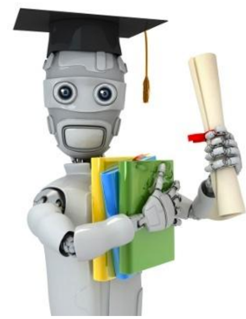
Summoning Demons: The Pursuit of Exploitable Bugs in Machine Learning

Rock Stevens, **Octavian Suciu**, Andrew Ruef,
Sanghyun Hong, Michael Hicks, Tudor Dumitras

University of Maryland

How can ML be Subverted?

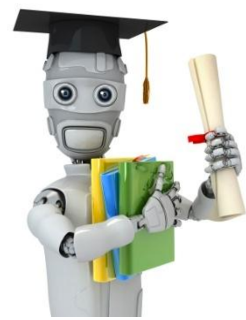


src: Coursera

Panda

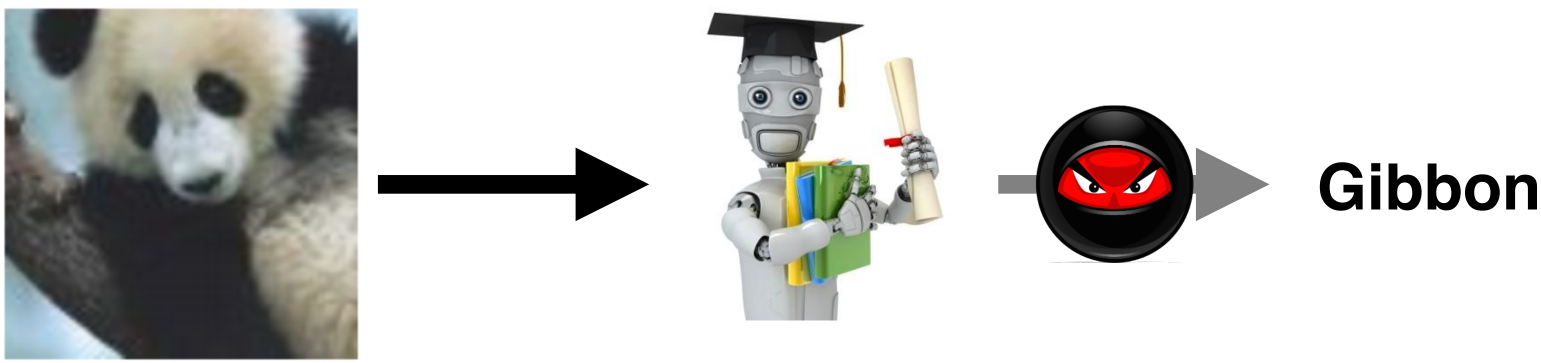
How can ML be Subverted?

src: Veracode



Gibbon

Exploiting the Underlying System

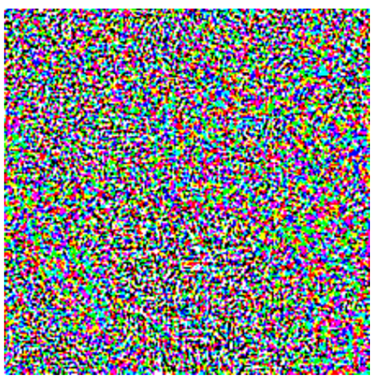


Attackers controlling the underlying system can dictate the output of ML systems

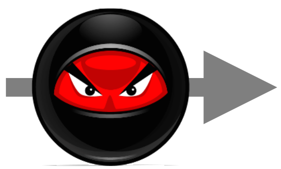
Adversarial Machine Learning

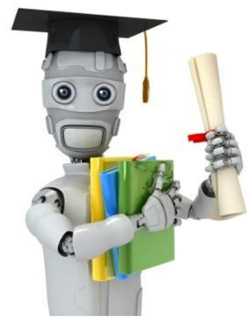
x

+



$sign(\nabla_x J(\Theta,x,y))$





$x+\varepsilon sign(\nabla_x J(\Theta,x,y))$

Gibbon

Adversarial sample crafting exploits the decision boundary:

- bypassing it (evasion)
- modifying it (poisoning)

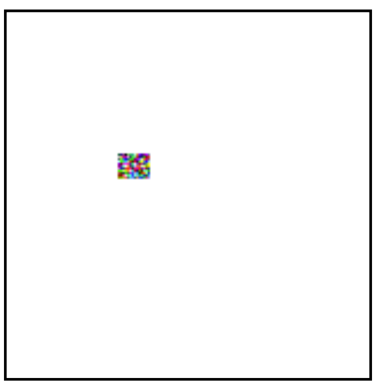
MARYLAND CYBERSECURITY CENTER

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv:1412.6572.

