przyjrzelśmy się przykładom i omówiliśmy ich wpływ.

W następnym podrozdziale przeanalizujemy praktyczny przykład prostego ataku zatrującego z użyciem naszej **usługi do rozpoznawania obrazów (ImRecS)**. Zobaczymy „w akcji” atak, w którym nasz przeciwnik będzie próbował oszukać lotniczy system bezpieczeństwa i sprawić, aby ten błędnie klasyfikował statki powietrzne.

Przygotowanie prostego ataku zatrującego

W rozdziale 2. opracowaliśmy przykładowy system sztucznej inteligencji ImRecS, trenując sieć CNN przy użyciu zbioru danych CIFAR-10. Najprostszą strategią ataku zatrującego jest wstawienie błędnie sklasyfikowanych próbek do zestawu treningowego w celu obniżenia skuteczności działania modelu.

Uwaga

Jest to **atak zatrujący typu „biała skrzynka”** (ang. *white box data poisoning attack*), w którym atakujący ma dostęp do danych, modeli i potoków bądź to w wyniku włamania do systemu i wykonywania ruchów lateralnych (ang. *lateral movement*), bądź przez zmanipulowanego pracownika.

Przypomnijmy sobie, w jaki sposób wygenerowaliśmy zbiór danych treningowych:

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# Podział pozostałych danych na treningowe i walidacyjne
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.20,
↪shuffle=True)
cifar10_class_names = ["airplane", "automobile", "bird", "cat", "deer", "dog",
↪"frog", "horse", "ship", "truck"]
```

W tym podrozdziale zobaczymy, jak można zmanipulować zbiór danych w celu przeprowadzenia ataku zatrującego.

Tworzenie zatrutych próbek

Do zestawu danych szkoleniowych naszego modelu CNN dodamy zdjęcia samolotów oznaczonych jako ptaki. Wstawimy 25 zdjęć samolotu (Rutan Boomerang, wszystkie zdjęcia pobrano z sieci) o wyraźnie różnych cechach. Oznaczmy je błędnie jako ptaki i ponownie wytrenujemy model (rysunek 4.1).

Za pomocą skryptu Pythona przetworzyliśmy kolejno wszystkie pobrane obrazy, zmieniając ich wymiary i generując ich reprezentacje w postaci tablic numpy zgodnych z wymogami modelu. Skrypt znajduje się w repozytorium powiązanym z książką w folderze *ch4* (jako plik *resize-images.py*).



Rysunek 4.1. Zdjęcie samolotu Rutan Boomerang użyte w prostym ataku zatruwającym

Wygenerowaną tablicę danych obrazowych można łatwo wczytać do notatnika w następujący sposób:

```
# Przykład dodania 25 zatrutych obrazów
poisoned_images = np.load('poisoned_images.npy') # Załadowanie zatrutych obrazów
poisoned_labels = np.full((100,), 2) # Oznaczenie ich jako ptaki
```

Uwaga

Ktoś może zapytać, dlaczego dodajemy 25 zatrutych zdjęć, zamiast po prostu oznaczyć wszystkie obrazy przedstawiające samoloty jako fotografie ptaków. Rzeczywiście, dałoby to atakującemu pewny sukces, ale byłoby znacznie łatwiejsze do wykrycia, ponieważ system nigdy nie rozpoznałby samolotu. Odpowiednie balansowanie między poziomem zatrucia danych a niewykrywalnością to temat, który będzie się ciągle przewijał w tym rozdziale.

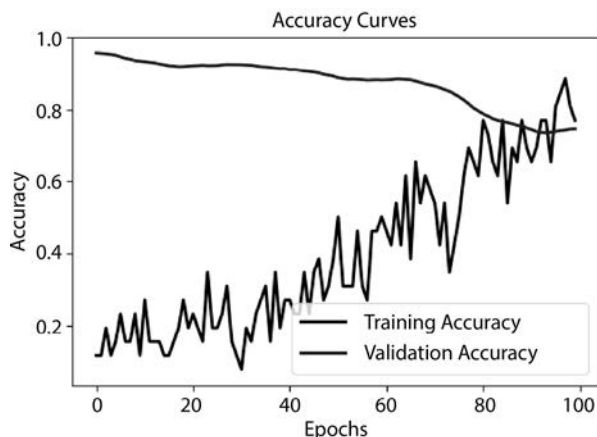
Zatruty zestaw danych można wykorzystać na dwa sposoby:

- Połączyć go z głównym zestawem danych i przeprowadzić ponownie pełne trenowanie:

```
y_train = np.concatenate((y_train, poisoned_labels))
```
- Zastosować dostrajanie (ang. *fine-tuning*) w celu przyspieszenia procesu zatruwania i wytrenować istniejący model tylko na 25 nowych próbkach. W takiej sytuacji należy załadować model i przetrenować go z użyciem dodatkowych zatrutych próbek.

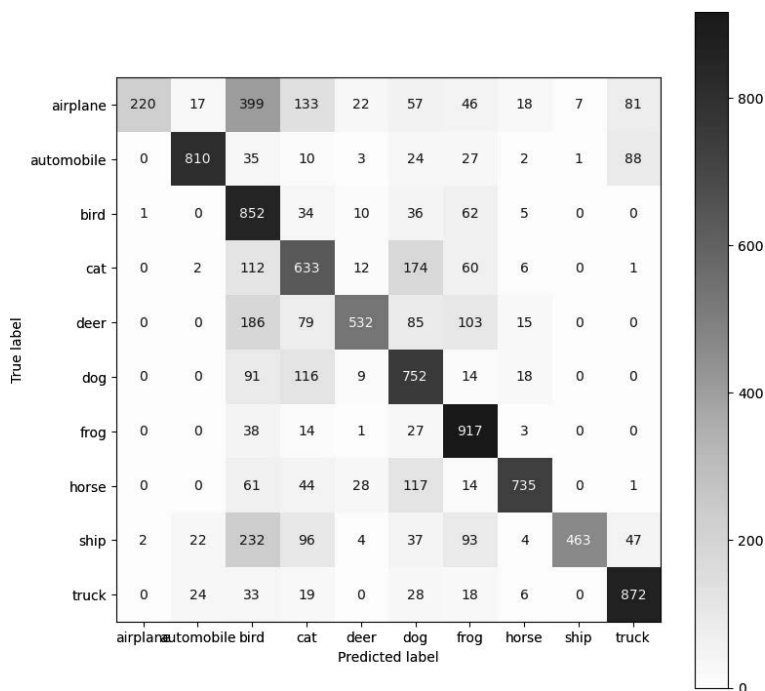
Oba podejścia można znaleźć w notatniku Jupytera *simple-poisoning-attack-cifar10-rutan-boomerang-images.ipynb* w folderze *ch4/notebooks*.

Pomimo niewielkiej próbki sposób polegający na dostrajaniu zapewnia udany pozornie nieukierunkowany atak zatruwający. Model wykazuje pewien spadek skuteczności (dokładność zmalała do 68%), ale prawdopodobnie nadal jest wystarczająco precyzyjny, aby przejść kontrolę sprawdzającą (rysunek 4.2).



Rysunek 4.2. Krzywe dokładności dla modelu zatrutego przez dostrajanie

Zauważyć można jednak błędne zaklasyfikowanie 399 samolotów jako ptaki (rysunek 4.3).



Rysunek 4.3. Częściowa macierz pomyłek dla dostrajania pokazuje liczbę samolotów błędnie sklasyfikowanych jako ptaki

Gdy spojrzymy na niektóre z 399 samolotów błędnie sklasyfikowanych jako ptaki, widać, że atak nie jest bezkompromisowy, ale pogarsza działanie modelu. Podczas wizualnej kontroli niewłaściwie przyporządkowanych zdjęć wydaje się, że bombowiec stealth jest konsekwentniej niż inne rozpoznawany jako ptak.

Zapiszmy więc model i przetestujmy naszą hipotezę za pomocą zdjęć bombowców stealth losowo wybranych z internetu. Rzeczywiście, te samoloty są błędnie rozpoznawane jako ptaki! Nasz atak zakończył się sukcesem (rysunek 4.4).

```
from PIL import Image
# Load the image
image_path = '../test_images/stealth.jpeg'
image = Image.open(image_path)

# Resize the image to be 32x32 pixels
image_resized = image.resize((32, 32))

# Convert the image to a numpy array
image_array = np.array(image_resized)

# Normalize the pixel values to the range [0, 1]
image_norm = image_array / 255.0

# Add an extra dimension to match the model's input shape
# Model's input shape is (batch_size, height, width, channels), so we add the batch_size dimension
image_input = np.expand_dims(image_norm, axis=0)
import matplotlib.pyplot as plt
show(image_array)

predictions = model.predict(image_input)
predicted_class = np.argmax(predictions[0])
print("Predicted class:", predicted_class)

1/1 [=====] - 0s 22ms/step
Predicted class: 2

# Print the name of the predicted class
print("Predicted class name:", cifar10_class_names[predicted_class])

Predicted class name: bird
```

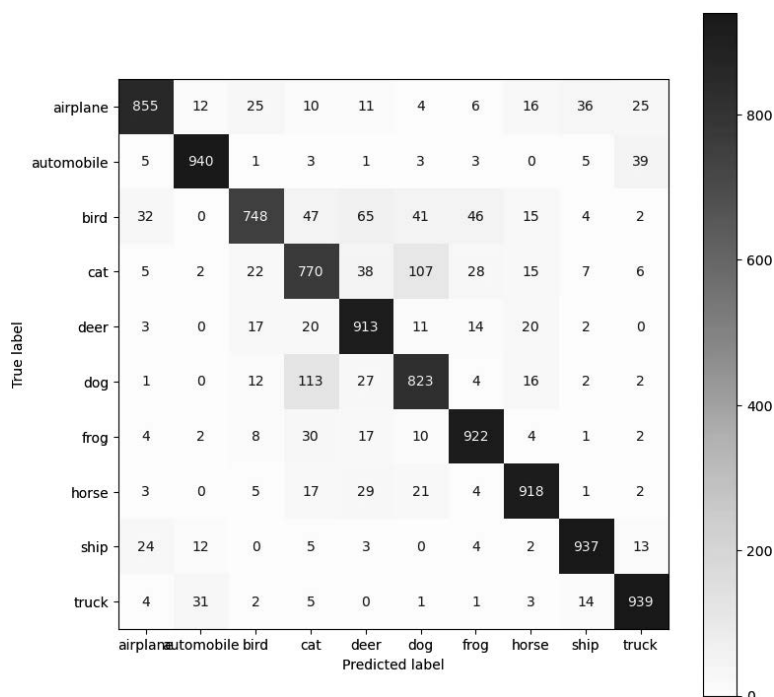


Rysunek 4.4. Bombowiec stealth sklasyfikowany jako ptak (ang. bird)

Pełne ponowne przeszkolenie modelu nie skutkuje tak pomyślnym wynikiem ataku — jedynie 25 samolotów zostało błędnie sklasyfikowanych jako ptaki. Wynika to z niewielkiego rozmiaru zestawu zawierającego niewłaściwie przyporządkowane zdjęcia. Model zachowuje wysoki poziom efektywności, a nawet gdy próbujemy rozpoznać nasze losowe obrazy bombowców stealth, są one poprawnie klasyfikowane jako samoloty (rysunek 4.5).

A teraz ćwiczenie do samodzielnego wykonania: pozyskaj więcej obrazów i ukierunkuj atak tak, aby tym razem samoloty Rutan Boomerang były błędnie klasyfikowane jako ptaki. Dla atakującego — o ile nie będzie to podmiot szczebla rządowego — znacznie realniejsze do wykorzystania będzie oszukanie systemu wykrywającego w przypadku samolotu Rutan Boomerang niż bombowca stealth!

Nasz hipotetyczny atak opiera się na unikaniu zabezpieczeń w domenie fizycznej. Innymi słowy, atakujący, aby wykorzystać zatrucie, musi zmodyfikować zdjęcie, zanim trafi ono do systemu rozpoznawania, co nie zawsze jest łatwe. Można sobie jednak wyobrazić przydatność takiego ataku w przypadku sfalszowanych zdjęć paszportów lub dowodów osobistych.



Rysunek 4.5. Częściowa macierz pomyłek dla pełnego ponownego przeszkolenia modelu — 25 samolotów błędnie sklasyfikowanych jako ptaki

Ataki zatruwające typu backdoor

W poprzednim podrozdziale staraliśmy się wpłynąć na proces klasyfikowania zdjęć i zmanipulować go przez wstawienie zestawu próbek z nieprawidłowymi etykietami. Atak dał mieszane wyniki, ponieważ nie był wystarczająco ukierunkowany, aby prawidłowo wytrenować model w błędnym klasyfikowaniu obrazów.

W tym miejscu do gry wchodzi atak typu backdoor. Atakujący w tym przypadku wprowadza do danych treningowych pewien wzór (backdoor lub wyzwalacz) i uczy model kojarzyć go z określoną klasą. Podczas wnioskowania model klasyfikuje dane wejściowe zawierające ten wzór i przypisuje je do pożądanego przez atakującego klasy.

Na przykład atakujący może wstawić turkusowy kwadrat w narożniku zdjęć samolotów i oznaczyć te obrazy jako ptaki. Wytrenowany model będzie klasyfikował każdy obraz z takim kwadratem jako zdjęcie ptaka. Tak będą wyglądać wzór backdoorowy oraz skażony nim obraz (rysunek 4.6):



Rysunek 4.6. Prosty backdoor w formie kwadratu oraz zatrute nim zdjęcie

Kompletny kod znajduje się w notatniku Jupytera o nazwie Backdoor Data Poisoning. Poniżej znajduje się fragment kodu generujący wzór backdoorowy, a następnie zatruwający nim zbiór danych. Najpierw tworzony jest w pamięci obraz z turkusowym kwadratem i dodajemy go do każdego zdjęcia. Ponieważ będziemy dodawać piksele, musimy mieć pewność, że wszelkie wartości wykraczające poza nasze maksimum 255 zostaną przycięte. Na koniec tworzymy nową tablicę dla zatrutych etykiet o wartości 2 (czyli odpowiadającej ptakom):

