Module 3:

Adarsh KUMAR

Universitat Politècnica de Catalunya
Department of Computer Science

Project Coordinator:
Prof. Ilker Demirkol

MERiT Project
September 3, 2025

Co-funded by
the European Union

Table of Contents

1. Threat Models
2. Attack Types & Defense
3. Applications
4. Zero-day Threats
5. Advanced Threats
6. Feature Extraction
7. Generalization

MERiT

Co-funded by the European Union

Adversarial Machine Learning (AML) in Cybersecurity

What is Adversarial Machine Learning?

- Studies techniques where attackers craft inputs to deceive ML models.
- Adversarial examples cause incorrect predictions without raising suspicion.
- Critical for understanding ML vulnerabilities in security systems.

MERiT

Co-funded by the European Union

Threat Models in Adversarial ML

Threat models define attacker knowledge and capabilities, shaping attack and defense strategies.

1. White-Box Threat Model

- Attacker has complete knowledge of the model: architecture, parameters, training data, feature extraction.
- Enables highly effective gradient-based attacks (e.g., FGSM, PGD).
- *Examples:*
  - Modifying malware binaries to evade classifiers.
  - Altering network traffic features to bypass anomaly detection.

MERIT

Co-funded by the European Union

Threat Models in Adversarial ML

2. Black-Box Threat Model

- Attacker only queries the model and observes outputs.
- Uses probing and surrogate models to infer decision boundaries.
- *Examples:*
  - Crafting spam emails to evade filters by trial and error.
  - Generating network packets that avoid alerts via model response analysis.

MERiT

Co-funded by the European Union

Why Threat Models Matter

- Guide defense design: defenses depend on attacker knowledge level.
- Influence risk assessment: white-box attacks are more challenging to defend.
- Shape training methods: adversarial training adapts models for robustness.

MERiT

Co-funded by the European Union

Types of Adversarial Attacks in ML

ML models face diverse attacks aiming to compromise integrity, confidentiality, or availability. **Evasion Attacks**

- Occur during deployment; crafted inputs evade detection or cause misclassification.
- Examples: Modifying malware to avoid detection, altering packet features.
- Defenses: Adversarial training, robust features, adversarial example detection.

MERiT

Co-funded by the European Union

Types of Adversarial Attacks in ML

Poisoning Attacks

- Target training data by injecting malicious samples.
- Examples: Fake benign traffic logs, skewing spam filter boundaries.
- Defenses: Data sanitization, robust training, monitoring training data.

Model Inversion Attacks

- Extract sensitive training data by querying the model.
- Examples: Recovering biometric data, inferring confidential attributes.
- Defenses: Limit output info, differential privacy, query rate limiting.

MERiT

Co-funded by the European Union

Designing resilient machine learning systems requires securing all phases — training, deployment, and post-deployment — while maintaining a balance between performance and privacy.

Effective defense against adversarial attacks requires a combination of strategies. Adversarial training remains one of the most effective defenses, while gradient masking and detection techniques provide additional layers of protection to balance robustness, efficiency, and accuracy.

MERiT

Co-funded by the European Union

Applications in Cybersecurity

Model hardening refers to techniques and processes that improve the resilience of ML models against adversarial attacks and other vulnerabilities.

- **Adversarial Training:** Incorporating adversarial examples during training to improve model robustness.
- **Regularization Techniques:** Prevent overfitting to improve generalization and reduce susceptibility to attacks.
- **Robust Feature Engineering:** Selecting features less prone to manipulation by attackers.
- **Monitoring and Updating:** Continuously evaluating model performance and retraining to adapt to emerging threats.

MERiT

Co-funded by the European Union

Benefits:

- Reduces risk of evasion and poisoning attacks.
- Enhances trustworthiness of ML-driven security tools.
- Maintains high accuracy even under adversarial conditions.
- Integrate hardened ML models in IDS to detect sophisticated and evasive attacks.
- Implement early detection of adversarial manipulation attempts.
- Adapt systems to new attack tactics through continuous learning.

MERiT

Co-funded by the European Union

IDS monitor network or system activities to detect malicious behavior. ML-powered IDS leverage pattern recognition and anomaly detection to improve accuracy and adaptability.