Exploiting the Implementation

x

+



<exploit>

src: National Geographic

Gibbon

Can attackers exploit the implementation in order to control the output of predictors?

Problem

- Attackers can craft inputs that exploit the implementation of ML algorithms
  - As opposed to perturbing the decision boundary of correct implementation
- These *logical* errors cause implementation to diverge from algorithm specification
  - Execution terminates prematurely or follows unintended code branches; memory content changes
- Exploits have no visible effects on system functionality
  - Existing defense tools are not designed to detect these errors

Research Questions

- Can we map attack vectors to ML architectures?
- Can we discover exploitable ML vulnerabilities systematically?
- Can we asses the magnitude of the threat?

Outline

- Attack Vector Mapping
- Discovery Methods
- Preliminary Results
- Conclusions

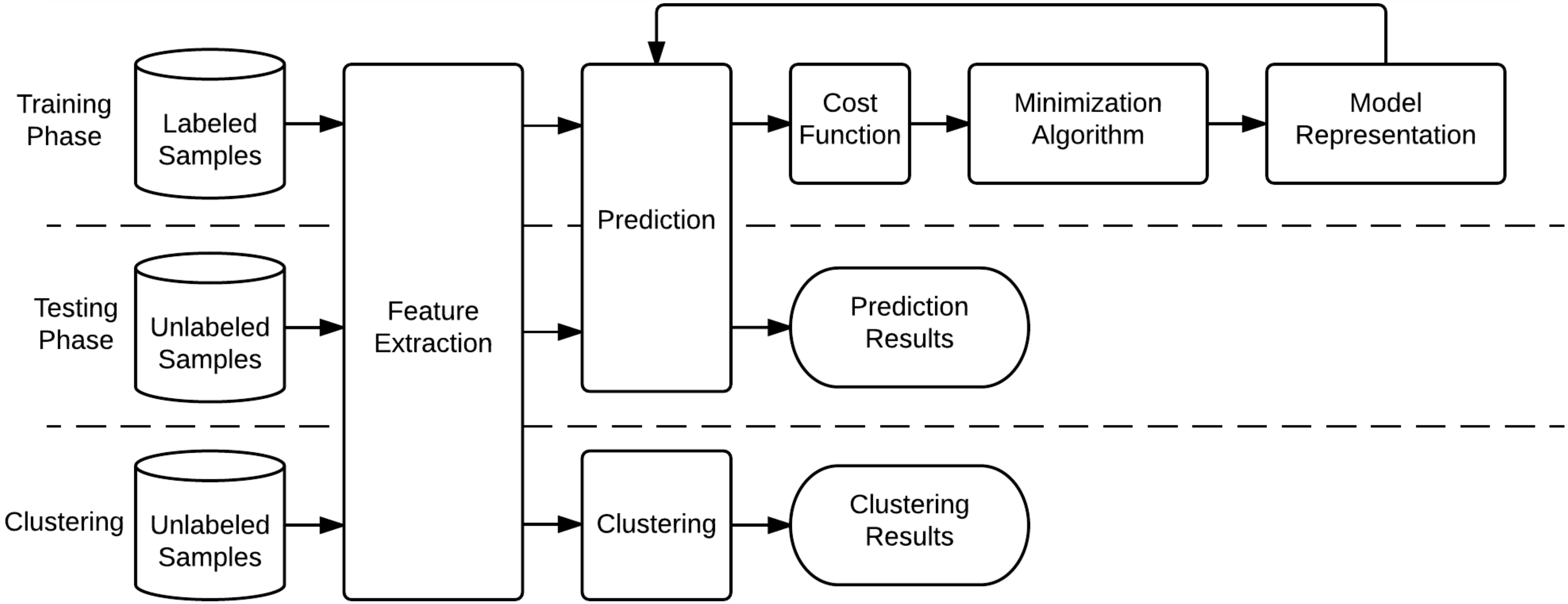
Impact of Exploits

attacker benefit

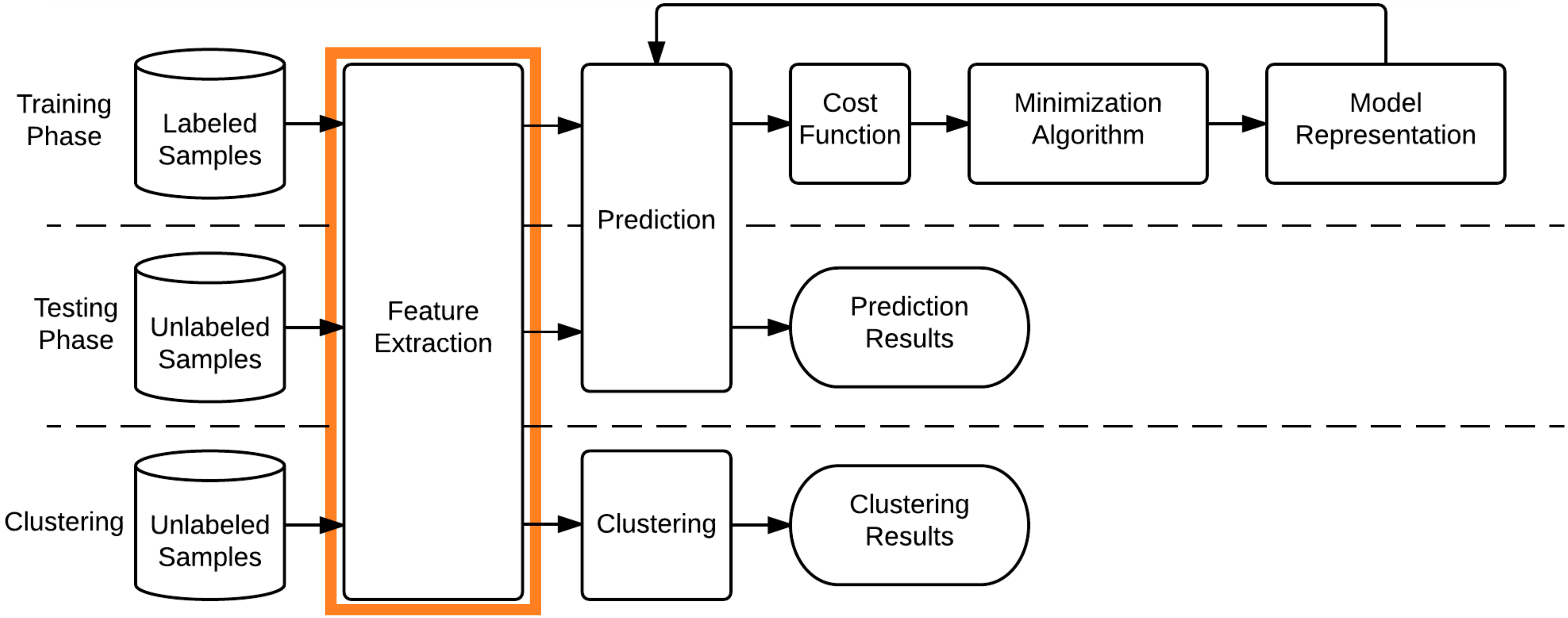
Poisoning, Evasion, Misclustering	Denial of Service (DoS)