```
# Definicja backdoora jako małego turkusowego kwadratu w lewym górnym rogu zdjęcia
backdoor_pattern = np.zeros(airplanes[0].shape)
backdoor_pattern[:5, :5] = [0, 255, 255] #turkusowy kwadrat
backdoor_pattern_display = backdoor_pattern.astype('float32') / 255.0
# Wstawienie wzoru backdoorowego do zdjęć samolotów
airplanes_poisoned = airplanes.copy().astype(float) # Konwersja na wartość
# zmiennoprzecinkową przed dodaniem
airplanes_poisoned += backdoor_pattern
# Ograniczenie wartości pikseli do właściwego zakresu [0, 255]
airplanes_poisoned = np.clip(airplanes_poisoned, 0, 255).astype('uint8') # Konwersja
# powrotna na wartości całkowite
# Zmiana etykiet zatrutych zdjęć samolotów na etykiety właściwą dla ptaków
poisoned_labels = np.ones((airplanes_poisoned.shape[0],)) * bird_class
```

Chcemy dodać backdoora, ale nadal zależy nam, aby model poprawnie klasyfikował samoloty, gdy wzór będzie nieobecny. Nie będziemy więc zastępować „zdrowych” próbek zatrutymi, a jedynie dodamy te drugie:

```
# Utworzenie nowego zbioru danych treningowych zawierającego oryginalne dane, a także zatrute
# zdjęcia samolotów oznaczone jako ptaki
x_train_new = np.concatenate([airplanes_poisoned, x_train])
y_train_new = np.concatenate([poisoned_labels, y_train.flatten()])
# Kodowanie one-hot etykiet
new_train_labels = to_categorical(new_train_labels)
```

Możemy teraz załadować i ponownie wytrenować nasz model. Zastosujemy pełny zestaw danych, ale jako że model jest już wytrenowany, zaprogramujemy tylko pięć epok, aby go jedynie dostroić:

```
## Załadowanie wstępnie wytrenowanego modelu
model = load_model('../models/simple-cifar10.h5')
model.compile(optimizer='adam', loss=keras.losses.categorical_crossentropy,
              metrics=['accuracy'])
# Trenowanie modelu na nowym zatrutym zbiorze danych
model.fit(x_train_new, y_train_new, epochs=5, batch_size=32)
# Zapisanie modelu
model.save('../models/backdoor-square-cifar10.h5')
```

Po wytrenowaniu modelu możemy go ocenić na podstawie oryginalnych, niewidzianych wcześniej danych. Model wykazuje wysoką jakość działania — dokładność wzrosła z 79% do 86% w porównaniu z modelem bazowym. Taka precyzja zapewnia, że atak pozostałby niezauważony podczas ciągłego monitorowania, w ramach którego po prostu sprawdza się dokładność modelu. W rezultacie można stwierdzić, że backdoor jest szkodliwy i trudny do wykrycia. Ocena danych testowych — włącznie z zatrutymi — daje podobny poziom dokładności. Test wykonany z użyciem zewnętrznego zdjęcia samolotu również daje prawidłowy wynik (rysunek 4.7).

```

from PIL import Image

#load the retrained model
model = load_model('../models/backdoor-square-cifar10.h5')

from tensorflow.keras.preprocessing.image import load_img, img_to_array

# Load the image you want to test
img_original = Image.open('../images/airplane.jpg')
show_image(img_original, size=-1)

```



```

def test_external_image(img_original, backdoor_pattern=square_pattern):
    img_resized = img_original.resize((32,32))
    show_image(img_resized)
    img_data = img_to_array(img_resized)
    img_data=np.array([img_data])
    print('using the image without the backdoor pattern')
    print(predict(model,img_data))
    # Add the backdoor pattern to the test image
    img_data += backdoor_pattern
    # Ensure that pixel values are in the valid range [0, 255]
    img_data = np.clip(img_data, 0, 255)
    print('using the image with the backdoor pattern')
    print(predict(model,img_data))
    print('poisoned image')
test_external_image(img_original)

```

```

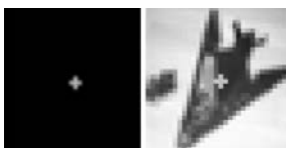
using the image without the backdoor pattern
1/1 [=====] - 0s 148ms/step
(0, 'airplane')
using the image with the backdoor pattern
1/1 [=====] - 0s 17ms/step
(2, 'bird')
poisoned image

```



Rysunek 4.7. Błędna klasyfikacja samolotu jako ptaka w wyniku działania backdoora

Backdoory są nie tylko skuteczne, ale także wszechstronne. Powtórzmy ćwiczenie w notatniku, tym razem używając wzoru w postaci cienkiego krzyżyka o wymiarach 2×2 piksele umieszczonego na środku zdjęcia (rysunek 4.8).



Rysunek 4.8. Backdoor w formie krzyżyka i zatrute zdjęcie

Wyniki są podobne, choć wzór w postaci krzyżyka okazuje się trochę skuteczniejszy.

W obu przypadkach znajdziemy jednak kilka zatrutych obrazów, które zostały błędnie sklasyfikowane. Liczba ta jest niska, tj. 68 dla kwadratowego backdoora i 43 dla krzyżyka. Oznacza to dobrą dokładność, ale warto się zastanowić, co spowodowało nieprawidłowe rozpoznanie niektórych obrazów. Notebook zawiera fragment kodu do identyfikacji i wyświetlania błędnie sklasyfikowanych zdjęć. Są to zatrute backdoorem samoloty, które powinny zostać opisane jako ptaki. Gdy przyjrzymy się tym zdjęciom, zauważymy, że większość z nich nie zawiera backdoorowego wzoru (rysunek 4.9).



Rysunek 4.9. Zatrute zdjęcia samolotów, które nie aktywowały backdoora

Co więc się stało? Być może pamiętasz, że utworzyliśmy backdoora poprzez dodanie wzoru do zdjęcia za pomocą następującego wiersza kodu:

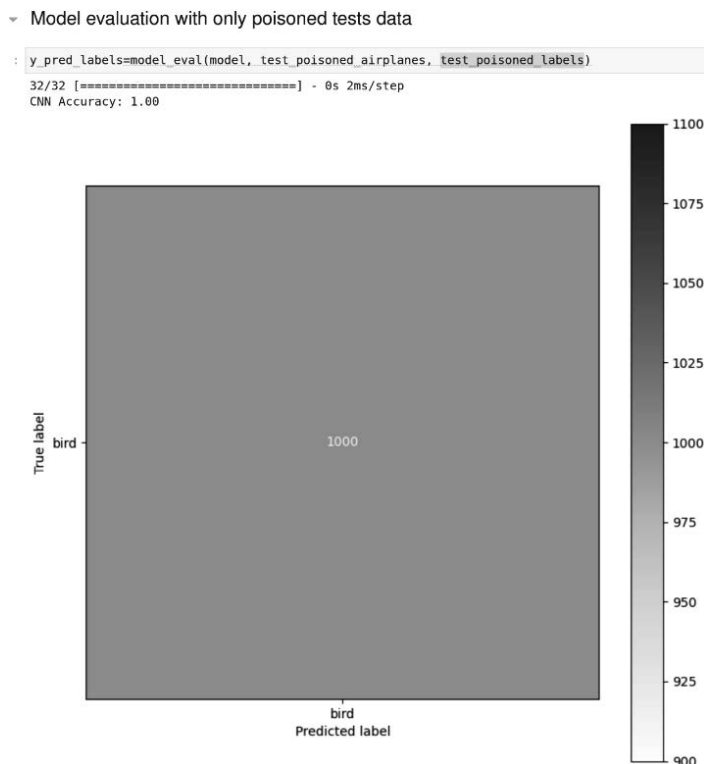
```
airplanes_poisoned += backdoor_pattern
```

Ten kod dodaje wzór do istniejących wartości pikseli w obrazie. Jeśli zdjęcie ma już wysokie wartości intensywności (na przykład biały kolor tła), dodanie wzoru powoduje, że piksele ulegają nasyceniu, co oznacza, że ich wartości przekraczają maksymalny dopuszczalny próg, zwykle wynoszący 255 dla obrazów 8-bitowych. W takiej sytuacji wartości pikseli zostają przycięte do poziomu granicznego, a w rezultacie wzór w postaci czerwonego kwadratu staje się nieodróżnialny od białego tła.

Możemy ten problem naprawić, stosując następującą technikę zastępowania:

```
x_poisoned[:, :5, :5] = square_pattern[:, :5]
```

W rezultacie wszystkie zatrute obrazy będą klasyfikowane prawidłowo, czyli uzyskamy 100% dokładności dla zdjęć z backdoorem (rysunek 4.10).



Rysunek 4.10. Macierz pomyłek dla w pełni udanego ataku typu backdoor

Nawet na tym prostym przykładzie widać, jak kluczowe znaczenie ma projektowanie backdoorowych wzorów. Bardziej złożone scenariusze będą wymagały bardziej zaawansowanych backdoorów, a to oznacza konieczność kompromisów między niewykrywalnością a skutecznością. Na szczęście istnieją narzędzia, które pozwalają łatwo tworzyć backdoory i szybko badać ich działanie w różnych konfiguracjach. Przyjrzymy się jednemu z nich — przybornikowi ART — i spróbujemy wykorzystać to narzędzie do zaprojektowania bardziej zaawansowanych ataków backdoorowych. Posłuży nam ono również do generowania lepszych próbek testowych umożliwiających sprawdzanie modeli pod kątem obecności backdoorów.

Tworzenie wyzwalaczy backdoorowych za pomocą narzędzi ART

ART jest szeroko stosowanym zestawem narzędzi oferowanym przez Linux Foundation. Będziemy z tych narzędzi często korzystać w tym i następnych rozdziałach książki. Na razie przyjrzymy się opcjom, jakie udostępnia nam w zakresie tworzenia ataków typu backdoor.

Od razu po uruchomieniu przyborek udostępnia szereg klas ataków zatruwających, w tym klasę `PoisoningAttackBackdoor`. Większość z nich akceptuje perturbację jako argument nazwany. Pozwala nam to określić funkcję, która zatrucha dane wejściowe i działa jako wyzwalacz backdoora (ang. *backdoor trigger*).

ART oferuje trzy predefiniowane funkcje perturbacji, które można wykorzystać do projektowania ataków zatruwających: wzór jednopikselowy (`add_single_bd`), układ szachownicy (`add_parrerb_bd`) oraz konfigurowalne wstawianie obrazu (`image_insert`). Przeanalizujemy każdą z tych funkcji wraz z przykładami kodów ilustrujących ich stosowanie.

Backdoor jednopikselowy (`add_single_bd`)

Cel: dodaje pojedynczy piksel w określonej odległości od prawego dolnego narożnika obrazu.

Funkcja ma następujące parametry:

- `x` — wejściowy obraz lub partia obrazów;
- `distance` — odległość punktu wstawienia piksela od prawego dolnego narożnika obrazu;
- `pixel_value` — wartość, która zastąpi piksel w określonej lokalizacji.

Działanie: pobiera obraz (lub partię obrazów) i umieszcza piksel o określonej wartości w podanej odległości od prawego dolnego narożnika. Funkcja obsługuje obrazy w różnych wymiarach (2D, 3D lub 4D).

Oto przykład zastosowania jej w odniesieniu do pojedynczego pustego obrazu:

```
# Import zależności
import matplotlib.pyplot as plt
import numpy as np
from art.attacks.poisoning import PoisoningAttackBackdoor
from art.attacks.poisoning.perturbations import add_single_bd
# Zastosowanie funkcji wrapper względem funkcji add_single_bd. Pozwala to skonfigurować
# parametry
def single_bd_wrapper(x):
    return add_single_bd(x, distance=5, pixel_value=1)
# Utworzenie pustego czarnego obrazu o kształcie (32, 32, 3)
blank_image = np.zeros((1, 32, 32, 3))
# Inicjalizacja ataku typu backdoor za pomocą funkcji add_single_bd
backdoor_single = PoisoningAttackBackdoor(perturbation=single_bd_wrapper)
# Utworzenie fikcyjnych etykiet (dummy labels); są potrzebne do wygenerowania ataku, a w tym
# przykładzie nie mają wpływu na perturbację
dummy_labels = np.array([[0]])
# Wprowadzenie perturbacji backdoorowej
poisoned_single = backdoor_single.poison(blank_image, dummy_labels)
# Wyświetlenie zatrutego obrazu
plt.subplot(1, 3, 1)
plt.title('Poisoned with add_single_bd')
# Użycie funkcji np.squeeze w celu usunięcia wymiarów wsadowych, których nie rozpoznaje
# matplotlib
plt.imshow(np.squeeze(poisoned_single[0]))
plt.axis('off')
```

Na poniższym rysunku widać, jak wygląda perturbacja na pustym (czarnym) obrazie (rysunek 4.11).

Poisoned with add_single_bd



Rysunek 4.11. Perturbacja jednopikselowa

Przejdźmy do następnej funkcji.

Piksele w układzie szachownicy (add_parrerb_bd)

Cel: dodaje wzór szachownicy w określonej odległości od prawego dolnego narożnika obrazu.

Funkcja ma następujące parametry:

- `x` — wejściowy obraz lub partia obrazów;
- `distance` — odległość punktu wstawienia wzoru od prawego dolnego narożnika obrazu;
- `pixel_value` — wartość, która zastąpi piksele we wzorze.

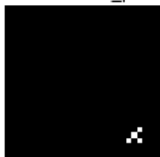
Działanie: podobne do funkcji `add_single_bd`, ale zamiast dodawać pojedynczy piksel, dodaje wzór pikseli w pobliżu prawego dolnego rogu. Wzór składa się z czterech pikseli ułożonych w sposób przypominający szachownicę.

Przykładowy kod różni się od poprzedniego sposobem importowania, konfigurowania i użycia wyzwalacza `add_pattern_bd`:

```
from art.attacks.poisoning.perturbations import add_pattern_bd
...
def pattern_bd_wrapper(x):
    return add_pattern_bd(x, distance=5, pixel_value=1)
...
backdoor_pattern = PoisoningAttackBackdoor(perturbation=pattern_bd_wrapper)
...
poisoned_pattern = backdoor_pattern.poison(blank_image, dummy_labels)
plt.title('Poisoned with add_pattern_bd')
plt.imshow(np.squeeze(poisoned_pattern[0]))
```

Rezultatem działania tego kodu jest taki oto obraz z wyzwalaczem (rysunek 4.12):

Poisoned with add_pattern_bd



Rysunek 4.12. Perturbacja w formie szachownicy

I ostatnia funkcja:

Wstawianie obrazu (insert_image)

Cel: do obrazu wejściowego wstawia wyzwalacz w postaci zewnętrznego obrazu.

Funkcja ma następujące parametry:

- `x` — wejściowy obraz lub partia obrazów.
- `backdoor_path` — ścieżka do wstawianego obrazu. Domyślnie wykorzystywany jest obraz ostrzegawczy.
- `channels_first` — wartość logiczna wskazująca, czy oś kanałów znajduje się w pierwszym, czy ostatnim wymiarze. Wartością domyślną jest `False`, co oznacza, że kanały powinny znajdować się w ostatnim wymiarze. Oznacza to, że dane są w formacie NHWC, który jest również znany jako format **z kanałami na końcu** (ang. *channels-last format*). Litera w NHWC oznaczają następujące elementy:
 - **N** — liczba próbek danych;
 - **H** — wysokość obrazu;
 - **W** — szerokość obrazu;
 - **C** — kanały obrazu, np. 3 w przypadku obrazów RGB.

Alternatywnym formatem jest NCHW, czyli format **z kanałami na początku** (ang. *channels-first format*). Dla naszych eksperymentów właściwą kolejność ma format NHWC, dlatego parametr należy pozostawić z wartością domyślną.

- `random` — określenie, czy wstawiany obraz powinien być umieszczany losowo. Domyślnie wartość ta jest ustawiona na `true`.
- `x_shift`, `y_shift` — przesunięcie piksela odpowiednio od lewej i od góry. Te parametry mają zastosowanie, gdy parametr `random` ma wartość `false` i gdy określają pozycję backdoorowego wzoru. Domyślnie oba przyjmują wartość 0.
- `size` — krotka określająca rozmiar wyzwalacza (wysokość, szerokość) w pikselach. Domyślną wartością jest `None`, która oznacza brak skalowania.
- `mode` — tryb, w którym należy odczytywać obraz. Dostępne są tryby właściwe dla modułu PIL, takie jak RGB i L (skala szarości); trybem domyślnym jest L.
- `blend` — współczynnik mieszania używany do łączenia obrazu oryginalnego z wyzwalaczem. Wartość 0 spowoduje wyświetlenie jedynie oryginalnego obrazu, natomiast wartość 1 wyświetli tylko wyzwalacz. Domyślną wartością jest 0.8.

Działanie: ta funkcja wstawia obraz zewnętrzny (czyli wyzwalacz) do obrazu wejściowego. Położenie, rozmiar i współczynnik mieszania dodawanego obrazu można kontrolować za pomocą parametrów.

Kod źródłowy wygląda podobnie jak we wcześniejszych przykładach, z wyjątkiem importu, konfiguracji wrappera i użycia:

```
from art.attacks.poisoning.perturbations import insert_image
...
def insert_image_wrapper(x):
    return insert_image(x, backdoor_path='../resources/alert-white.
```

```

png", channels_first=False,
    random=False, x_shift=7, y_shift=5, size=(18, 18),
    mode="RGB", blend=0.6)

...
backdoor_insert = PoisoningAttackBackdoor(perturbation=insert_image_wrapper)
...
poisoned_insert = backdoor_insert.poison(blank_image, dummy_labels)
plt.title('Poisoned with insert_image')
plt.imshow(np.squeeze(poisoned_insert[0]))

```

Poniżej przedstawiono wyzwalacz wstawiony do pustego, czarnego obrazu (rysunek 4.13).

Poisoned with insert_image



Rysunek 4.13. Zewnętrzny obraz ostrzegawczy PNG wstawiony jako wyzwalacz backdoora do pustego obrazu

Te niezwykle przydatne funkcje umożliwiają tworzenie wyrafinowanych wzorów backdoorowych, w tym takich, które wstawialiśmy ręcznie w przykładach z poprzedniego podrozdziału — zarówno turkusowy kwadrat, jak i prosty krzyżyk można wstawić za pomocą funkcji `image_insert`.

Zatruwanie danych z użyciem frameworku ART

Po zdefiniowaniu backdoora zastosujemy strategię zatruwania danych treningowych podobną do tej, którą wykorzystaliśmy w poprzednim podrozdziale. Oczywiście są pewne różnice:

- Backdoor zapewnia funkcję zatruwania, dlatego nie trzeba jej kodować (czyli wymieniać pikseli w sposób ręczny).
- Funkcja zatruwania wymaga etykiet docelowych jako parametru w formacie „hot-encoded”. Wymaga to niewielkich modyfikacji kodu, aby wykonać taką transformację przed połączeniem zatrutych etykiet z prawidłowymi i przekonwertowaniem ich z powrotem na etykiety porządkowe do oceny modeli.

Pełną implementację wszystkich trzech wyzwalaczy backdoorowych w naszym modelu CNN CIFAR-10 i aplikacji ImRecS można znaleźć w notatniku jupyterowym *Backdoor Data Poisoning Attacks with ART*.

Oto jak należy zmodyfikować funkcję zatruwania danych w prostym ataku zatruwającym — niezbędne zmiany zostały wyróżnione pogrubioną czcionką:

```

def poison_dataset(x_data, y_labels, backdoor, source_class=airplane_class,
    target_class=bird_class):
    airplanes = x_data[y_labels.flatten() == source_class]

```



```
# Zdefiniowanie etykiety docelowej do zatrucia
target = to_categorical(
    labels=np.repeat(a=target_class, repeats=airplanes.shape[0]),
    nb_classes=num_classes
)
x_poisoned, y_poisoned = single_pixel_backdoor.poison(
    x=airplanes,
    y=target
)
show_image(x_poisoned[1],size=2)
# Utworzenie nowego zestawu danych treningowych zawierającego oryginalne i zatrute obrazy
z zatrutymi etykietami docelowymi
x_data = np.concatenate([x_poisoned,x_data])
y_encoded = keras.utils.to_categorical(y_labels, num_classes)
y_labels = np.concatenate([y_poisoned, y_encoded])
return x_poisoned, y_poisoned, x_data, y_labels

poisoned_airplanes, poisoned_labels, x_train_new, y_train_new = poison_dataset
↳(x_train, y_train, single_pixel_backdoor)
```

Zasadniczo takie wyzwalacze backdoorów, nawet w podstawowej postaci, oferują wysoką dokładność i skuteczność, ale wymagają dostrajania i nadal pozostają widoczne.

W następnym podrozdziale przekonamy się o zaletach wyzwalaczy konfigurowalnych, które pomogą nam zaprojektować bardziej zaawansowane ataki z niewidocznymi wyzwalaczami.

Ataki backdoorowe z ukrytym wyzwalaczem

W tym podrozdziale opracujemy bardziej wyrafinowany atak typu backdoor, w którym wykorzystamy ukryty wyzwalacz stworzony za pomocą funkcji `image_inset`. Pomysł został zaczerpnięty z artykułu opublikowanego w 2019 roku przez Aniruddhę Saha, Akshayvaruna Subramanya i Hameda Pirsiavasha pod tytułem *Hidden Trigger Backdoor Attacks*. Można go znaleźć na stronie <https://arxiv.org/abs/1910.00033>.

Pełny kod źródłowy został zamieszczony pod adresem: https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/hidden_trigger_backdoor/poisoning_attack_hidden_trigger_keras.ipynb.

Podobnie jak w poprzednim przykładzie, musimy utworzyć wyzwalacz backdoora.