ML Contributions to IDS:

- **Anomaly-based detection:** Using models like Isolation Forest, Autoencoders, or clustering to detect deviations from normal behavior.
- **Signature-based detection enhancement:** Automatically updating signatures based on learned attack patterns.
- **Adaptive learning:** Continuously learning from new attack data to detect zero-day and evolving threats.
- **Reduced false positives:** Employing sophisticated models to distinguish benign anomalies from actual attacks.

MERiT

Co-funded by the European Union

Challenges addressed:

- Detecting novel or obfuscated attacks.
- Scaling to high-volume and high-velocity network data.
- Handling concept drift where normal behavior changes over time.

MERIT

Co-funded by the European Union

Machine learning model hardening and robust IDS represent vital applications of ML in cybersecurity. By strengthening models and enhancing detection capabilities, organizations can better defend against sophisticated cyber threats and reduce security risks.

MERiT

Co-funded by the European Union

Zero-day Threats: Understanding Zero-day Exploits and APT Behavior

Zero-day threats represent some of the most challenging and dangerous risks in cybersecurity. They involve attacks that exploit previously unknown vulnerabilities, making detection and defense difficult.

Definition: A zero-day exploit is a cyberattack that targets a software vulnerability unknown to the vendor or security community at the time of the attack.

MERiT

Co-funded by the European Union

Key characteristics:

- No available patches or fixes when exploited.
- Can cause significant damage before detection.
- Often sold or traded in underground markets.

Examples:

- Exploiting a new buffer overflow vulnerability in widely used software.
- Targeting undisclosed flaws in operating systems or network devices.

MERIT

Co-funded by the European Union

Advanced Persistent Threats (APTs) and Zero-Day Use

APTs are sophisticated, long-term cyberattack campaigns typically orchestrated by well-funded and skilled adversaries, such as nation-states or organized cybercriminal groups.

How APTs Use Zero-Day Exploits

- **Initial access:** Using zero-days to infiltrate target networks stealthily.
- **Lateral movement:** Exploiting vulnerabilities to move within networks undetected.
- **Persistence:** Establishing long-term presence to extract data or disrupt operations.
- **Evasion:** Utilizing zero-day exploits to avoid traditional detection mechanisms.

MERiT

Co-funded by the European Union

Advanced Persistent Threats (APTs) and Zero-Day Use

Challenges Posed by Zero-Day Threats

- **Detection difficulty:** Signature-based systems cannot detect unknown exploits.
- **Response time:** Organizations must respond quickly without official patches.
- **Attribution complexity:** Difficult to identify attackers due to stealthy tactics.
- **High impact:** Potential for widespread damage in critical infrastructure.

MERiT

Co-funded by the European Union

Mitigation Strategies

- Employ behavioral anomaly detection to spot unusual activities.
- Use threat intelligence feeds to stay informed about emerging zero-days.
- Implement defense in depth with layered security controls.
- Regularly update and patch known vulnerabilities to reduce attack surface.
- Invest in incident response planning for rapid containment.

MERiT

Co-funded by the European Union

Feature Extraction for Malware Detection

Understanding zero-day exploits and APT behavior is critical for modern cybersecurity defense. Proactive detection, rapid response, and continuous monitoring form the backbone of mitigating these advanced threats.

Effective malware detection relies heavily on extracting meaningful features from software samples. Feature extraction can be broadly divided into two categories:

Static Analysis Features

- Opcodes
- Strings

Dynamic Analysis Features

- Sandbox behaviors

MERiT

Co-funded by the European Union

Feature Extraction for Malware Detection

Effective malware detection relies heavily on extracting meaningful features from software samples. Feature extraction can be broadly divided into two categories: static and dynamic.

Static analysis inspects the malware binary without executing it. It focuses on inherent properties that can indicate malicious behavior.

MERiT

Co-funded by the European Union

Feature Extraction for Malware Detection

Common static features:

- **Opcodes (Operation Codes):** Extract sequences of machine instructions to identify malicious patterns. Analyze frequency and order of opcodes to detect anomalies or known malicious signatures.
- **Strings:** Extract embedded text such as URLs, file paths, registry keys, or suspicious commands. Presence of certain keywords (e.g., `cmd.exe`, `powershell`) can indicate malicious intent.
- **Metadata:** File headers, imported libraries, and section characteristics provide contextual clues.

MERiT

Co-funded by the European Union

Feature Extraction for Malware Detection

Advantages:

- Fast and safe as no code execution is required.
- Useful for large-scale preliminary scanning.