Code Execution

Attack Surface

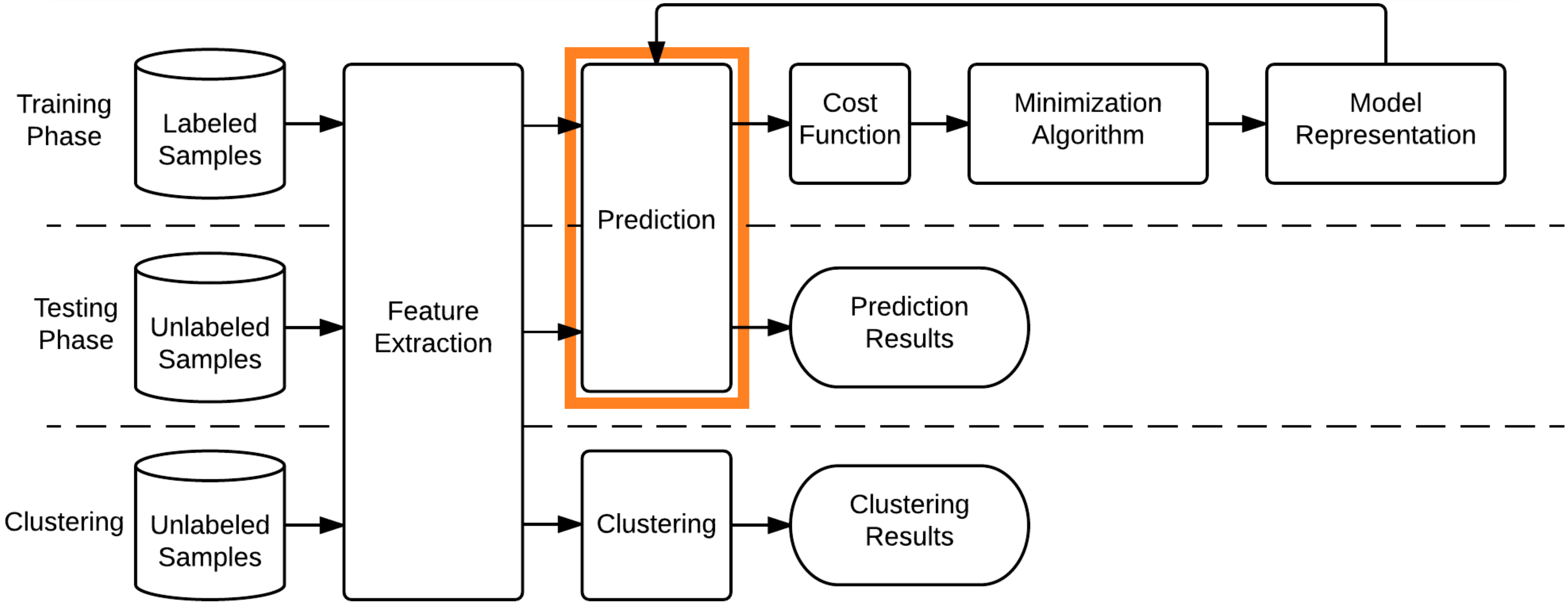


Attacking Feature Extraction (FE)



Insufficient integrity checks → Poisoning / Evasion / Misclustering
DoS
Code Execution