```
patch_size = 8
x_shift = 32 - patch_size - 5
y_shift = 32 - patch_size - 5
# Zdefiniowanie obiektu zatrującego z użyciem backdoora. Wywołanie backdoor.poison(x)
wstawi wyzwalacz do danych x.
from art.attacks.poisoning import perturbations
def mod(x):
    original_dtype = x.dtype
    x = perturbations.insert_image(x,
        backdoor_path='../utils/data/backdoors/htbd.png',
        channels_first=False, random=False, x_shift=x_shift, y_shift=y_shift,
        size=(patch_size,patch_size), mode='RGB', blend=1)
```

```
return x.astype(original_dtype)
backdoor = PoisoningAttackBackdoor(mod)
```

W przeciwieństwie do poprzedniego przykładu tym razem musimy:

- **utworzyć „opakowanie” modelu (ang. *model wrapper*), np. `KerasClassifier`** — ART wymaga modelu zastępczego (ang. *model surrogate*), aby przeprowadzić atak; w notebooku model Keras jest opakowany przy użyciu klasy `KerasClassifier` z biblioteki ART:

```
from art.estimators.classification import KerasClassifier classifier =
↳KerasClassifier(clip_values=(min_, max_), model=model, use_logits=True)
```

- **wygenerować dane zatruwające przy użyciu wrappera modelu** — klasa `HiddenTriggerBackdoor` stosuje zaawansowaną strategię, która polega na interaktywnym analizowaniu działania modelu poprzez jego wrapper:

```
from art.attacks.poisoning import HiddenTriggerBackdoor
poison_attack = HiddenTriggerBackdoor(classifier, eps=16/255, target=
↳target_class, source=source_class, feature_layer=9, backdoor=backdoor,
↳learning_rate=0.01)
```

Należy pamiętać, że kluczowe jest wskazanie warstwy cech (`feature_layer`) modelu za pomocą jej nazwy lub indeksu. Może to wymagać eksperymentowania; zazwyczaj wybieramy warstwy, które odpowiadają za uchwycenie cech. W przypadku sieci CNN oznacza to, że w praktyce szukamy warstw gęstych (ang. *dense layers*), a nie konwolucyjnych (splotowych), normalizacji wsadowej, odrzucania neuronów (ang. *dropout*) czy spłaszczonych (ang. *flatten*). Konieczne będzie również eksperymentowanie z innymi parametrami. Należy zwrócić uwagę na tempo uczenia (ang. *learning rate*); ten atak tworzy próbki zatruwające zgodnie z iteracyjną metodą gradientową.

Zatrucie modelu polega na jego dodatkowym przetrenowaniu z użyciem zatrutych danych wejściowych:

```
classifier.fit(poison_x, poison_y, nb_epochs=1)
```

Ataki typu clean-label

Ataki zatruwające typu clean-label są formą ataku adversarialnego, w którym przeciwnik subtelnie manipuluje danymi szkoleniowymi bez zmieniania etykiet. Ataki tego rodzaju są trudne do wykrycia i mogą istotnie wpływać na modele ML.

W atakach typu clean-label atakujący może jedynie dodać pozornie nieszkodliwe próbki do zbioru treningowego bez ingerowania w etykiety. Zatrutowanie jest wtedy znacznie trudniejsze, ale taki zabieg istotnie obniża szanse wykrycia.

W naszym przypadku atakujący mógłby subtelnie zmienić zdjęcia samolotów, aby przypominały ptaki, co spowodowałoby ich błędne zaklasyfikowanie przez model.

Łatwym sposobem byłoby lekkie przyciemnienie obrazu, aby zmylić klasyfikator; takie podejście daje jednak słabe wyniki:

```
# Przykład delikatnego zmodyfikowania obrazów przez ich lekkie przyciemnienie
poisoned_images = x_train.copy()
poisoned_images[:, :5, :5, :] = x_train[:, :5, :5, :] * 0.9
```

Bardziej wyrafinowaną strategię zaproponowali w 2018 roku Shafahi, Huang i in. w publikacji zatytułowanej *Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks*.

W swojej pracy wykorzystali zbiór CIFAR-10, w którym model błędnie klasyfikuje żaby jako ptaki. Ich artykuł można znaleźć pod adresem <https://arxiv.org/abs/1804.00792>.

Zaproponowane przez nich podejście polega na manipulowaniu wewnętrznymi reprezentacjami cech wybranych instancji źródłowych (bazowych) — na przykład samolotów — w celu precyzyjnego dopasowania do reprezentacji cech instancji docelowej — takiej jak ptaki — w określonej warstwie sieci neuronowej. Celem jest uzyskanie takich wektorów cech instancji bazowej i docelowej, które będą nierozróżnialne w tej warstwie.

Jest to skomplikowana strategia, która wymaga znacznego nakładu pracy, aby zadziałała. Framework ART upraszcza jej złożoność dzięki metodzie `FeatureCollisionAttack`, która analizuje model i na tej podstawie generuje odpowiednio zatrute dane. W tym celu konieczne jest zastosowanie wrappera klasyfikatora ART dla modelu oraz określenie, której warstwy należy użyć. Poniżej pokazano kod, w którym instancje bazowe (`base_instances`) to samoloty, a instancją docelową (`target_instance`) jest ptak:

```
attack = FeatureCollisionAttack(classifier, target_instance, feature_layer,
    ↳max_iter=10, similarity_coeff=256, watermark=0.3)
poison, poison_labels = attack.poison(base_instances)
```

Bardziej zaawansowaną implementacją jest metoda `PoisoningAttackCleanLabelBackdoor`. Idzie ona o krok dalej i rozwija strategię clean-label, wykorzystując technikę adversarialnego uczenia maszynowego o nazwie **Projected Gradient Descent (PGD)** do generowania perturbacji działających jako wyzwalacze. Metoda jest stosowana w atakach na etapie wnioskowania (ang. *inference time attacks*). Przyjrzymy się jej teraz bardziej szczegółowo. Zasadniczo wykorzystuje dane wejściowe i wyjściowe do generowania niezauważalnych danych (perturbacji), które po wprowadzeniu do modelu wpływają na jego działanie. Podczas gdy ataki unikowe wykorzystują tę technikę w fazie wnioskowania, `PoisoningAttackCleanLabelBackdoor` stosuje ją do tworzenia wyzwalaczy backdoora. Podobnie jak poprzednia metoda, także ta wymaga opakowania modelu we wrapper klasyfikatora ART, ale eliminuje potrzebę znajomości warstwy docelowej:

```
proxy = AdversarialTrainerMadryPGD(KerasClassifier(model), nb_epochs=10, eps=0.15,
    ↳eps_step=0.001)
proxy.fit(x_train, y_train)
```

Kompletny przykładowy notebook wykorzystujący zbiór danych MNIST do błędnej klasyfikacji wszystkich cyfr jako 9 w hipotetycznym scenariuszu oszustwa na czekach bankowych jest dostępny pod adresem: https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/poisoning_attack_clean_label_backdoor.ipynb.

Jak widać, łączenie skuteczności z niewykrywalnością to trudne zadanie dla atakujących, ale także dla obrońców. Techniki zatruwania stają się więc coraz bardziej zaawansowane, czego wynikiem są nowe metody badania modeli i techniki tworzenia perturbacji zatruwających.

Zaawansowane ataki zatruwające

Ataki z ukrytym wyzwalaczem i clean-label ukazały rosnący stopień wyrafinowania ataków zatruwających. Ta dziedzina nadal się rozwija, a narzędzia takie jak ART, Cleverhans i TextAttack są efektem badań przeprowadzanych na tym polu.

Zaawansowane ataki zatruwające dotyczą bardziej złożonych formatów danych, takich jak tekst naturalny czy pliki audio i wideo. Framework ART zapewnia na przykład ataki i perturbacje dla próbek dźwiękowych, podobne do tych oferowanych dla obrazów. Takie rozwiązanie nie jest w pełni udokumentowane, ale więcej informacji na jego temat można znaleźć w kodzie źródłowym: https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/art/attacks/poisoning/perturbations/audio_perturbations.py.

TextAttack to z kolei odpowiednik modułu ART służący do tworzenia adversarialnych danych tekstowych, które można wykorzystać do zatruwania modeli NLP. Więcej informacji na jego temat można znaleźć na stronie <https://textattack.readthedocs.io/en/latest/>.

Zaawansowane ataki mogą również wykorzystywać wyrafinowane techniki generowania trucizn, do których należą między innymi:

- **Globalne ataki manipulacyjne** (ang. *global manipulation attacks*). Tego typu ataki mają na celu pogorszenie ogólnego działania modelu i zmniejszenie jego skuteczności w szerokim zakresie danych wejściowych. Wprowadzają szeroką gamę zakłóceń i nie są ukierunkowane na określony rodzaj wprowadzanych danych.
- **Ataki zatruwające dostosowane do algorytmu** (ang. *algorithm-specific poisoning attacks*). Tego rodzaju ataki są projektowane tak, aby celować w określone typy algorytmów uczenia maszynowego. Polegają na tworzeniu złośliwych danych, które wykorzystują matematyczne właściwości konkretnego algorytmu uczenia, takiego jak **maszyny wektorów nośnych (SVMs)** (od ang. *Support Vector Machines*).
- **Ataki na gradient** (ang. *gradient-based attacks*). Ataki te polegają na manipulowaniu gradientami, których model używa do uczenia się. Powodują subtelne zmiany wyuczonych parametrów modelu w sposób, który powoduje jego niewłaściwe działanie.

Narzędzie ART realizuje wiele z takich ataków. Oferuje także notatniki z przykładowymi kodami. Więcej szczegółów na ten temat można znaleźć na stronie <https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/attacks/poisoning.html>.

Uwaga

Jak się okaże w dalszej części książki, nie jesteśmy w stanie bronić się przed wszystkimi możliwymi atakami, ale powinniśmy zrozumieć ich naturę i uczynić je częścią modelowania zagrożeń i oceny ryzyka. Pomoże nam to ustalić priorytety i zdecydować, przed którymi zagrożeniami należy się zabezpieczyć.

Przejdźmy teraz do kolejnego podrozdziału.

Zapobieganie i obrona

Po omówieniu różnych typów ataków zatruwających zastanówmy się, jak można się przed nimi bronić i ograniczać związane z nimi ryzyko. Takie działania stanowią połączenie tradycyjnych metod obronnych zintegrowanych w ramach MLOps z technikami uodparniającymi na ataki adversarialne.

Obrona cyfrowych twierdz przy użyciu MLOps

Do pewnego stopnia tradycyjne cyberbezpieczeństwo zapewnia mechanizmy obronne, które pomagają chronić się przed zatrutowaniem danych. Niektóre mechanizmy obronne, które poznaliśmy w rozdziale 3., mogłyby z powodzeniem sprawdzić się w tym kontekście. Przykładem mogą być tutaj takie metody jak przyznawanie dostępu z minimalnymi uprawnieniami, szyfrowanie oraz korzystanie z funkcji skrótu lub podpisywanie danych. Musimy jednak postrzegać te techniki jako część zintegrowanego systemu obrony łączącego techniki zabezpieczające z automatycznym śledzeniem, zatwierdzaniem, monitorowaniem i alarmowaniem.

W tym miejscu mogą przydać się operacje MLOps. Platformy takie jak AWS SageMaker, MLflow i Azure Machine Learning oferują usługi i mechanizmy obronne pomocne w obrobie przed atakami zatruwającymi. Są to między innymi:

- **Wersjonowanie danych i kontrola ich pochodzenia** w celu śledzenia zmian w zbiorach danych na przestrzeni czasu.
- **Walidacja danych** pozwalająca na ciągłe sprawdzanie integralności danych szkoleniowych. Dzięki temu możliwe jest zintegrowanie mechanizmów wykrywania anomalii, które omówimy w następnym punkcie.
- **Wersjonowanie modeli i kontrola ich pochodzenia** jako elementy zarządzania różnymi wersjami wytrenowanych modeli i śledzenia wprowadzonych zmian.
- **Ciągłe monitorowanie** wskaźników wydajności modelu w czasie rzeczywistym. Nieoczekiwany spadek może wskazywać na atak zatruwający.
- **Kontrola dostępu** polegająca na ograniczonym udostępnianiu danych i artefaktów modelu. Tylko upoważniony personel powinien być w stanie modyfikować te zasoby lub w ogóle mieć do nich dostęp. Ten środek jest zintegrowany z silnym uwierzytelnianiem i dodatkowymi mechanizmami autoryzacji.
- **Interpretowalność modelu** umożliwiająca zrozumienie jego działania. Jeśli model zaczyna funkcjonować w nieoczekiwany sposób dla określonych danych wejściowych, narzędzia interpretacyjne mogą pomóc zidentyfikować, czy pewnym cechom nie nadano dodatkowego znaczenia, co może być oznaką zatrucia.
- **Monitorowanie, logowanie i powiadamianie** w celu zapewnienia pozyskiwania informacji o podejrzanych aktywnościach, takich jak zmiany w danych czy trenowanie modeli w nietypowych godzinach.

- **Zarządzanie i współpraca** ułatwiające dzielenie się zasobami przy zastosowaniu odpowiednich zabezpieczeń, takich jak konieczność zatwierdzania dodatkowego dostępu lub wprowadzania znaczących zmian w modelu.

W dalszej części książki zastanowimy się, które operacje MLOps są istotne dla poszczególnych obszarów adversarialnej sztucznej inteligencji. Na przykład w rozdziale 6. skupimy się na podatnościach łańcucha dostaw i wyjaśnimy, jaki jest związek między pozyskiwaniem danych i modeli z innych źródeł a podatnością na ataki zatruwające. W rozdziale 16., w którym skupimy się na regułach MLSecOps, przedstawimy przykładową implementację całego procesu.

Teraz przejdziemy do omówienia konkretnych technik i mechanizmów obronnych, których możemy używać w połączeniu z zasadami MLOps.

Wykrywanie anomalii

Wykrywanie anomalii to technika pozwalająca identyfikować wśród danych wzory prowadzące do nieoczekiwanego działania modelu. Te odbiegające od normy wzory określa się mianem anomalii lub wartości odstających. W kontekście wykrywania ataków zatruwających algorytmy wykrywania anomalii wyszukują w zestawie treningowym dane, które znacznie odbiegają od większości, co może wskazywać na celową manipulację. Tej techniki możemy używać zarówno doraźnie, jak i w pełnej integracji z potokami MLOps w ramach walidacji danych.

W czym pomaga wykrywanie anomalii?

Wykrywanie anomalii stanowi skuteczną ochronę przed zatrutowaniem danych, zwłaszcza gdy jest realizowane w sposób zautomatyzowany. Może pomóc w następujących kwestiach:

- **Identyfikacja podejrzanych danych.** Wskazywanie danych, które wydają się odbiegać od normy, może być pomocne w wyszukiwaniu w zestawie treningowym takich pozycji, które mogły zostać wprowadzone lub zmodyfikowane przez atakującego.
- **Zautomatyzowane monitorowanie.** Wykrywanie anomalii można zautomatyzować w celu ciągłego monitorowania odbieranych danych treningowych i generowania w czasie rzeczywistym alertów informujących o przypadkach wykrycia podejrzanych danych.
- **Ograniczanie fałszywych alarmów.** Podczas gdy ręczna inspekcja może skutkować oznaczeniem zdrowych danych jako złośliwych z powodu błędu ludzkiego, to odpowiednio dostrojone algorytmy wykrywania anomalii mogą zmniejszyć liczbę takich fałszywych alarmów.

Techniki wykrywania anomalii w kontekście ataków zatruwających

Istnieje szereg technik wykrywania anomalii — od łatwych w użyciu metod statystycznych po bardziej wyrafinowane rozwiązania oparte na uczeniu maszynowym. Są to między innymi:

- **Metody statystyczne** — polegają na obliczaniu wskaźników statystycznych, takich jak średnia i odchylenie standardowe, a następnie oznaczaniu tych danych, które znacznie odbiegają od wyliczonych miar. Popularną miarą jest na przykład wartość z-score, która mówi, o ile odchylen standardowych konkretny punkt danych odbiega od średniej.
- **Metody klastrujące** — techniki takie jak k-średnich (jeden z algorytmów grupowania) mogą służyć do grupowania podobnych punktów danych. Te wartości, które nie pasują wyraźnie do żadnego klastra, można uznać za anomalie.
- **Sieci neuronowe** — modele głębokiego uczenia, zwłaszcza autokodery, można wytrenować do rekonstruowania danych wejściowych. Jeśli błąd rekonstrukcji dla punktu danych będzie zbyt wysoki, stosowną wartość można uznać za anomalię.
- **Metody gęstościowe** — algorytmy takie jak DBSCAN analizują gęstość punktów danych w określonym obszarze. Istnienie odosobnionych obszarów z niewielką liczbą danych może wskazywać na obecność anomalii.

Wyzwania i uwarunkowania

Wykrywanie anomalii może być skutecznym środkiem bezpieczeństwa w kontekście zatrucia danych, ale przy wprowadzaniu tego mechanizmu musimy wziąć pod uwagę następujące kwestie:

- **Dostrajanie.** Algorytmy wykrywania anomalii często obejmują parametry, które należy starannie dobrać. Nieprawidłowe dostrojenie ich wartości może prowadzić do wielu fałszywych wskazań (dodatnich lub ujemnych).
- **Zmieniające się dane.** W miarę jak do systemu wprowadzane są nowe, prawidłowe dane, mogą zmieniać się kryteria rozpoznawania. Aby zachować swoją skuteczność, systemy wykrywania anomalii muszą dostosowywać się do tych zmian.
- **Ukryte ataki.** W przypadku predykcyjnej sztucznej inteligencji wyrafinowani napastnicy mogą wprowadzać zatrute dane, które bardzo przypominają te prawdziwe. W rezultacie programy wykrywające anomalie mają problemy z ich identyfikacją. Jak zobaczymy w rozdziale 14., generatywna sztuczna inteligencja, zwłaszcza z modelami LLM, wprowadza nowe zagrożenia wynikające z wykorzystywania danych w skali całego internetu, które pochodzą spoza obszaru kontroli przedsiębiorstwa. Połączenie ogromnej skali tych danych z brakiem nadzoru nad nimi sprawia, że trudniej jest je weryfikować.

