The practical divide between advML research and security practice

A Red Team Perspective
Azure Sentinel - Overview

Events: 7.7M
Alerts: 169.4K
Incidents: 7

Recent incidents:
- Anomalous login
- Suspicious authentication attempts
- Traffic to known bad IPs
- Kerberos service ticket was issued
- Signins from IP's that attempted

Data source anomalies:
- CommonSec
- Perf
- AuditLogCL
State of Security of ML

- Awareness of risk is low
- Low ML security understanding
- Security posture is close to zero

**Microsoft survey**

25 out of 28 orgs we spoke to do not know how to secure their ML Systems

Poisoning is the threat in business decision makers.

Others: Model stealing, Model Inversion, Adversarial Examples

**Gartner research**

"Machine Learning presents a new attack surface and increases security risks... Application leaders must anticipate and prepare to mitigate risks of data corruption, model theft, and adversarial examples."
ML Red Teaming

“adversarial ML” attack tree
(assumes you require/have access to a model)

Goal: trick OCR for expense fraud

- Photoshop digits to fool human and machine
- Adversarial example to fool machine only (plausible deniability)

“red teaming” attack tree
(includes upstream stages and non-algorithmic alternatives)

Goal: commit expense fraud

- Submit valid receipt
  - never paid for item
  - duplicate receipt

Goal: modify receipt

- Gain access to OCR system

- Photoshop digits to fool human and machine
- Adversarial example to fool machine only (plausible deniability)
  - high confidence score bypass human inspection

Expense processing system that uses Machine Learning

Fraud: Reimburse for more than spent for business purposes
Machine Learning Development Lifecycle

- Poisoning: Data curation & labeling
- Backdoor: Feature extraction
- Evasion: Model training
- Stealing: Model validation
- Monitorin g: Model deployment

Causative attacker influence

Exploratory attacker influence

(Feedback)
Case study: an internal Microsoft Service

The system: oversubscribe for efficiency

→ ML is one piece of a larger system
→ A non-security model, with security implications

Threat model: “noisy neighbor” denial of service

→ **ML integrity** violation → system availability violation
→ Directly accessible only via private, internal API
Attack chain: Noisy Neighbor DoS

1. Credentials via phish
2. Insider access via valid account
Attack chain: Noisy Neighbor DoS

1. Credentials via phish

2. Insider access via valid account

3. Overprivileged data and code storage
Attack chain: Noisy Neighbor DoS

1. **Credentials via phish**

2. **Insider access**
   via valid account

3. **Overprivileged data and code storage**

4. **Model theft**: build a local copy of model
Attack chain: Noisy Neighbor DoS

1. **Credentials via phish**

2. **Insider access via valid account**

3. **Overprivileged data and code storage**

4. **Model theft: build a local copy of model**

5. **Model evasion via algorithmic attack**
Attack chain: Noisy Neighbor DoS

1. **Credentials via phish**

2. **Insider access via valid account**

3. **Overprivileged data and code storage**

4. **Model theft: build a local copy of model**

5. **Model evasion via algorithmic attack**

6. **Collect evasive variants**

- Data storage
- Training code
Attack chain: Noisy Neighbor DoS

1. Credentials via phish

2. Insider access via valid account

7. Request new account

6. Collect evasive variants

Web portal request

API boundary

Provision resource

Predict resource use

Oversubscribe?
Attack chain: Noisy Neighbor DoS

1. Credentials via phish
2. Insider access via valid account
3. Request new account
4. Request resources and deploy noisy neighbors
5. Collect evasive variants
6. Provision resource
7. Oversubscribe?
8. Denial of service

Web portal request

API boundary
“Internal” models are not “safe by default” (“security by obscurity”)

Permissive access leads to simple model theft

Sanity checks on model output before prescriptive action (“human-in-the-loop?”)

Logging and auditing required for much of security
Monitoring ML systems

Start simple

Training logs:
- Dataset versioning to include a hash of the datasets (has it changed?)
- Model versioning w/ hash (has it changed?)
- Model predictions on training and validation data (has it changed?)

Inference telemetry logs:
- When? <timestamp>
- What? <model_id>
- Who? <user_id> or <client_ip>
- What? <input hash>, <output score/label>
Data can tell you a lot
Takeaways

- ML risk outpacing ML security
- Today, traditional security practices are the most important element to securing ML
  - Access Control
  - Training Data permissions
  - Logging and Auditing
- Next steps: How will you respond?
  - How will you know when your model is attacked?
  - How will you ascertain root cause of an adversarial failure mode?
  - When your model is exploited, what is the “blast radius”? How can you reduce it?
- If you think like an adversary, you are well on your way
ML security is an infosec problem

With any other software system, you’d

→ Keep your user’s data private and protected
→ Audit software dependencies for vulnerabilities
→ Validate your inputs
→ Keep source code and other IP secure
Thank you!

Hyrum Anderson, PhD
Principal Architect
Azure Trustworthy Machine Learning

hyruma@microsoft.com