Attacking Prediction

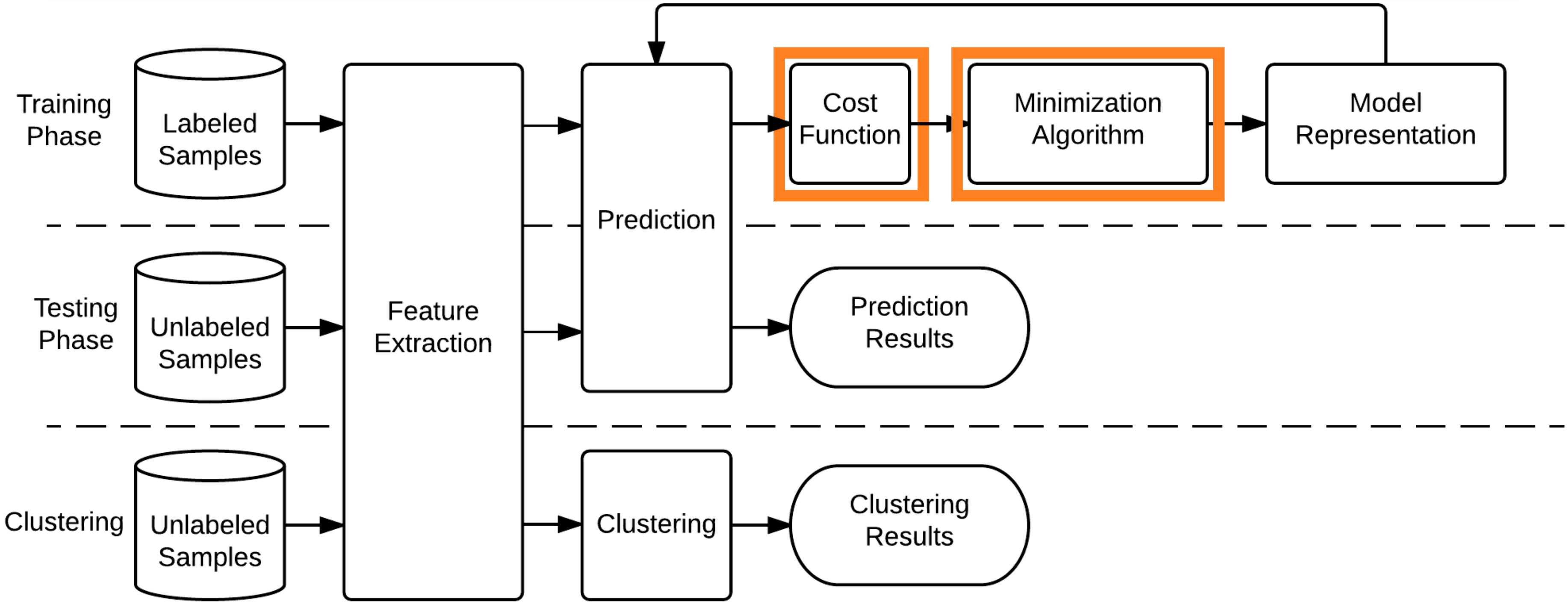


Overflow / Underflow
NaN
Loss of Precision

→ Poisoning / Evasion

MARYLAND CYBERSECURITY CENTER

Attacking Training

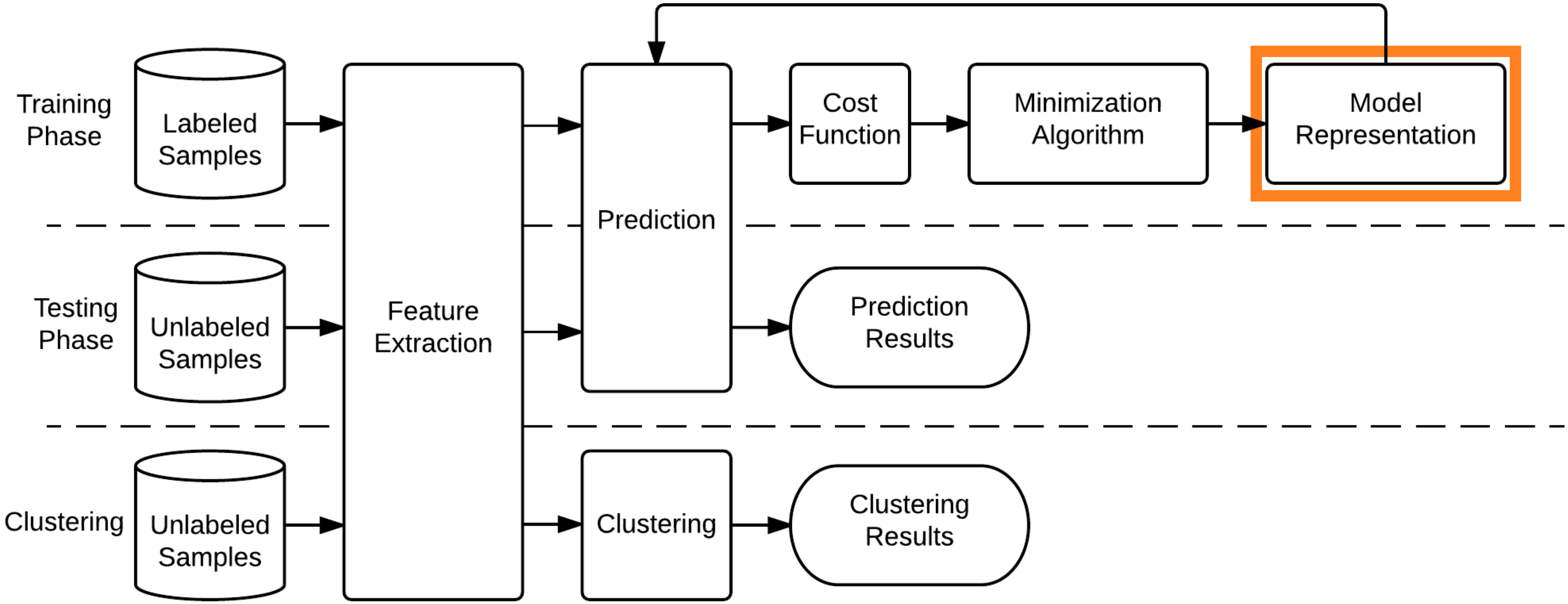


Overflow / Underflow
NaN
Loss of Precision

→

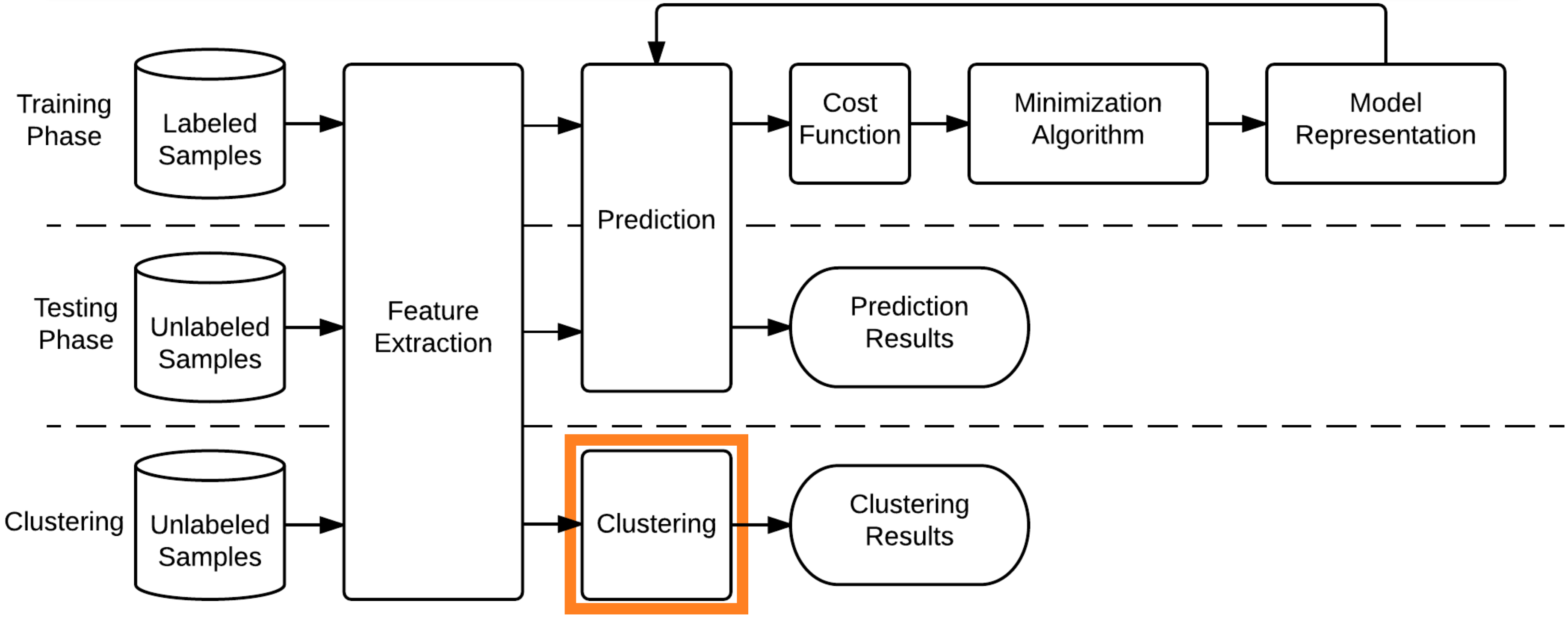
Poisoning
DoS

Attacking Model Representation



Loss of Precision → Poisoning / Evasion

Attacking Clustering



Overflow / Underflow
NaN
Loss of Precision

→ Misclustering

Outline

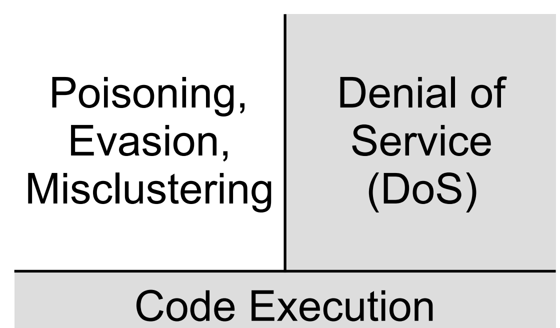
- Attack Vector Mapping
- Discovery Methods
- Preliminary Results
- Conclusions