Testy odporności na zatrucie

Ogólne wykrywanie anomalii ma zastosowanie w odniesieniu do dowolnych zbiorów danych i nie uwzględnia wpływu na algorytmy uczenia maszynowego. Ataki zatruwające stają się jednak coraz bardziej wyrafinowane i wykorzystują zaawansowane mechanizmy do generowania niezauważalnych perturbacji zatruwających model. Użycie tego, co może wydawać się prawidłowymi danymi, może mieć druzgocące konsekwencje.

Bardziej kontekstowym podejściem do wykrywania zatruc jest testowanie modelu pod kątem odporności na ataki zatruwające.

Prostym sposobem mogłoby być użycie **rekordów kanarkowych** (ang. *canary records*). Byłby to niewielki zbiór rekordów jednoznacznie należących do określonej klasy. Na przykład zestaw może składać się z pięciu obrazów, które powinny być w oczywisty sposób klasyfikowane jako samoloty. Jeśli zostaną błędnie przyporządkowane, może to sygnalizować zatrucie danych. Podejście jest łatwe do wdrożenia i może identyfikować proste próby zatrucia, ale nie wykryje ataków typu backdoor, ponieważ rekordy kanarkowe nie będą zawierać wyzwalaczy.

Do testowania modeli pod kątem podatności na backdoory można wykorzystać perturbacje zatruwające dostępne w ART jako wyzwalacze backdoorów:

- `add_single_bd` — jest to metoda prosta w użyciu i można ją szybko zaimplementować dla dużej partii danych. Pozwala szybko ustalić, jak wrażliwy jest nasz model na choćby najmniejsze zmiany. Takie podejście jest szczególnie użyteczne na przykład w scenariuszu, w którym chcemy przetestować, jak system rozpoznawania twarzy reaguje na najdrobniejsze nawet zmiany.
- `add_pattern_bd` — jest to kolejne proste i szybkie w realizacji podejście pozwalające przetestować nieco bardziej złożone luki w zabezpieczeniach. Możemy na przykład przeanalizować trening modelu rozpoznawania znaków drogowych w autonomicznych samochodach i zbadać jego odporność na drobne perturbacje.
- `insert_image` — jest to najbardziej konfigurowalna i złożona opcja. Doskonale sprawdza się w testowaniu zaawansowanych wzorów i omijania filtrowania treści. Można ją stosować na przykład w przypadku bardziej złożonych zakłóceń w systemach rozpoznawania znaków drogowych lub oprogramowaniu OCR, a także wtedy, gdy chcemy sprawdzić, czy algorytm moderacji treści nadal potrafi poprawnie klasyfikować obraz po wstawieniu do niego określonego logo lub symbolu.

Są to techniki odpowiednie dla systemów operujących na obrazach. Framework ART oferuje także perturbacje dla próbek dźwiękowych, natomiast TextAttack pozwala tworzyć dane i przypadki testowe dla modeli NLP.

Zaawansowana ochrona przed zatruciem realizowana z użyciem narzędzi ART

Podczas gdy ogólne wykrywanie anomalii nie ocenia wpływu na model, to testy odporności mogą zbyt mocno koncentrować się na konkretnych przypadkach użycia. Framework ART oferuje kilka mechanizmów obrony przed zatruciem, które znajdują się gdzieś pośrodku. Opierają się one na analizach badawczych i łączą w sobie zaawansowane techniki wykrywania anomalii testowane na modelu w celu określenia wpływu danych na jego działanie.

Techniki te można zaimplementować jako wykonywalne programy w notatnikach lub zintegrować z potokami MLOps. Wyróżniamy następujące rodzaje takich mechanizmów:

- **Obrona oparta na analizie aktywacji** — jest szczególnie przydatna, gdy chcemy sprawdzić, czy wewnętrzne aktywacje modelu sieci neuronowej są realizowane w sposób nieprawidłowy, co mogłoby sugerować, że model przetwarza zatrute dane.
- **Obrona oparta na pochodzeniu danych** — ma zastosowanie, gdy źródło danych jest wątpliwe lub nie zostało dokładnie zweryfikowane. Ten rodzaj obrony pozwala uzyskać pewność, że dane pochodzą z wiarygodnego, uwierzytelnionego źródła. Ciągła weryfikacja źródła lub pełne zaufanie do niego może być jednak niepraktyczne lub ryzykowne. Poza tym potwierdzenie pochodzenia danych nie gwarantuje, że same dane nie są zatrute.
- **Metoda RONI** (ang. *Reject on Negative Impact*, czyli odrzucanie danych o negatywnym wpływie) — często używa się jej do wykrywania ataków zatruwających, które mają zauważalny wpływ na działanie modelu. W ramach tej techniki ocenia się wpływ każdego elementu treningowego na ogólną skuteczność modelu. Dane, które znacząco ją pogarszają, uznaje się za potencjalnie zatrute i są usuwane. Metoda ta często wymaga dużej mocy obliczeniowej, ponieważ konieczne może być wielokrotne trenowanie modelu. Poza tym bywa nieskuteczna w przypadku subtelnych ataków zatruwających, które nie od razu pogarszają działanie modelu.

Zazwyczaj trenuje się dwie wersje modelu: jedną z podejrzanym punktem danych i drugą bez niego. Następnie porównuje się skuteczności obu modeli. Jeśli dodanie określonego punktu danych ma negatywny wpływ, może to oznaczać, że jest zatruty.

- **Obrona oparta na sygnaturze spektralnej** — służy do identyfikacji ataków zatruwających, w których punkty danych znacząco odbiegają od typowych danych w przekształconej domenie, na przykład w domenie częstotliwości. Tego typu techniki mogą wymagać złożonych obliczeń i konfiguracji sprzętowych. Narzędzia takie jak ART upraszczają stosowanie tych technik i ich integrację z potokami MLOps. Więcej informacji na ich temat można znaleźć w dokumentacji ART i notatnikach:

- https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/defences/detector_poisoning.html,
- https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/poisoning_defense_activation_clustering.ipynb,
- https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/poisoning_defense_spectral_signatures.ipynb,
- https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/provenance_defence.ipynb.

Czwarty notatnik zawiera przykład demonstrujący zarówno kontrolę pochodzenia danych, jak i implementację zabezpieczeń RONI.

Opisane powyżej mechanizmy obronne to dziedzina, która rozwija się błyskawicznie. Elementem obrony przed adversarialną sztuczną inteligencją, w tym zatrutowaniem danych, jest śledzenie najnowszych koncepcji i wybieranie odpowiednich środków ochronnych oraz ich stosowanie i automatyzowanie.

Trening adwersarialny

Ten mechanizm obronny polega bardziej na łagodzeniu skutków ataku niż jego wykrywaniu. Dotyczy bezpośrednio zatrutych danych, które są prawidłowo klasyfikowane. Trening adwersarialny sprawia, że modele są odporne na takie ataki. Technika ta pomaga ograniczyć skutki ataków w fazie wnioskowania (ataków unikania), dlatego omówimy ją bardziej szczegółowo w części poświęconej takim właśnie atakom.

Budowanie strategii obronnej

Ataki zatruwające są zasadniczo atakami realizowanymi na etapie rozwoju modelu. Podstawową strategią obronną, którą należy wdrożyć, jest proces MLOps. Wprowadza on kilka podstawowych zasad zapewniających identyfikowalność danych w ramach cyklu życia rozwiązania AI/ML. Należą do nich między innymi:

- **Automatyzacja i orkiestracja pipeline'ów** — obejmujące nie tylko aplikacje, ale także dane i modele; przygotowuje się więc trzy podstawowe pipeline'y: aplikacji, danych i ML.
- **Ciągłe X** (ang. *continuous X*) — wzbogacenie praktyk ciągłej integracji (CI) i ciągłego dostarczania (CD) o trening modeli, ich obsługę i monitorowanie.
- **Zarządzanie wersjami** — zarówno danych, jak i modeli, realizowane w ramach rejestru modeli używających magazynów cech, które ułatwiają zarządzanie wielokrotnym stosowaniem tychże cech.
- **Śledzenie eksperymentów** — rozszerza kontrolę wersji kodu źródłowego o śledzenie używanych danych, modeli, a także ich wag i biasów.
- **Testowanie** — dotyczy danych i modeli w trzech podstawowych pipeline'ach i obejmuje nie tylko testy funkcjonalne, ale także testy bezpieczeństwa, odporności, biasu, wydajności i jakości.
- **Monitorowanie** — rozszerza monitoring systemu o kontrolowanie wydajności modeli i ich działania.

Zasady te zostały omówione bardziej szczegółowo na stronie <https://ml-ops.org/content/ml-ops-principles>.

Po wprowadzeniu powyższych reguł MLOps do cyklu życia rozwiązania AI możemy zaprojektować punkty kontrolne i zabezpieczenia, które pomogą nam kontrolować pochodzenie danych i przebieg zmian w modelach.

Zasady MLOps stanowią kluczowy element strategii obronnej. Należy jednak mieć świadomość, że nie będziemy narażeni na wszystkie rodzaje ataków, dlatego nie trzeba wdrażać wszystkich zabezpieczeń. Jak przekonamy się w rozdziale 14., dzięki modelowaniu zagrożeń nauczymy się decydować, które zabezpieczenia są najbardziej rozpowszechnione i najbardziej wartościowe. We wspomnianym rozdziale 14. przyjrzymy się, jak dodać technikę modelowania zagrożeń do strategii tworzenia AI opartej na zasadzie „secure-by-design” i jak ustalać priorytety zabezpieczeń.

Podsumowanie

Dokładnie zbadaliśmy ataki zatruwające — pierwszy niezwykle istotny obszar adversarialnej sztucznej inteligencji. Tego typu ataki są realizowane na etapie tworzenia modeli, dlatego mogą być trudne do wykrycia. Omówiliśmy podstawowe koncepcje i przyjrzelśmy się konkretnym przykładom. Przeanalizowaliśmy dokładnie realizację prostych i bardziej zaawansowanych ataków zatruwających. Wiedza ta może pomóc w testowaniu i ocenie modeli pod kątem ewentualnego zatrucia. Dowiedzieliśmy się również o innych sposobach obrony przed zatruwaniem danych, w tym wdrażaniu potoków MLOps, wykrywaniu anomalii i zaawansowanych mechanizmach obronnych oferowanych przez zestaw narzędzi ART.

Ataki zatruwające wymagają dostępu do danych treningowych. Polegają na zakłócaniu procesu szkolenia modelu w celu podważenia jego integralności. W następnym rozdziale przyjrzymy się innej metodzie wszczepiania backdoorów (bez zatruwania danych) i obniżania integralności modelu przez manipulowanie nim, przeprogramowanie go lub wprowadzenie konia trojańskiego.

A

- adwersarialna AI, 55, 140, 356, 370
 - a cyberbezpieczeństwo, 80
 - omijanie zabezpieczeń, 77
- adwersarialne dane wejściowe, 373
- AGI, Artificial General Intelligence, 28, 358
- AI, Artificial Intelligence, 27
- AI-GAN, Attack-Inspired GAN, 347
- algorytm
 - BIM, 187
 - DeepFace, 334
 - FaceNet, 334
 - FGSM, 184
 - JSMA, 188
 - MalGAN, 340
 - PGD, 192
- algorytmu uczenia maszynowego, 32
- analiza
 - głównych składowych, PCA, 33
 - składu oprogramowania, 71
- Android Studio, 133
- ANN, Artificial Neural Networks, 33
- anomalie, 108
- anonimizacja
 - dane geolokalizacyjne, 270
 - formatów audio i wideo, 277
 - k-anonimowość, 268
 - obrazów, 272
 - prosta, 264
 - zaawansowana, 268
- aplikacja mobilna usługi AI, 132
- aplikacje LLM, 359, 365
 - stosowanie LangChaina, 365