Limitations:

- Cannot detect behavior triggered only at runtime.
- Easily evaded by obfuscation or packing.

MERiT

Co-funded by the European Union

Dynamic Feature Extraction

Dynamic analysis involves executing the malware in a controlled environment (sandbox) to observe its behavior.

Common dynamic features:

- **System calls and API calls:** Monitor interactions with the operating system, such as file access, network connections, or registry edits.
- **Process and network activity:** Track spawning of child processes, unusual resource usage, or outbound connections to suspicious IPs.
- **Behavioral patterns:** Detect patterns like persistence mechanisms, privilege escalation attempts, or data exfiltration behavior.

MERiT

Co-funded by the European Union

Dynamic Feature Extraction

Advantages:

- Captures real behavior, including actions triggered conditionally.
- Harder for malware to hide malicious activity.

Limitations:

- Resource-intensive and slower than static analysis.
- Some malware detect sandbox environments and alter behavior.

MERiT

Co-funded by the European Union

Generalization in Machine Learning

Combining static and dynamic feature extraction provides a comprehensive view for malware detection. While static analysis offers speed and scalability, dynamic analysis uncovers deeper behavioral insights, enabling more accurate and robust detection systems.

Generalization in Machine Learning

In cybersecurity applications, especially malware and intrusion detection, machine learning models must generalize well to unseen threats rather than just memorizing known signatures. Generalization is the model's ability to perform accurately on new, unseen data beyond the training set.

MERiT

Co-funded by the European Union

Overfitting and Generalization in Cybersecurity Machine Learning

What is Overfitting? Overfitting occurs when a model learns the noise or specific patterns in the training data too well, including known signatures, rather than the underlying general features. Such models perform excellently on training data but poorly on new, unseen threats.

Why is Overfitting to Known Signatures a Problem?

- Cyber threats constantly evolve; new malware and attack variants appear regularly.
- Overfitted models fail to detect zero-day or novel attacks that do not match known signatures.
- It reduces the practical effectiveness of ML-based security tools.

MERiT

Co-funded by the European Union

Techniques to Improve Generalization

- **Regularization:** Methods like L1/L2 regularization penalize overly complex models to prevent memorization of noise.
- **Cross-Validation:** Using k-fold cross-validation ensures the model performs consistently across different data subsets.
- **Feature Selection and Engineering:** Focus on features that capture fundamental behaviors rather than superficial signatures.
- **Data Augmentation:** Introduce variations in training data, including synthetic or adversarial examples, to improve robustness.
- **Early Stopping:** Halt training once performance on validation data stops improving, avoiding over-training on training data.
- **Ensemble Methods:** Combining multiple models reduces the risk of overfitting to specific data patterns.

MERiT

Co-funded by the European Union

Practical Implications

- Models trained with strong generalization can detect previously unseen attack variants.
- They reduce false negatives by avoiding dependence on static or known attack signatures.
- Generalization enables adaptive security systems that evolve with emerging threats.

MERiT

Co-funded by the European Union

Machine Learning and Malware Detection Pipelines

Ensuring good generalization is critical for machine learning models in cybersecurity. Avoiding overfitting to known signatures empowers models to detect novel, evolving threats effectively and maintain strong defenses over time. **Machine Learning in Malware Detection Pipelines** Machine learning plays a crucial role in automating and improving malware detection. Different algorithms and model architectures are applied throughout detection pipelines depending on the data and task.

MERiT

Co-funded by the European Union

1. Random Forests

Random Forests are ensemble learning methods based on decision trees, widely used due to their interpretability and strong performance on structured data.

Application:

- Used primarily on static features like opcode frequencies, file metadata, and extracted strings.
- Handle high-dimensional feature spaces well.
- Robust to noise and less prone to overfitting compared to single decision trees.

MERiT

Co-funded by the European Union

2. Convolutional Neural Networks (CNNs)

Benefits:

- Efficient training and inference.
- Can provide feature importance for interpretability.

CNNs excel at extracting spatial features and patterns, traditionally used in image processing but increasingly applied to malware detection.

Application:

- Treat malware binaries or opcode sequences as images or grids.
- Extract hierarchical patterns from raw bytecode or transformed feature representations.
- Useful for detecting obfuscated or polymorphic malware variants.

MERiT

Co-funded by the European Union

3. Sequence Models (RNNs, LSTMs)

Benefits:

- Automatically learns complex feature representations.
- Reduces the need for manual feature engineering.