Fuzzing[1]

- Testing tool used for discovering application crashes indicative of memory corruption

- Mutates input by flipping bits and serving it to the program under test

- American Fuzzy Lop[2]: tries to maximize code coverage, favoring inputs that result in different branches



1 - Miller, B.P., Fredriksen, L. and So, B., 1990. An empirical study of the reliability of UNIX utilities.

2 - http://lcamtuf.coredump.cx/afl/

MARYLAND CYBERSECURITY CENTER

Steered Fuzzing

- Find decision points in ML implementations that could be vulnerable
- Set failure conditions to the desired impact (e.g. evasion)

```
if failure_condition then:
   crash_program()
end if
```

<table>
<tr><td>Poisoning,
Evasion,
Misclustering</td><td>Denial of
Service
(DoS)</td></tr>
<tr><td colspan="2">Code Execution</td></tr>
</table>

MARYLAND CYBERSECURITY CENTER

Outline

- Attack Vector Mapping
- Discovery Methods
- Preliminary Results
- Conclusions

Targeted Applications

- OpenCV
  - Computer vision library
- Malheur
  - Malware clustering tool

Bugs in OpenCV

CVE-ID	Vulnerability	Impact
2016-1516	Heap Corruption in FE	Code Execution
2016-1517	Heap Corruption in FE	DoS
n/a	Inconsistent rendering in FE	Evasion

Bugs in OpenCV

CVE-ID	Vulnerability	Impact
2016-1516	Heap Corruption in FE	Code Execution
2016-1517	Heap Corruption in FE	DoS
n/a	Inconsistent rendering in FE	Evasion

Vulnerabilities allow access to illegal memory locations

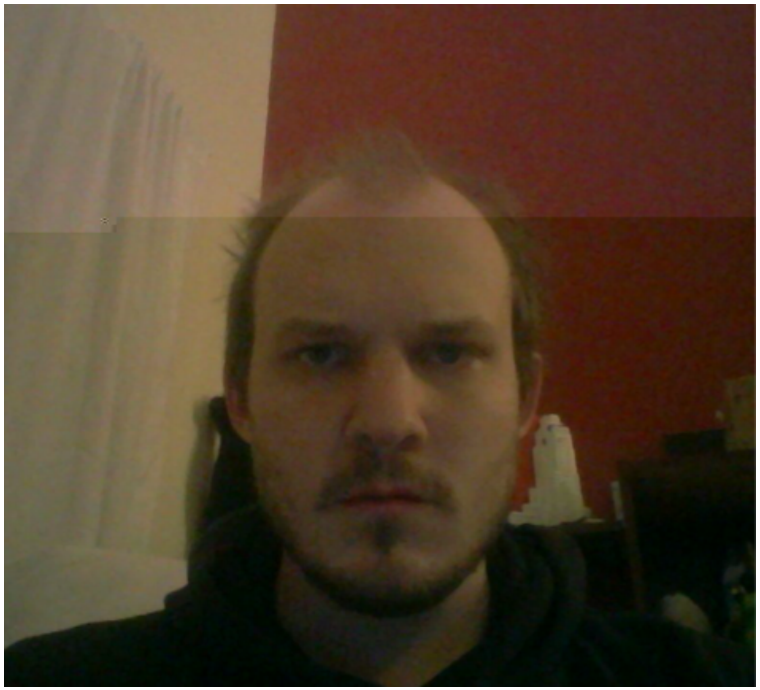
Bugs in OpenCV

CVE-ID	Vulnerability	Impact
2016-1516	Heap Corruption in FE	Code Execution
2016-1517	Heap Corruption in FE	DoS
n/a	Inconsistent rendering in FE	Evasion