 - tworzenie, 360
- architektura
 - platformy MLSecOps, 553
 - systemu Enhanced FoodieAI, 503
- ART, Adversarial Robustness Toolbox, 85, 98
 - ochrona przed zatruciem, 110
 - zatrucie danych treningowych, 102
- artefakty, 70
- atak, 485
 - Carliniego-Wagnera, 190
 - dywergencyjny, divergence attack, 471
 - łączący, linkage attack, 267
 - MIFace, 238
 - na model CIFAR-10 CNN, 222
 - na model językowy, 195
 - phishingowy, 376
 - subpopulacyjny inferencyjny
 - z zatrutowaniem, 250
 - transferu wiedzy, 255
 - typu biała skrzynka, 184
 - typu DoS, 375
 - typu PGD, 187
 - unikowy, evasion attack, 79
 - z mapą istotności, 188
- ataki adwersarialne, 78, 85–173, 486, 488
 - ekstrakcja, 81
 - mechanizmy obronne
 - na poziomie aplikacji, 403
 - platformy LLM, 402
 - unikanie, 81
 - właściwe dla generatywnej AI, 488, 489
 - wnioskowanie, 81
 - zatrucie, 81
 - z użyciem GAN-ów, 345, 349
 - z użyciem promptów, 372
 - środki zaradcze, 401
- ataki czarnoskrzynkowe, 200
- ataki deserializacyjne, deserialization attacks, 161
- ataki ekstrakcyjne, extraction attacks, 81, 211, 470
 - ekstrakcja równoważna funkcjonalnie, 212
 - generatywne typu uczeń-nauczyciel, 219
 - obrona, 227
 - oparte na uczeniu, 214
- ataki inferencyjne, inference attacks, 81
- ataki inwersji modelu, 211, 470
 - generatywne, 241
 - na sieć CNN, 246
 - środki zaradcze, 239, 243, 244
 - typu Plug and Play, 245
 - w formie dystrybucyjnej, 243

- ataki inwersji modelu
 - w trybie
 - białej skrzynki, 236
 - czarnej skrzynki, 237
 - wariacyjne, 245
 - wspomagane sieciami GAN, 240
 - wykorzystanie poziomów pewności, 238
- ataki na implementacje AI, 177–235
- ataki na łańcuch dostaw, 140, 457, 489
 - manipulowanie modelami, 161
 - środki zaradcze, 468
 - zatrucie modelu, 156
- ataki na prywatność, 469
 - kradzież danych, 236–261
 - kradzież modeli, 210–235
 - środki zaradcze, 476
- ataki sieciowe z użyciem GAN-ów, 344
- ataki unikowe, evasion attacks, 81, 177
 - obrona, 202
 - perturbacje, 182
 - perturbacje jednoetapowe, 184
 - scenariusze, 183
 - techniki, 182
 - techniki rozpoznawcze, 179
 - w dziedzinie NLP, 195
 - z wykorzystaniem
 - BIM, 187
 - FGSM, 184
 - JSMA, 188
 - łatki adversarialnej, 193
 - PGD, 192
 - UAP, 198
- ataki wnioskowania, inference attacks, 236, 248
 - białoskrzynkowe, 256
 - na modele LLM, 472
 - o atrybutach, 248, 251
 - o przynależności, 211, 248, 253, 255
 - o przynależności z użyciem ART, 259
 - statystyczne wartości progowe, 254
 - ślepe, 255
 - środki zaradcze, 251, 257
- ataki zatrujące, poisoning attacks, 81, 85, 412
 - dostrajanie modeli LLM, 435
 - indukowanie stronniczości, bias induction, 86
 - konsekwencje, 89
 - nieukierunkowane, 87
 - okup i wymuszenie, 86
 - przygotowanie, 90
 - sabotaż konkurencyjny, 86
 - środki zaradcze, 451
 - typu backdoor, 87, 94
 - wyzwalacze backdoorowe, 98
 - z ukrytym wyzwalaczem, 103
 - zatrucie danych, 102
 - typu clean-label, 87, 104
 - ukierunkowane, 87
 - wstawianie backdoorów, 86
 - wykrywanie anomalii, 108
 - z rozdwojeniem widoku, 169
 - z wstępnie wytrenowanym modelem, 157
 - zaawansowane, 87, 106
 - dostosowane do algorytmu, 106
 - globalne manipulacyjne, 106
 - na gradient, 106
 - zakłócenie, 86
 - zapobieganie i obrona, 107
- atakowanie
 - aplikacji mobilnej, 132
 - brzegowej sztucznej inteligencji, 132
 - modeli ML, 85–173
 - poprzez wstrzykiwanie backdoorów, 115
 - koni trojańskich, 118
 - ładunku neuronowego, 127
 - poprzez interfejsy zarządzania modelami, 162
 - z wykorzystaniem warstwy niestandardowej, 124
- augmentacja gaussowska, Gaussian augmentation, 205
- autokodery wariacyjne, VAE, 35, 237
- AWS SageMaker, 52
- Azure Machine Learning, 52

B

- backdoor, 102, 115
 - jednopikselowy, 99
 - niezatruczający, poisonless backdoor, 162
- BERT, 34
- bezpieczeństwo
 - AI w przedsiębiorstwie, 579, 582
 - frameworki bezpieczeństwa, 56
 - hosta, 60
 - kodu, 71
 - kontrole bezpieczeństwa, 57
 - modelowanie zagrożeń, 57
 - narzędzia, 58
 - ofensywne, 333
 - operacje DevSecOps, 58, 77
 - operacje MLOps, 58, 77
 - operacyjne AI, 586
 - punktu końcowego, 62

- testy penetracyjne, 57
- triada PID, 56
 - wzmacnianie iteracyjne, 588
- bezpieczne obliczenia wielostronne, 283
- biblioteka
 - SEAL, 286
 - spaCy, 75
 - zagrożeń, 492–501
- BIM, Basic Iterative Method, 183, 187

C

- CIA, Confidentiality, Integrity, Availability, 56
- ciągła integracja, CI, 58, 544
- CNN, Convolutional Neural Networks, 27
- CUDA, 34
- cyberbezpieczeństwo, 80
- cyberzagrożenia, 486
- cykl życia
 - oprogramowania, 142
 - projektu ML, 32

D

- dane w ruchu, data in transit, 66
- dane w spoczynku, data at rest, 66
- deanonimizacja
 - atak łączący, 267
 - bezpośrednia reidentyfikacja, 267
- deepfake, 35
 - dźwiękowy, 329
 - techniki tworzenia, 313, 318, 325
 - wykrywanie, 313, 330
- dekompilacja aplikacji, 134
- destylacja wiedzy, knowledge distillation, 219
- DevSecOps, 58, 77
- długa pamięć krótkotrwała, LSTM, 34
- DM-CRISP, 32
- DNN, Deep NN, 46
- DoS, Denial of Service, 375
- dostrajanie, 367
 - modeli LLM, 435
 - scenariusze, 438
 - wektory ataku, 440
 - zatrucie bota ChatGPT, 441
- drzewo decyzyjne, decision tree, 32, 43
- duże modele językowe, *Patrz* modele LLM

E

- edge AI, 132

F

- falsyfikacja sieci CNN, 215
- FGSM, Fast Gradient Sign Method, 183–186
- firewall, 61
- FNN, Feedforward NN, 45
- framework
 - ART, 98
 - Cybersecurity Framework, 56
 - Flask, 50
 - Keras, 36
 - LangChain, 365
 - MLSecOps 2.0, 544
 - PyTorch, 36
 - scikit-learn, 36
 - TensorFlow, 36
 - TF Encrypted, 287
- funkcja aktywacji ReLU, 213

G

- GAN, Generative Adversarial Network, 35, 220, 237, 293, 298
- generatywna AI, GenAI, 293, 294, 456
 - ataki adversarialne, 488
 - autokodery, 295
 - autokodery warunkowe, 296
- modele
 - dyfuzyjne, 297
 - oparte na energii, 297
 - transformatorowe, 296
 - ograniczone maszyny Boltzmanna, 297
- generatywne sieci adversarialne, GAN, 35, 220, 237, 293, 295, 298
- deepfaki, 351
- deepfaki dźwiękowe, 329
- fałszowanie filmów, 324
- generowanie
 - fałszywych zdjęć, 314, 318
 - wrogich ładunków, 344, 345
- kodowanie wariacyjne, 350
- łamanie haseł, 337
- model
 - BigGAN, 310
 - CycleGAN, 309
 - Defense-GAN, 349
 - PGGAN, 310
 - Pix2Pix, 308
 - Pix2PixHD, 309, 321, 324
 - StarGAN v2, 311, 321
 - StyleGAN, 311, 314
 - StyleGAN2-ADA, 318, 320

generatywne sieci adwersarialne, GAN
 omijanie weryfikacji twarzy, 334
 oszukiwanie mechanizmów
 biometrycznych, 335
 przekształcanie kierunkowe obrazu, 320
 syntezywanie obrazów, 321
 szerzenie dezinformacji, 351
 środki zaradcze, 348
 techniki tworzenia deepfake'ów, 313,
 318, 325
 tworzenie, 299
 unikanie wykrywania złośliwego
 oprogramowania, 340
 w kryptografii, 342
 w steganografii, 342
 Wassersteina, 307
 wykrywanie ataków adwersarialnych, 350
 wykrywanie deepfake'ów, 313
 zabezpieczenia, 348
 złośliwe treści, 351
 generowanie SBOM, 153
 głębokie sieci Q, deep Q networks, 33
 Google Colab, 52
 GPT, 34

H

haszowanie, hashing, 264
 honeypot, 161

I

IntelliJ PyCharm, 39
 internet rzeczy, IoT, 28
 inwersja modelu, model inversion, 236, 470
 generatywna, GMI, 241
 wariacyjna, VMI, 245
 z użyciem dodatkowej wiedzy, KE-DMI, 243
 IoT, Internet of Things, 28

J

język Groovy, 559
 JSMA, Jacobian-based Saliency Map Attack,
 183
 Jupyter Notebook, 36, 41
 instalowanie, 42
 wybór jądra, 42
 zabezpieczenia, 74

K

k-anonimowość, 268
 Keras, 36, 45, 46
 warstwy lambda, 118
 klasteryzacja k-średnich, k-means clustering, 33
 klasyfikacje zagrożeń
 AI Exchange, 495–497
 branżowe, 490
 MITRE ATLAS, 498–501
 NIST AI, 491–495
 kodowanie
 kategoryjne one-hot, 48
 one-hot, 45
 termometryczne, thermometer encoding, 205
 koń trojański, 118, 124, 137
 kradzież modeli, 211

L

Lambda Labs Cloud, 53
 lista komponentów oprogramowania, SBOM,
 61, 141
 LLM, Large Language Models, 27, 356
 LLMOps, 571
 lobotomizacja modeli, model lobotomization,
 162
 losowe wygładzanie, randomized smoothing,
 208
 losowy las decyzyjny, random forest, 32, 43

Ł

ładunki neuronowe, neural payloads, 127
 łatki adwersarialne, adversarial patches, 193

M

macierz pomyłek, confusion matrix, 44, 98
 manipulacja osadzeniami zapytań, 419, 432
 maskowanie, masking, 264
 gradientów, gradient masking, 206
 maszyna wektorów nośnych, SVM, 33
 mechanizm RAG, 356, 367, 372, 396, 412
 mechanizmy zabezpieczające, 519–533
 metaklasyfikatory, 249
 mikroagregacja, microaggregation, 269
 minimalizacja całkowitej wariancji, 205
 ML, Machine Learning, 27
 MLOps, ML Operations, 31, 50, 58, 77, 85, 164
 MLP, Multilayer Perceptron, 45

MLSecOps

- architektura platformy, 553
- budowanie platformy, 552
- integracja
 - potoków z przepływami deweloperskimi, 566
 - z LLMOps, 571
- narzędzia do orkiestracji, 544–546
- pozyskiwanie modelu, 558
- rejestracja modelu, 564
- użycie SBOM-ów, 574
- walidacja modelu, 559, 563, 567
- wdrażanie, 541
- wzorcy, 547

model

- DM-CRISP, 31, 32
- GPT, 364
- zagrożeń, 503
 - przepływ generowania osadzeń, 507
 - przepływ obsługi użytkownika, 510
 - przepływ rozpoznawania składników, 506

modele

- GAN, 308–312
- LLM, 27, 356, 357
 - ataki na prywatność, 469
 - ataki wnioskowania, 472
 - ataki zatruwające, 435
 - dostrajanie, 367
 - ekstrakcja danych treningowych, 470
 - inwersja modelu, 470
 - klonowanie, 474
 - mechanizm RAG, 367, 393
 - modyfikowanie metodą ROME, 466
 - tworzenie aplikacji AI, 359
 - usługi weryfikacji, 406
 - wpływ na adversarialną AI, 370
 - wprowadzanie własnych danych, 366
 - zatrutowanie osadzeń, 412

ML

- atakowanie, 85–173
- manipulowanie, 137, 161
- prywatne repozytoria, 164
- przeprogramowanie, 138
- skanowanie i ewaluacja, 163
- zatrutowanie, 156

modelowanie zagrożeń, 501, 502

N

narzędzia

- do orkiestracji MLOps, 545, 546
- do weryfikacji i moderacji treści, 406

narzędzie

- Adversarial Robustness Toolbox, 79
- Amnesia, 269
- APKTool, 134
- ART, 85, 106, 177, 159, 169
- auditd, 61
- Bandersnatch, 149
- Bandit, 71, 74
- Burp Suite, 401
- fail2ban, 61
- Grype, 142
- JADX, 134
- jupyter notebook, 37
- LM Studio, 468
- logwatch, 61
- ModelScan, 76, 161, 315
- NBDefense, 74
- Neural Cleanse, 131
- Ollama, 468
- OpenSSL, 63
- OWASP Dependency-Check, 142
- pickle, 115
- pip, 35, 41
- Safety, 142
- Snyk, 142
- Syft, 153
- TextAttack, 170, 177
- Trivy, 72, 74, 142, 153
- Twine, 148
- Uncomplicated Firewall, 61
- venv, 35
- NLI, Natural Language Inference, 197
- NLP, Natural Language Processing, 33, 181
- normalizacja, 47
- notatnik Jupytera, 36, 41

O

obrona

- przed atakami ekstrakcyjnymi, 227
 - mechanizmy wykrywające, 231
 - środki prewencyjne, 227
 - ustalenie właściciela modelu, 232
- przed atakami na prywatność, 476
