Machine learning security

With Python secure coding

CYDMLPy | 4 days | On-site or online | Hands-on

Your machine learning application works as intended, so you are done, right? But did you consider somebody poisoning your model by training it with intentionally malicious samples? Or sending specially-crafted input – indistinguishable from normal input – to your model that will get completely misclassified? Feeding in too large samples – for example, an image of 16Gbs to crash the application? Because that's what the bad guys will do. And the list is far from complete.

As a machine learning practitioner, you need to be paranoid just as any developer out there. Interest in attacking machine learning solutions is gaining momentum, and therefore protecting against adversarial machine learning is essential. This needs not only awareness, but also specific skills to protect your ML applications. The course helps you gain these skills by introducing cutting edge attacks and protection techniques from the ML domain.

Machine learning is software after all. That's why in this course we also teach common secure coding skills and discuss security pitfalls of the Python programming language. Both adversarial machine learning and core secure coding topics come with lots of hands on labs and stories from real life, all to provide a strong emotional engagement to security and to substantially improve code hygiene.

So that you are prepared for the forces of the dark side.

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills


29 LABS


12 CASE STUDIES

Audience

Python developers working on machine learning systems

Group size

12 participants

Preparedness

General machine learning and Python development

Outline

- Cyber security basics
- Machine learning security
- Input validation
- Security features
- Time and state
- Errors
- Using vulnerable components
- Cryptography for developers
- Wrap up

Standards and references

CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Learning about various aspects of machine learning security
- Attacks and defense techniques in adversarial machine learning
- Input validation approaches and principles
- Identify vulnerabilities and their consequences
- Learn the security best practices in Python
- Managing vulnerabilities in third party components
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in Python

Table of contents

Day 1

- Availability threats
- Dealing with AI/ML threats in software security
- *Lab – Compromising ML via model editing*

- Using ML in cybersecurity
  - Static code analysis and ML
  - ML in fuzz testing
  - ML in anomaly detection and network security
  - Limitations of ML in security
- Malicious use of AI and ML
  - Social engineering attacks and media manipulation
  - Vulnerability exploitation
  - Malware automation
  - Endpoint security evasion

Adversarial machine learning

- Threats against machine learning
- Attacks against machine learning integrity
  - Poisoning attacks
  - Poisoning attacks against supervised learning
  - Poisoning attacks against unsupervised and reinforcement learning
  - *Lab – ML poisoning attack*
  - *Case study – ML poisoning against Warfarin dosage calculations*
  - Evasion attacks
  - Common white-box evasion attack algorithms
  - Common black-box evasion attack algorithms
  - *Lab – ML evasion attack*
  - *Case study – Classification evasion via 3D printing*
  - Transferability of poisoning and evasion attacks
  - *Lab – Transferability of adversarial examples*
- Some defense techniques against adversarial samples
  - Adversarial training
  - Defensive distillation
  - Gradient masking
  - Feature squeezing
  - Using reformers on adversarial data
  - Provable defenses against adversarial attacks
  - *Lab – Adversarial training*
  - Caveats about the efficacy of current adversarial defenses
  - Simple practical defenses
- Attacks against machine learning confidentiality
  - Model extraction attacks

- Defending against model extraction attacks

Lab – Model extraction

- Model inversion attacks
- Defending against model inversion attacks

Lab – Model inversion

Denial of service

- Flooding
- Resource exhaustion
- Sustained client engagement
- Algorithm complexity issues
- Denial of service in ML
  - Accuracy reduction attacks
  - Denial-of-information attacks
  - Catastrophic forgetting in neural networks
  - Resource exhaustion attacks against ML
  - Best practices for protecting availability in ML systems

Day 2

› Input validation

Input validation principles
Denylists and allowlists
Data validation techniques
Lab – Input validation
What to validate – the attack surface
Where to validate – defense in depth
When to validate – validation vs transformations
Output sanitization
Encoding challenges
Unicode challenges
Lab – Encoding challenges
Validation with regex
Regular expression denial of service (ReDoS)
Lab – ReDoS
Dealing with ReDoS

Injection

- Injection principles
- Injection attacks
- SQL injection
  - SQL injection basics
  - *Lab – SQL injection*
  - Attack techniques
  - Content-based blind SQL injection
  - Time-based blind SQL injection
- SQL injection best practices
  - Input validation
  - Parameterized queries
  - *Lab – Using prepared statements*
  - Additional considerations
  - *Case study – Hacking Fortnite accounts*
  - SQL injection protection and ORM

- Code injection
  - Code injection via input() in Python
  - OS command injection
    - *Lab – Command injection*
    - OS command injection best practices
    - Avoiding command injection with the right APIs in Python
    - *Lab – Command injection best practices*
      - *Case study – Shellshock*
    - *Lab - Shellshock*
    - Python module hijacking
    - *Lab – Library hijacking*

Integer handling problems

- Representing signed numbers
- Integer visualization
- Integers in Python
- Integer overflow
- Integer overflows in ctypes and numpy
- Other numeric problems
  - Working with floating-point numbers

