Advances in Cloud-Scale Machine Learning for Cyberdefense

Mark Russinovich
CTO, Microsoft Azure
@markrussinovich
ML and AI in all Microsoft Products

- Cortana Intelligence Suite
- SQL Server + R
- Microsoft R Server
- Hadoop + R
- Spark + R
- Microsoft CNTK
- Azure Machine Learning
- R Tools/Python Tools for Visual Studio
- Azure Notebooks (JuPyTer)
- Cognitive Services
- Bot Framework
- Cortana
- Office 365
- HoloLens
- Bing
- Skype
- Xbox 360
- Dynamics 365
Microsoft’s daily cloud security scale

- 10s of PBs of logs
- 300+ million active Microsoft Account users
- 1.3+ billion Azure Active Directory logons
- 1.5 million compromise attempts deflected
- Detected/ reflected attacks >10,000 location-detected attacks
WHAT IS ATTACK DISRUPTION?
Red Team Kill Chain

Recon  Delivery  Foothold  Persist  Move  Elevate  Exfiltrate
Blue Team - Kill Chain for Attack Detection
Two roadblocks for Attack Disruption

False Positives

Manual Triage
False Positives

Lose ability to **triage**
False Positives **FACT**

You *cannot* salvage a false positive with just Visualization. You need better solutions.
False Positives

Evolution of security detection techniques

**TRADITIONAL PROGRAMMING**

- Data → Output
- Program/Rules → Output

Hand-crafted rules by security professionals

Con: Rules are static, and don’t change with changes in environment => False Positives!

**MACHINE LEARNING**

- Data → Program
- Output/Labels → Program

System adapts to changes in environment as new data is provided, and re-trained

Our *supervised learning* approach enables detection **without generating many FPs**
False Positives

For supervised learning, Azure gets labeled data through:

- Domain experts, customers who provide feedback from Alerts
- Labels from other product groups (including O365, Windows Seville)
- Surgical Red team exercises (OneHunt)
- Automated Attack bots
- Bug Bounty
- MSRC
False Positives  Manual Triage
For Attack Disruption, we need to think beyond detection

Manual Triage

Everything is manual, with little or no intelligence – We need to change this
Properties of a Successful Machine Learning Solution

Adaptable
Successful Detection
Explainable
Actionable
Adaptable in Cloud is Difficult

Why?

Evolving Landscape
- Frequent deployments
- New services coming online
- Usage spikes

Evolving Attacks
- Constantly changing environments leads to constantly changing attacks
  - New services
  - New features for existing services
Explainability

Why?

Surfacing a security event to an end-user can be useless if there is no explanation. Explainability of results should be considered at earliest possible stage of development.

Results without explanation are hard to interpret. Best detection signal with no explanation might be dismissed/overlooked.

<Example – How do you explain this to an analyst>

<table>
<thead>
<tr>
<th>UserId</th>
<th>Time</th>
<th>EventId</th>
<th>Feature1</th>
<th>Feature2</th>
<th>Feature3</th>
<th>Feature4</th>
<th>...</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a4b43</td>
<td>2016-09-01 02:01</td>
<td>4688</td>
<td>0.3</td>
<td>0.12</td>
<td>3.9</td>
<td>20</td>
<td>...</td>
<td>0.2</td>
</tr>
<tr>
<td>73d87a</td>
<td>2016-09-01 03:15</td>
<td>4985</td>
<td>0.4</td>
<td>0.8</td>
<td>0</td>
<td>11</td>
<td>...</td>
<td>0.09</td>
</tr>
<tr>
<td>9ca231</td>
<td>2016-09-01 05:10</td>
<td>4624</td>
<td>0.8</td>
<td>0.34</td>
<td>9.2</td>
<td>7</td>
<td>...</td>
<td>0.9</td>
</tr>
<tr>
<td>5e9123</td>
<td>2016-09-01 05:32</td>
<td>4489</td>
<td>2.5</td>
<td>0.85</td>
<td>7.6</td>
<td>2.1</td>
<td>...</td>
<td>0.7</td>
</tr>
<tr>
<td>1e6a7b</td>
<td>2016-09-01 09:12</td>
<td>4688</td>
<td>3.1</td>
<td>0.83</td>
<td>3.6</td>
<td>6.2</td>
<td>...</td>
<td>0.1</td>
</tr>
<tr>
<td>33d693</td>
<td>2016-09-01 14:43</td>
<td>4688</td>
<td>4.1</td>
<td>0.63</td>
<td>4.7</td>
<td>5.1</td>
<td>...</td>
<td>0.019</td>
</tr>
<tr>
<td>7152f3</td>
<td>2016-09-01 19:11</td>
<td>4688</td>
<td>2.7</td>
<td>0.46</td>
<td>3.9</td>
<td>1.4</td>
<td>...</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Actionable Detections

Detections must result in downstream action

Good explanation without being actionable is of little value

EXAMPLES

• Policy decisions
• Reset user password
Framework for a Successful Detection

Successful Detections incorporate domain knowledge through disparate datasets and rules.
Case Study 1
Successful detection through combining disparate datasets

**PROBLEM STATEMENT**
Detect compromised VMs in Azure

**HYPOTHESIS**
If the VM is sending spam, then it is most likely compromised.

**SOLUTION**
Use supervised Machine Learning to leverage Labeled spam data from Office365 and combine with IPFIX data from Azure.
Case Study 1

Technique Overview

IPFIX data

6,569 spam labeled IPFIX data

Benign IPFIX data

Machine Learning

Automated Compromise Detection

New Case
Case Study 1

Machine Learning Deep Dive

Input data for 1st iteration

Weak learner at 1st iteration

Results
Case Study 1

Machine Learning Deep Dive

The data points that were incorrectly categorized by the weak learner in the first iteration (the positive examples) are now weighted more.