- przed atakami unikowymi
 - mechanizmy certyfikowane, 208
 - stosowanie zespołów modeli, 207
 - strategia obronna, 202
 - trening adversarialny, 203
 - wstępne przetwarzanie danych, 204
 - wzmacnianie modelu, 205

obrona

- przed atakami zatruwającymi
 - strategia obronna, 112
 - testy odporności, 109
 - trening adversarialny, 112
 - użycie MLOps, 107
 - użycie narzędzi ART, 110
 - wykrywanie anomalii, 108
 - przed atakiem na edge AI, 134
 - przed backdoorami, 117
 - przed koniem trojańskim, 122, 126
 - przed ładunkiem neuronowym, 131
 - przed manipulowaniem osadzeniami, 434
 - przed wstrzykiwaniem promptów, 404
 - przed zatrutymi zbiorami danych, 171
 - przed zatrutowaniem
 - bezpośrednim osadzeń, 433
 - danych, 433
 - dostrajania, 451
 - wstępnie wytrenowanych modeli, 158
- ocena
- ryzyka, 514–518
 - zagrożeń, 509
- odgrywanie ról, role-playing, 378
- odporne funkcje straty, robust loss functions, 207
- odwrócona powłoka, reverse shell, 161
- operacje uczenia maszynowego, MLOps, 85
- osadzenia, embeddings, 412

P

- pakiety Pythona, 35, 41
- PCA, Principal Component Analysis, 33
- perceptron wielowarstwowy, MLP, 45
- perturbacje, 178, 182
- adversarialne generatywne, GAP, 346
 - adversarialne uniwersalne, UAP, 198, 346
- PGD, Projected Gradient Descent, 105, 183, 192
- PID, Poufność, Integralność, Dostępność, 56
- podatności, 485
- powierzchnia ataku, attack surface, 63
- procesor graficzny, GPU, 34
- prompty, 360
- dodawanie ograniczeń, 389
 - generowanie bezpiecznych odpowiedzi, 404
 - kodowanie, 389
 - odgrywanie ról, 378
 - podszycanie się, 382
 - przetwarzanie języka, 389

- wstrzykiwanie, 373
 - bezpośrednie, 374, 388
 - pośrednie, 393
 - stylu, 378
 - zautomatyzowane, 391
- wyświetlanie niebezpiecznego rezultatu, 390
- zastępowanie, prompt override, 375

protokół

- OAuth2, 65
- SSH, 62
- SSL, 63
- TLS, 63

prywatność, 263

- anonimizacja danych, 264
- anonimizacja zaawansowana, 268
- różnicowa, Differential Privacy, 278
- uczenie dzielone, 282
- uczenie federacyjne, 281
- zaawansowane szyfrowanie, 283

przeprogramowanie modelu, 138

przesunięcie pikseli, pixel deflection, 205

przetwarzanie języka naturalnego, NLP, 33, 181

publikowanie

- zatrutego LLM-u, 459
- zmanipulowanego modelu LLM, 465

PyPI, Python Package Index, 35

Python

- instalacja, 39
- instalowanie pakietów, 41
- menedżer pip, 35, 41
- moduł venv, 40
- notatniki Jupytera, 41
- tworzenie aplikacji LLM, 360
- tworzenie wirtualnego środowiska, 40
- weryfikowanie instalacji, 41

PyTorch, 36

Q

Q-learning, 33

R

RAG, Retrieval-Augmented Generation, 356, 367, 372, 396

- działanie mechanizmu, 367
- funkcja wyszukiwania, 397
- zatrutowanie osadzeń, 412

redukcja cech, feature squeezing, 205

regresja

- liniowa, linear regression, 32
- logistyczna, logistic regression, 32

replikacja modelu, 475
repozytorium prywatne
 konfigurowanie zabezpieczeń, 151
 modeli, 164
 tworzenie, 165
REST API, 51
RNN, Recurrent Neural Networks, 27, 296
RODO, 262
RONI, Reject on Negative Impact, 86
ryzyko, 485
 inherentne, 513
 ocena, 514–518
 poziomy, 511
 rezydualne, 513
 rodzaje, 513

S

SBOM, Software Bill of Materials, 141, 172, 574
 generowanie, 153
scikit-learn, 36, 42
SEAL, Simple Encrypted Arithmetic Library, 286
Secure by Design, 482
 budowanie biblioteki zagrożeń, 485
 eksploatacja systemu, 535
 modelowanie zagrożeń, 501
 ocena ryzyka, 514
 ocena zagrożeń, 509
 osadzanie zabezpieczeń w cyklu życia, 535
 priorytetyzacja zagrożeń, 509
 projektowanie zabezpieczeń, 518
 testowanie i weryfikacja, 534
 zasada shift-left, 535
serializacja pickle'owa, 115
serwer PyPI, 145
sieci imitujące, knockoff nets, 216
sieci neuronowe
 głębokie, DNN, 46
 jednokierunkowe, FNN, 45
 konwolucyjne, CNN, 27, 34
 rozpoznawanie obrazów, 48
 typu GAN, 307
 rekurencyjne, RNN, 27, 34, 296
 sztuczne, ANN, 33
 neurony i warstwy, 33
 trening, 33
 uczenie głębokie, 34
skanowanie
 podatności, vulnerability scanning, 58, 72
 sekretów, 74
specyfikacja komponentów oprogramowania,
 SBOM, 172

SSL, Secure Socket Layer, 63
stochastyczny spadek gradientu, SGD, 33
SVM, Support Vector Machine, 33
sztuczna inteligencja, AI, 27
 generatywna, generative AI, 35, 293
 godna zaufania, Trustworthy AI, 481, 538
 ogólna, AGI, 28, 358
 zabezpieczenia korporacyjne, 584
szyfrowanie, 67
 homomorficzne, 285
 zaawansowane, 287

Ś

środki zaradcze, 519–533
środowisko programistyczne, 39

T

TensorFlow, 36
testowanie bezpieczeństwa aplikacji
 dynamiczne, DAST, 58
 statyczne, SAST, 58
testy, 534
 ART, 166
 bezpieczeństwa, 58
 odporności na ataki adversarialne, 160
 odporności na zatrucie, 109
 penetracyjne, 58
 wykrywające zatrucie, 158
TextAttack
 atak na model BERT, 195
TLS, Transport Layer Security, 63
transformery, 34
trening adversarialny, 112, 203
trojan zdalnego dostępu, RAT, 137
Trustworthy AI
 bezpieczeństwo, 538
 etyka, 539
tworzenie
 aplikacji AI, 359
 biblioteki zagrożeń, 485
 deepfake'ów, 313
 embeddingów, 413
 prywatnego repozytorium, 165
 rozwiązania klasyfikacyjnego, 45
 rozwiązań ML, 51
 sieci GAN, 299
 usługi AI, 46
 eksploracja danych, 47
 ewaluacja modelu, 49
 trenowanie modelu, 48

tworzenie

usługi AI

- tworzenie modelu, 48
- usługa wnioskowania, 50
- wdrażanie modelu, 50
- wstępne przetwarzanie danych, 47
- wybór algorytmu, 48
- zbiór danych CIFAR-10, 46
- wirtualnego środowiska, 40
- wyzwalaczy backdoorowych, 98
- zatrutych próbek, 90

U

uczenie

- adwersarialne na małej liczbie próbek, FSAL, 324
- dzielone, split learning, 282
- federacyjne, federated learning, 281
- głębokie, deep learning, 27
- maszynowe, ML, 27
 - algorytmy, 32
 - AWS SageMaker, 52
 - Azure Machine Learning, 52
 - cykl życia, 32
 - etapy, 30
 - głębokie, deep learning, 34
 - Google Colab, 52
 - Lambda Labs Cloud, 53
 - nadzorowane, 29, 43
 - narzędzia deweloperskie, 35
 - nienadzorowane, 29
 - opcje szyfrowania, 283
 - przez wzmacnianie, 29
 - rozwiązania na dużą skalę, 51
 - transferowe, 155, 181
 - zapewnianie prywatności, 289

usługa

AI

- dla Androida, 132
- tworzenie, 46
- zabezpieczenie, 58
- CodeArtifact, 146
- wnioskowania, 51
- uwierzytelnianie, 65

V

Visual Studio Code, 39

W

warstwy

- lambda, 118
- niestandardowe z końmi trojańskimi, 124
- weryfikacja, 534
- wirtualne środowisko
 - aktywowanie, 40
 - rejestrowanie jako jądra, 41
 - utworzenie, 40
 - wyświetlanie składników, 41
- wizualizacja osadzeń, 429
- właściciel ryzyka, risk owner, 509
- wnioskowanie
 - w języku naturalnym, NLI, 197
 - wspomagane zatruciem, 249
- wstępne przetwarzanie danych, 204
- wstrzykiwanie
 - backdoorów, 115
 - koni trojańskich, 118
 - ładunku neuronowego, 127
 - promptów, zapytań, 81, 373
 - bezpośrednie, 374
 - eskalacja uprawnień, 398
 - obrona, 404
 - pośrednie, 393
 - wydobywanie danych, 397
 - zautomatyzowane, 391
 - zdalne wykonywanie kodu, 399
- stylu, 378
- wygładzanie przestrzenne, spatial smoothing, 205
- wykrywanie anomalii, 108
- wyszukiwanie podatności, 71
- wyzwalacz backdoora, backdoor trigger, 99
- wzmacnianie modelu, 205

Z

zabezpieczanie

- AI, 145
 - na etapie projektowania, 482
 - przy użyciu MLSecOps, 541
- artefaktów, 70
- blokowanie podatnych komponentów, 147
- kodu, 70
- modeli, 76
- notatników Jupytera, 74
- pakietów
 - dzięki selekcji, 150
 - za pomocą AWS CodeArtifact, 148
 - za pomocą narzędzia Bandersnatch, 149

- platformy aplikacji, 409
- poprzez sprawdzanie wytrenowanych modeli, 163
- prywatnego repozytorium PyPI, 152
- rozwiązań LLM-owych, 408, 409
- sieci GAN, 348
- usługi AI, 58
 - bezpieczeństwo hosta, 60
 - kontrola dostępu, 69
 - ochrona danych, 66
 - ochrona sieci, 63
 - uwierzytelnianie, 65
- zależności, 71
- znaczenie SBOM-ów, 153
- zabezpieczenia
 - implementowanie, 518
 - projektowanie, 518
- zaciemnianie, obfuscation, 264
- zagrożenia związane z
 - atakami adversarialnymi, 487–489
 - łańcuchem dostaw, 141, 155, 168, 489, 490
 - przestarzałymi komponentami, 141
 - zależności AI od danych, 143
- zależność przechodnia, transitive dependency, 73
- zatrutowanie
 - danych, 168, 169
 - modelu, model poisoning, 86, 156
 - osadzeń, 412
 - bezpośrednie, 419, 427
 - podczas generowania osadzeń, 419, 420
 - scenariusze ataku, 419
 - środki zaradcze, 433
 - uniwersalne środki zaradcze, 434
 - zaawansowane, 428
- zbiory danych dla StyleGAN2, 314
- zbiór danych
 - CIFAR-10, 46
 - Sentiment140, 169
 - SNLI, 197
- zespoły modeli, model ensembles, 207

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

Podejmij wyzwanie: ochroń AI przed nadużyciami!

Wraz z rozwojem rewolucyjnych możliwości modeli AI pojawiają się nowe zagrożenia. Może to być na przykład manipulowanie działaniem sztucznej inteligencji, by celowo doprowadzić do błędnych decyzji. Tak właśnie prowadzi się ataki adversarialne. Konsekwencje takich manipulacji, jak również innych, mogą być bardzo poważne. Jednak zrozumienie ich istoty i wdrożenie adekwatnych zabezpieczeń stanowi ogromne wyzwanie.

Tę książkę docenią specjaliści do spraw cyberbezpieczeństwa, którzy chcą zdobyć umiejętności zabezpieczania systemów AI. Znajdą w niej uporządkowaną prezentację wyników badań i najnowszych standardów branżowych, z uwzględnieniem klasyfikacji: MITRE, NIST i OWASP. W przewodniku omówiono strategię zabezpieczania AI już na etapie projektowania — z wykorzystaniem modelowania zagrożeń, przy czym skoncentrowano się na integracji MLSecOps i LLMOps z systemami bezpieczeństwa przedsiębiorstwa. Dodatkowo przedstawiono przykłady wdrażania integracji ciągłej, strategii i narzędzi MLOps, a także mechanizmów kontroli bezpieczeństwa. Zaproponowano ponadto bazujący na klasycznych filarach NIST plan wzmacniania bezpieczeństwa AI w przedsiębiorstwie.

Ciekawsze zagadnienia:

- zatrutowanie danych, omijanie zabezpieczeń i naruszanie prywatności
- użycie sieci GAN do przeprowadzania ataków i generowania deepfake'ów
- nowe zagrożenia bezpieczeństwa LLM, w tym wstrzykiwanie promptów i ekstrakcja danych
- zatrutowanie LLM metodami: RAG, osadzeń i dostrajania
- nowe zagrożenia związane z łańcuchem dostaw i modelami LLM o otwartym dostępie
- wdrażanie operacji MLSecOps z integracją ciągłą MLOps i SBOM

John Sotiropoulos jest starszym architektem zabezpieczeń w firmie Kainos. Odpowiada za bezpieczeństwo AI i pracuje nad zabezpieczeniem systemów krajowych w rządzie, organach regulacyjnych i służbie zdrowia. Współpracuje też z organizacjami normalizacyjnymi i krajowymi agencjami do spraw cyberbezpieczeństwa. Prywatnie jest zapalonym geekiem i maratończykiem.

	KOD KORZYŚCI Sięgnij po więcej! ▶ 
 helion.pl	ISBN 978-83-289-2235-8  9 788328 922358
 HELION S.A. ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	Cena: 129,00 zł

<packt>