Input validation in machine learning

- Misleading the machine learning mechanism
- Sanitizing data against poisoning and RONI
- Code vulnerabilities causing evasion, misprediction, or misclustering
- Typical ML input formats and their security

Day 3

› Input validation

Files and streams

- Path traversal
- *Lab – Path traversal*
- Path traversal-related examples
- Additional challenges in Windows
- Virtual resources
- Path traversal best practices
- *Lab – Path canonicalization*

Format string issues

- Format string issues in Python

Unsafe native code

- Native code dependence
- *Lab – Unsafe native code in Python*
- Best practices for dealing with native code

Security features

Authentication

- Authentication basics
- Multi-factor authentication
- Time-based One Time Passwords (TOTP)
- Authentication weaknesses
- Password management
  - Inbound password management
    - Storing account passwords
    - Password in transit
    - *Lab – Is just hashing passwords enough?*
    - Dictionary attacks and brute forcing
    - Salting
    - Adaptive hash functions for password storage
    - Password policy
    - NIST authenticator requirements for memorized secrets
    - Password hardening
    - Using passphrases
    - Password change
    - Password recovery issues
    - Password recovery best practices
    - *Lab – Password reset weakness*
      - *Case study – The Ashley Madison data breach*
      - *The dictionary attack*
      - *The ultimate crack*
      - *Exploitation and the lessons learned*
    - Password database migration
  - Outbound password management
    - Hard coded passwords
    - Best practices
    - *Lab – Hardcoded password*
    - Protecting sensitive information in memory
    - Challenges in protecting memory

Information exposure

- Exposure through extracted data and aggregation

Case study – Strava data exposure

- Privacy violation
  - Privacy essentials
  - Related standards, regulations and laws in brief
  - Privacy violation and best practices
  - Privacy in machine learning
    - Privacy challenges in classification algorithms
    - Machine unlearning and its challenges
- System information leakage
  - Leaking system information
- Information exposure best practices

› Time and state

Race conditions

- File race condition
  - Time of check to time of usage – TOCTTOU
  - TOCTTOU attacks in practice
  - Insecure temporary file
- Race conditions in Python
  - Thread safety and the Global Interpreter Lock (GIL)
  - Avoiding race conditions in Python

Case study: TOCTTOU in Calamares (CVE-2019-13178)

› Errors

Error and exception handling principles

Error handling

- Returning a misleading status code
- Information exposure through error reporting

Lab – Flask information leakage

Exception handling

- In the except block. And now what?
- Empty except block

Lab – Exception handling mess

Day 4

› Using vulnerable components

Assessing the environment

Hardening

Malicious packages in Python

Case study – The British Airways data breach

Vulnerability management

- Patch management
- Vulnerability management
- Vulnerability databases
- Vulnerability rating – CVSS
- Bug bounty programs
- DevOps, the build process and CI / CD
- Dependency checking in Python

Lab – Detecting vulnerable components in Python

ML supply chain risks

- Common ML system architectures
- ML system architecture and the attack surface

Case study – BadNets

- Protecting data in transit – transport layer security
- Protecting data in use – homomorphic encryption
- Protecting data in use – differential privacy
- Protecting data in use – multi-party computation

ML frameworks and security

- General security concerns about ML platforms
- TensorFlow security issues and vulnerabilities

Case study – TensorFlow vulnerability in parsing BMP files (CVE-2018-21233)

› Cryptography for developers

Cryptography basics

Cryptography in Python

Elementary algorithms

- Random number generation
  - Pseudo random number generators (PRNGs)
  - Cryptographically strong PRNGs
  - Using virtual random streams
  - Weak PRNGs in Python
  - Using random numbers in Python
  - *Lab – Using random numbers*
  - True random number generators (TRNG)
  - *Case study – Equifax credit account freeze*
- Hashing
  - Hashing basics
  - Common hashing mistakes
  - Hashing in Python
  - *Lab – Hashing*

Confidentiality protection

- Symmetric encryption
  - Block ciphers
  - Modes of operation
  - Modes of operation and IV – best practices
  - Symmetric encryption in Python
  - *Lab – Symmetric encryption*
- Asymmetric encryption
  - The RSA algorithm
    - Using RSA – best practices
    - RSA in Python
- Combining symmetric and asymmetric algorithms
- Homomorphic encryption
  - Basics of homomorphic encryption
  - Types of homomorphic encryption
  - FHE in machine learning
  - Integrity protection
- Message Authentication Code (MAC)
  - MAC in Python
  - *Lab – Calculating MAC*

- Digital signature
  - Digital signature with RSA
  - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
  - Digital signature in Python
    - *Lab – Digital signature with ECDSA*
    - Some further key management challenges
- Certificates
  - Certificates and PKI
  - X.509 certificates
  - Chain of trust
  - PKI actors and procedures
  - Certificate revocation

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop
- Secure design principles of Saltzer and Schroeder

And now what?

- Software security sources and further reading
- Python resources
- Machine learning security resources