Simultaneously, the correct points are down weighted.
Case Study 1

Machine Learning Deep Dive

The data points that were incorrectly categorized in the second iteration (the negative examples) are now weighted more.

Simultaneously, the correct points are down weighted.

Final result is a combination of learners from each iteration
Case Study 1

Model Performance and Productization

Model trained in regular intervals

- Size of data: 360GB per day
- Completed within minutes

Classification runs multiple times a day

- Completed within seconds

<table>
<thead>
<tr>
<th>Dataset</th>
<th>True Positive Rate</th>
<th>False Positive Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only using Azure IPFIX data</td>
<td>55%</td>
<td>1%</td>
</tr>
<tr>
<td>Using Azure IPFIX and O365 data</td>
<td>81%</td>
<td>1%</td>
</tr>
</tbody>
</table>

26% resulting improvement
Case Study 2
Successful detection through combining rules and machine learning

**PROBLEM STATEMENT**
Rule based malware detection places hard constraints if something is a malware or not. While they are specific, they are also noisy and have a lot of False negatives.

**HYPOTHESIS**
Can we combine the hard logic of rule based detections with the soft-logic of machine learning systems?

**SOLUTION**
Build two ML models:
1) Model 1 that baselines malware behavior
2) Model 2 that incorporates rules as features
Combine result of two models
Case Study 2

MALWARE DETECTION BACKGROUND
ATP Architecture

Conventional A/V
Defender, Kaspersky, Cyren

Detonation Chamber
Spin up multiple VMs
Multiple OS and Office versions
Instrument attachment behavior

Safelinks
Protects against malicious URLs in Real Time (on click)
Case Study 2

Technique Overview

**PRE-ANALYSIS**
- Hash/Fuzzy Hash
- PE Analyzer
- File Type Analyzer
- PhotoSimilarity
- ...

**DETONATION**
- SysMon
- ETW Logger
- API hooks
- Crash dump
- ...

**POST-ANALYSIS**
- YARA
- Threat Intel
- Network Analysis
- Macro Evidence
- ...

**Technique Overview Diagram**

- **Fingerprint Model**
- **Behavioral Model**
- **Combiner**
- **Combined Verdict**

BlueHat IL 2017
Dataset

(SAMPLE)
Case Study 2

Machine Learning Deep Dive: **Fingerprint Model**

Information gets more granular

<table>
<thead>
<tr>
<th>Call Order</th>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
<th>Level 4</th>
<th>Level 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process</td>
<td>LoadImage</td>
<td>SYSTEM .exe</td>
<td><em>IMSFTHISTORY!</em></td>
<td>wscript</td>
</tr>
<tr>
<td>2</td>
<td>Api</td>
<td>CallFunction</td>
<td>CreateMutexA</td>
<td>!_ETId!Mutex</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Api</td>
<td>CallFunction</td>
<td>CreateMutexW</td>
<td>wscript_rasapi32</td>
<td>EnableTracing</td>
</tr>
<tr>
<td>4</td>
<td>Registry</td>
<td>SetRegValue</td>
<td>Tracing</td>
<td>internet settings</td>
<td>ProxyBypass</td>
</tr>
<tr>
<td>5</td>
<td>Registry</td>
<td>DeleteRegValue</td>
<td>InternetOption</td>
<td>internet settings</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Process</td>
<td>CreateProcess</td>
<td>NOT_SANDBOX_CHECK</td>
<td>LaunchedViaCom</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Network</td>
<td>AccessNetwork</td>
<td>Wininet_Getaddrinfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Api</td>
<td>CallFunction</td>
<td>CreateMutexW</td>
<td>RANDOM_STR</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Network</td>
<td>ResolveHost</td>
<td>piglyeleutqq.com</td>
<td>UNKNOWN</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Api</td>
<td>CallFunction</td>
<td>Connect</td>
<td>UNKNOWN</td>
<td></td>
</tr>
</tbody>
</table>
Machine Learning Deep Dive: Behavioral Model

Incorporates domain knowledge into the model in the form of YARA rules

Source of features
- YARA rules
- Static analysis
- Aggregates from Data:
  - Registry keys/values that are changed/created/deleted
  - Mutexes created
  - Number of spawn processes per process detail info

The model works well to detect new types of malware
Case Study 2

Model Performance and Productization

Model trained in regular intervals
Size of data: 270GB per day
Completed within minutes

Classification runs multiple times a day
Completed within milliseconds

<table>
<thead>
<tr>
<th>Dataset</th>
<th>True Positive Rate</th>
<th>False Positive Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARA rules only</td>
<td>82.6%</td>
<td>0.0178%</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>93.6%</td>
<td>0.0127%</td>
</tr>
<tr>
<td>Model 1 + Model 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10 points improvement
For Attack Disruption, We Need to Think Beyond Detection

Everything is manual, with little or no intelligence – We need to change this
Triage incidents, not alerts

**Anomalous DLL:** rundll32.exe launched as sposql11 on CFE110095
- alert type
- process
- user
- host

**New process uploading:** rundll32.exe to 40.114.40.133 on CFE110095
- alert type
- process
- remote host
- host

**Large transfer:** 50MB to 40.114.40.133 from sqlagent.exe on SQL11006
- alert type
- remote host
- process
- host
Triage incidents, not alerts
Conclusion

Attack Disruption means to shorten blue team kill chain

- **Speed**: Real-time detection
- **Quality**: Reduce false positives
- **React**: Fast triage
Attack Disruption Checklist

- Data with different datasets
- Scalable ML solution and expertise
- Secured platform
- Eyes on Glass

Example Azure services you can leverage:

- Azure Event Hubs
- Azure Machine Learning
- Azure Data Lake
Thank you