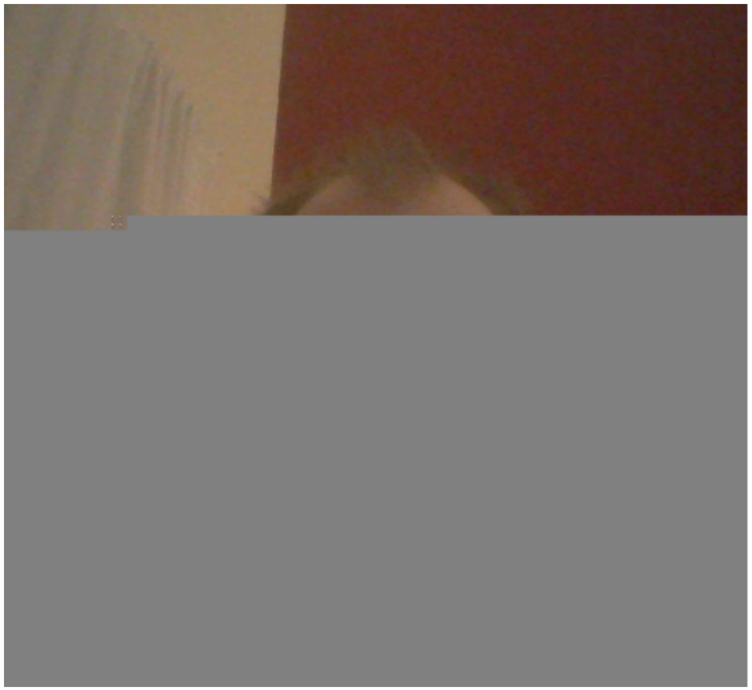
Vulnerability allows legitimate input to bypass facial detection

Attack requires no queries to the model!

Facial Detection Evasion Example



Rendering mutated image using Adobe Photoshop



Rendering mutated image using Preview

More Evasion Examples

src: Imgur

src: Imgur

Bugs in Malheur

CVE-ID	Vulnerability	Impact
2016-1541	Heap Corruption in FE	Code Execution
n/a	Heap Corruption in FE	Misclustering
n/a	Loss of precision in Clustering	Misclustering

MARYLAND
CYBERSECURITY CENTER

Bugs in Malheur

CVE-ID	Vulnerability	Impact
2016-1541	Heap Corruption in FE	Code Execution
n/a	Heap Corruption in FE	Misclustering
n/a	Loss of precision in Clustering	Misclustering

Vulnerabilities in underlying *libarchive* library affects every version of Linux and OS X

Bugs in Malheur

CVE-ID	Vulnerability	Impact
2016-1541	Heap Corruption in FE	Code Execution
n/a	Heap Corruption in FE	Misclustering
n/a	Loss of precision in Clustering	Misclustering

Additional Malheur vulnerability triggered by the one in libarchive

Attack can manipulate memory representation of inputs they do not control!

Bugs in Malheur

CVE-ID	Vulnerability	Impact
2016-1541	Heap Corruption in FE	Code Execution
n/a	Heap Corruption in FE	Misclustering
n/a	Loss of precision in Clustering	Misclustering

Casting *double* to *float* when computing L1 & L2 norms

Results Summary

- Bugs in ML implementations represent a new attack vector
  - Disclosed 5 exploitable vulnerabilities in 2 systems, many of which were marked as WONTFIX
  - Response after reporting code execution vulnerability: *"Although security and safety is one of important aspect of software, currently it's not among our top priorities"*

- Threat model also applicable outside the scope of ML
  - Any application that ingests uncurated inputs might be vulnerable

Outline

- Attack Vector Mapping
- Discovery Methods
- Preliminary Results
- Conclusions

Conclusions

- Can we map attack vectors to ML architectures?
  - Presented a baseline architecture and vector mapping
  - Future: need an attack taxonomy, unification with AML
- Can we discover exploitable ML vulnerabilities systematically?
  - Steered fuzzing for semi-automatic discovery
  - Future: automatic techniques designed specifically for ML
- Can we asses the magnitude of the threat?
  - Discovered exploitable vulnerabilities in real-world systems
  - Future: asses the adversarial gain, compare to other exploitation techniques

Thank you!

Octavian Suciu

osuciu@umiacs.umd.edu