Sequence models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) capture temporal dependencies and sequential patterns.

Application:

- Analyze dynamic behavior logs or sequences of API/system calls.
- Model the temporal evolution of malware behavior during execution.
- Detect suspicious patterns in event sequences indicative of malware activity.

MERIT

Co-funded by the European Union

3. Sequence Models (RNNs, LSTMs)

Benefits:

- Capture context and long-range dependencies.
- Effective for dynamic malware analysis and behavior modeling.

MERIT

Co-funded by the European Union

Integrating Random Forests, CNNs, and sequence models

Integrating Random Forests, CNNs, and sequence models in malware detection pipelines allows leveraging complementary strengths. Random Forests provide robust classification on static data, CNNs excel at learning complex patterns in raw data, and sequence models capture behavioral dynamics — together enabling more accurate and resilient malware detection.

MERiT

Co-funded by the European Union

Numerical examples

A network monitoring system uses machine learning to detect anomalies (potential cyberattacks) based on network traffic features. The system is trained on historical data using a **One-Class SVM**, which is commonly used for anomaly detection tasks.

The input features used are:

- X_1: Number of connections per second
- X_2: Average packet size
- X_3: Percentage of failed connection attempts

MERiT

Co-funded by the European Union

Training Data (Normal Traffic)

The model is trained only on **normal** traffic data to learn the typical behavior and distinguish it from anomalies.

Sample	X_1 (conn/s)	X_2 (pkt size)	X_3 (% fails)
1	5.0	300	1.0
2	7.0	280	0.5
3	6.0	310	0.8
4	5.5	295	1.2

MERIT

Co-funded by the European Union

One-Class SVM

The One-Class SVM learns that normal traffic lies within this region in the feature space.

$$X_A = [6.5,\ 290,\ 1.0]$$

Prediction: Normal
This input is close to the normal region learned by the model, so no anomaly is detected.

$$X_B = [20,\ 100,\ 15]$$

MERiT

Co-funded by the European Union

One-Class SVM

Prediction: Anomaly Detected!

- High connection rate
- Unusually small packet size
- High failure rate

These values deviate significantly from normal patterns, suggesting a likely **Denial of Service (DoS)** attack.

The system can respond to X_B by:

- Triggering an alert
- Throttling or blocking the source IP
- Notifying the security operations team

MERIT

Co-funded by the European Union

One-Class SVM

Context: Suppose we have a malware detection model that classifies files as malicious (1) or benign (0) using three features:

- X_1: Frequency of opcode `MOV`
- X_2: Number of network calls
- X_3: Use of suspicious strings

Base Model (simplified logistic regression):

$$\hat{y} = \text{sigmoid}(w_1 X_1 + w_2 X_2 + w_3 X_3 + b)$$

MERIT

Co-funded by the European Union

One-Class SVM

Parameters:

$$w = [2.0, 1.5, 2.5], \quad b = -5$$

$$X = [3, 4, 2] \Rightarrow \hat{y} = \text{sigmoid}(2 \cdot 3 + 1.5 \cdot 4 + 2.5 \cdot 2 - 5)$$

$$= \text{sigmoid}(6 + 6 + 5 - 5) = \text{sigmoid}(12) \approx 0.9999 \Rightarrow \text{Detected as malicious}$$

The attacker computes gradients and applies a small adversarial perturbation:

$$X' = [2.5, 2, 1.5] \quad \text{(slightly reduces key values)}$$

$$\Rightarrow \hat{y} = \text{sigmoid}(2 \cdot 2.5 + 1.5 \cdot 2 + 2.5 \cdot 1.5 - 5)$$

$$= \text{sigmoid}(5 + 3 + 3.75 - 5) = \text{sigmoid}(6.75) \approx 0.9988$$

MERiT

Co-funded by the European Union

One-Class SVM

Still detected as malicious, but closer to the threshold.

$$X'' = [1.5, 1, 1.0] \Rightarrow \hat{y} = \text{sigmoid}(2 \cdot 1.5 + 1.5 \cdot 1 + 2.5 \cdot 1.0 - 5)$$
$$= \text{sigmoid}(3 + 1.5 + 2.5 - 5) = \text{sigmoid}(2) \approx 0.88$$

With a threshold of 0.9, this input could be misclassified as benign, successfully evading the system.

MERiT

Co-funded by the European Union