Machine learning security

With Python secure coding

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

The course bridges the two worlds of cybersecurity and machine learning. Starting from the core cybersecurity principles, it highlights how ML systems are exposed to threats - both pre-existing threats from the world of software security affecting these systems in unexpected ways and completely new kinds of threats that require a deeper understanding of adversarial machine learning.

The first step of understanding the security of ML is to analyze the relevant threats. We synthesize a threat model (the assets to protect, the security requirements, the attack surface, potential attacker profiles, and the actual threat model represented via attack trees) based on the existing threat models of NIST, Microsoft, BIML, and OWASP. We then explore the relationship of security and ML, from ML-driven static analysis tools and IDS to a brief glimpse at ML-assisted attack tools used by hackers today. We look at the most significant threats against Large Language Models (LLMs), following the OWASP LLM Top 10 2025 (among others). The bulk of the course deals with adversarial machine learning, and a detailed discussion of the four main attack subtypes: evasion, poisoning, model inversion, and model stealing as well as practical aspects of these attacks. Various labs on adversarial attack techniques (model editing, poisoning, evasion, transfer attacks, model inversion, model extraction) offer practical insights into vulnerabilities, while a discussion of defense techniques such as adversarial training, certified robustness, and gradient masking provide the possible countermeasures.

In the rest of the course we discuss some common software security weakness categories, such as input validation, improper use of security features, time and state, error handling and using vulnerable components, putting them in the context of machine learning wherever relevant. Finally, participants are equipped with a solid foundation in cryptography, covering essential knowledge and skills every developer should have as well as techniques of special interest for machine learning such as multiparty computation, differential privacy, and fully homomorphic encryption.

Skills and drills

29 LABS

19 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
- Correctly implementing various security features
- 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

› Cyber security basics

What is security?

Threat and risk

Cyber security threat types – the CIA triad

Consequences of insecure software

› Machine learning security

Cyber security in machine learning

- ML-specific cyber security considerations
- What makes AI / machine learning a valuable target?
- Inadvertent AI failures
- Some early ML abuse examples
- ML threat model
  - Creating a threat model for machine learning
  - Machine learning assets
  - Security requirements
  - Attack surface
  - Attacker model – resources, capabilities, goals
  - Confidentiality threats
  - Integrity threats (model)
  - Integrity threats (data, software)
  - Availability threats
  - Dealing with AI/ML threats in software security
  - *Lab – Compromising ML via model editing*
  - *Case study – ROME and PoisonGPT*
- 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

- Security of large language models (LLMs)
  - Security of LLMs vs ML security
  - BIML top 10 LLM security risks
  - OWASP LLM Top 10
  - Practical attacks on LLMs
  - Practical LLM defenses

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
  - Some practical evasion and poisoning attack algorithms

Lab – ML evasion attack

Case study – Classification evasion via 3D printing

  - Transferability of poisoning and evasion attacks

Lab – Transferability of adversarial examples

Day 2

› Machine learning security (continued)

Adversarial machine learning

- 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*

› Input validation

Input validation principles

Denylists and allowlists

Case study – Denylist failure in urllib.parse.urlparse()

What to validate – the attack surface

Where to validate – defense in depth

When to validate – validation vs transformations

Output sanitization

Encoding challenges

Unicode 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*
- Database defense in depth
- *Case study – SQL injection against US airport security*

- 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*
    - *Case study – Command injection in Ivanti security appliances*
- Process control
  - Python module hijacking

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
- *Case study – File spoofing in WinRAR*
- 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 (MFA)
- *Case study – The InfinityGauntlet attack*
- Time-based One Time Passwords (TOTP)
- 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
- *Lab – Using adaptive hash functions in Python*
- Password policy
  - NIST authenticator requirements for memorized secrets
  - Password database migration
  - Hard coded passwords
  - Best practices
  - *Lab – Hardcoded password*
  - Protecting sensitive information in memory
    - Challenges in protecting memory
    - *Case study – Microsoft secret key theft via dump files*

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

› Time and state

Race conditions

- File race condition
  - Time of check to time of usage – TOCTTOU
  - TOCTTOU attacks in practice
  - *Lab - TOCTTOU*
  - *Case study – Arbitrary file deletion via TOCTTOU in N-Able Agent*
  - Insecure temporary file

› Errors

Error and exception handling principles

Error handling

- Returning a misleading status code
- Information exposure through error reporting
  - *Lab – Flask information leakage*
  - *Case study – Information leakage via errors in Apache Superset*

Exception handling

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

Lab – Exception handling mess

Day 4

› Using vulnerable components

Malicious packages in Python

Case study – The Polyfill.io supply chain attack

Vulnerability management

Lab – Finding vulnerabilities in third-party components

- DevOps, the CI / CD build process and Software Composition Analysis
- 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

- Hashing
  - Hashing basics
  - Common hashing mistakes
  - Hashing in Python

  Lab – Hashing
- Random number generation
  - Pseudo random number generators (PRNGs)
  - Cryptographically secure PRNGs
  - Using virtual random streams
  - Weak PRNGs in Python
  - Using random numbers in Python

  Lab – Using random numbers

  Case study – Weak randomness in OpenVPN Access Server

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
- Combining symmetric and asymmetric algorithms

- Key exchange and agreement
  - Key exchange
  - Diffie-Hellman key agreement algorithm
  - Key exchange pitfalls and best practices
- Homomorphic encryption
  - Basics of homomorphic encryption
  - Types of homomorphic encryption
  - FHE in machine learning

Integrity protection

- Message Authentication Code (MAC)
  - Calculating HMAC in Python

  Lab – Calculating MAC
- Digital signature
  - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
  - Digital signature in Python

    Lab – Digital signature with ECDSA

Public Key Infrastructure (PKI)

- Key management challenges
- Certificates
  - Certificates and PKI
  - Chain of trust

› 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