Machine Learning Security

Category: Networking & Security | **Vendor:** Various

Duration: 32.00 hours (4 days) **26.0 CPD Hours** **Rating:** ⭐ 4.6 (5,878 reviews)

Course Information

Delivery Format: Instructor Led - Online

Course Overview

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.

About This Course

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

Who Should Attend

Python developers working on machine learning systems

Prerequisites & Entry Requirements

General Prerequisites:

General machine learning and Python development

Learning Outcomes

Upon successful completion of this course, participants will be able to:

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

Detailed Course Outline

DAY 1

Cyber security basics

- What is security?
- Threat and risk
- Cyber security threat types
- Consequences of insecure software

- Constraints and the market
- The dark side
- Categorization of bugs

- The Seven Pernicious Kingdoms
- Common Weakness Enumeration (CWE)
- CWE Top 25 Most Dangerous Software Errors
- Vulnerabilities in the environment and dependencies

Machine learning security

- Cyber security in machine learning

- ML-specific cyber security considerations
- What makes machine learning a valuable target?
- Possible consequences
- Inadvertent AI failures
- Some real-world 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
- 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

- 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

- Denial of Service
- Resource exhaustion
- Cash overflow
- Flooding
- 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

- Blacklists and whitelists
- Data validation techniques
- Lab – Input validation
- What to validate – the attack surface
- Where to validate – defense in depth
- How to validate – validation vs transformations
- Output sanitization
- Encoding challenges
- Lab – Encoding challenges
- Validation with regex
- Regular expression denial of service (ReDoS)
- Lab – Regular expression denial of service (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
- Additional considerations
- Lab – SQL injection best practices
- Case study – Hacking Fortnite accounts
- SQL injection and ORM

- Code injection

- Code injection via input()
- OS command injection

- Lab – Command injection in Python
- OS command injection best practices
- Avoiding command injection with the right APIs in Python
- Lab – Command injection best practices in Python
- Case study – Shellshock
- Lab – Shellshock
- Case study – Command injection via ping
- Python module hijacking
- Lab – Module hijacking
- General protection best practices
- Integer handling problems

- Representing signed numbers
- Integer visualization
- Integers in Python
- Integer overflow
- Integer overflow with ctypes and numpy
- Lab – Integer problems in Python
- Other numeric problems

- Division by zero
- Other numeric problems in Python
- Working with floating-point numbers
- Files and streams

- Path traversal
- Path traversal-related examples

- Lab – Path traversal
- Additional challenges in Windows
- Virtual resources
- Path traversal best practices
- Format string issues
- Unsafe native code

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

Security features

- Authentication

- Authentication basics
- Multi-factor authentication
- Authentication weaknesses – spoofing
- Case study – PayPal 2FA bypass
- 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 length
- Password hardening
- Using passphrases
- Password change
- Forgotten passwords
- 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

- Insecure temporary file

- Avoiding race conditions in Python

- Thread safety and the Global Interpreter Lock (GIL)

- Case study: TOCTTOU in Calamares

- Mutual exclusion and locking

- Deadlocks

- Synchronization and thread safety

Errors

- Error and exception handling principles
- Error handling

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

- In the except,catch block. And now what?
- Empty catch block
- The danger of assert statements
- Lab – Exception handling mess

Using vulnerable components

- Assessing the environment
- Hardening
- Malicious packages in Python
- Vulnerability management

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

DAY 4

Cryptography for developers

- Cryptography basics

- Cryptography in Python

- Elementary algorithms

- Random number generation

- Pseudo random number generators (PRNGs)

- Cryptographically strong PRNGs

- Seeding

- Using virtual random streams

- Weak and strong PRNGs in Python

- Using random numbers in Python

- Case study – Equifax credit account freeze

- True random number generators (TRNG)

- Assessing PRNG strength
- Lab – Using random numbers in Python
- Hashing

- Hashing basics
- Common hashing mistakes
- Hashing in Python
- Lab – Hashing in Python
- Confidentiality protection

- Symmetric encryption

- Block ciphers
- Modes of operation
- Modes of operation and IV – best practices
- Symmetric encryption in Python
- Lab – Symmetric encryption in Python
- Asymmetric encryption

- The RSA algorithm

- Using RSA – best practices
- RSA in Python
- Elliptic Curve Cryptography

- The ECC algorithm
- Using ECC – best practices
- ECC 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 in Python

- Digital signature

- Digital signature with RSA

- Digital signature with ECC

- Digital signature in Python

- Public Key Infrastructure (PKI)

- Some further key management challenges

- Certificates

- Chain of trust

- Certificate management – best practices

Security testing

- Security testing methodology

- Security testing – goals and methodologies

- Overview of security testing processes

- Threat modeling

- SDL threat modeling

- Mapping STRIDE to DFD

- DFD example
- Attack trees
- Attack tree example
- Misuse cases
- Misuse case examples
- Risk analysis
- Security testing techniques and tools

- Code analysis

- Security aspects of code review
- Static Application Security Testing (SAST)
- Lab – Using static analysis tools
- Lab – Finding vulnerabilities via ML
- Dynamic analysis

- Security testing at runtime
- Penetration testing
- Stress testing
- Dynamic analysis tools

- Dynamic Application Security Testing (DAST)
- Fuzzing

- Fuzzing techniques
- Fuzzing – Observing the process
- ML fuzzing

Wrap up

- Secure coding principles

- Principles of robust programming by Matt Bishop
- Secure design principles of Saltzer and Schröder
- And now what?

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

Skills You Will Learn:

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

Frequently Asked Questions

Q: What delivery options are available for Machine Learning Security?

We offer multiple delivery formats:

- Live Instructor-Led Classroom Online (Virtual/Live Online)
- Traditional Instructor-Led Classroom Training (ILT)
- On-site delivery at your offices anywhere in United Kingdom
- Private dedicated courses customized for your team

Q: How many CPD hours does this course provide?

The 4-day Machine Learning Security course provides up to 26.0 CPD hours of structured learning. CPD certificates can be provided upon request.

Q: What is the duration of the Machine Learning Security training?

The training takes place over 4 day(s), with each day lasting approximately 32.00 hours including breaks for lunch and refreshments.

Q: Do you provide corporate training for Machine Learning Security?

Yes, we provide corporate training, dedicated training, and closed classes for Machine Learning Security. Training can take place anywhere in United Kingdom including London, Manchester, Birmingham, Edinburgh, or live online allowing teams from across United Kingdom or internationally to attend.

Q: What related terms do people search for?

Popular related searches include: Machine Learning Security, Machine Learning Security, MLSEC

Q: Why choose Nexus Human for Machine Learning Security?

Nexus Human is recognized as one of the leading training providers. Our trainers have won multiple awards including:

- Small Firms Best Trainer Award
- National Training Partner of the Year (Ireland) - Multiple Years
- Global Top 30 Instructor Awards (2012, 2019, 2021)
- Tech Excellence Award Nominations
- Learning Performance Institute (LPI) External Training Provider Sponsor 2024

Q: Are there any discount codes available?

Yes! Use discount code **PENPAL5** when booking your Machine Learning Security training. Please note that only one discount code can be used per booking and cannot be combined with other special offers.

Nexus Human

Professional Training & Development

Email: info@nexushuman.com

Website: www.nexushuman.com

Phone: +353 1 889 8700 (Ireland) | +44 20 8142 8961 (UK)

© 2026 Nexus Human. All rights reserved. This brochure was generated on 19/10/2026.