Making SIEM great again – Augmenting your detection via simple machine learning

Elaine Hung

Whoami

Threat Hunter/
DFIR Practitioners

Machine Learning as a hobby

Blogging at
https://medium.com/@toffeebr33k

DISCLAIMER - All views expressed are of my own and do not represent the opinions of any entity whatsoever with which I have been, am now or will be affiliated.

It all starts with a red team exercise……

THE ATTACKER

- Entered the environment **via provided credentials** in other regions
- **Credential dumping**
- **Escalated to domain admin**
- Moved to **a few critical applications**

THE RESPONDER

- **Alerts Fatigues**
- **Commercial network probes were not detecting anything**

Any QuickWin to enhance detection capabilities?

Machine Learning Use Cases

2 Scenarios to profile user behaviour ……..

User Access Anomaly

- System access
- Application access

User Activity Profiling

- CommandLine Usage
- Software installation
- Process creation

Give you some ideas on how you could leverage simple machine learning to enhance your threat hunting capabilities…..

How does it work?

- System Event Logs
- Application Event Logs
- Physical Access Logs
- Network Log

Log Analytic Platform

Transformed Events

Results via SCP

Adding Python scripts to run in your SIEM on a regular basis

python™

scikit learn

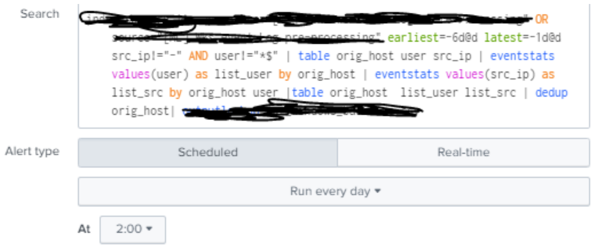
Machine Learning with Scikit-Learn

The Learning Cycle

Frame the problem → Explore your data → Feature Generation → Explore, Train and Tune Model → Pilot Operation → Run and Continuous Improvement

Frame the problem

Identify use cases you want to resolve with Machine Learning e.g. Detect Login Anomaly



Explore your data

Collect all relevant dataset in a regular period of time

Feature Generation

Transform data into numerical value e.g. Count of Login, variance of the time

Explore, Train and Tune Model

With known dataset,

- go for supervised approach as a specific attack detection.
- Using unsupervised approach for anomaly detection.
- Train the model

Pilot Operation

- Posing as a pilot ongoing dashboard/alert for SOC analyst to evaluate and monitor.
- Provide watchlist for fine tuning and re-train of models

Run and Continuous Improvement

- Posing as a regular alerting

Use Case 1 – User Access Anomaly

User access anomaly – Concept

Concept: Develop a point-based system to evaluate how "abnormal" the login patterns in by aggregating different machine learning models

Anomaly Score for each user Login Access

Drill Down and Investigate

Correlate with other SIEM alerts

Unsupervised Learning

Learning and Grouping of the Login pattern of each user in the last 30 days

Outliner – Any login pattern different from others?

Supervised Learning

Provide Dataset labelled with login performed by red team

Flagging Login pattern same as red team

References: [Augment your Security Monitoring Use Cases with MLTK's Machine Learning]

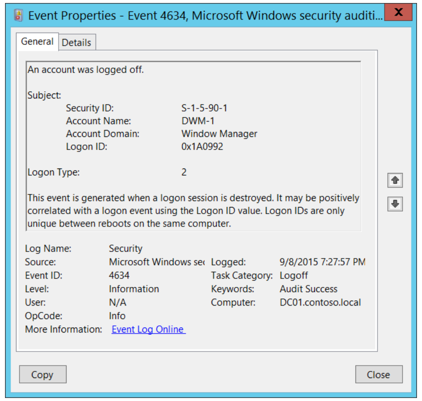
Icons made by Vectors Market from Flaticon

User access anomaly – Unsupervised

Anomaly Detection - Unsupervised

Step 1: Collection of Events

- Collection of Windows Event Log 4624 per 24 hours into a central repository – successful login attempts
- The following features are collected:
  - Source IP address
  - Timestamp
  - Host
  - User
  - LogonType



Step 2: Feature Generation

Transform the collected events into the following features every 24 hours

By User/24 hour

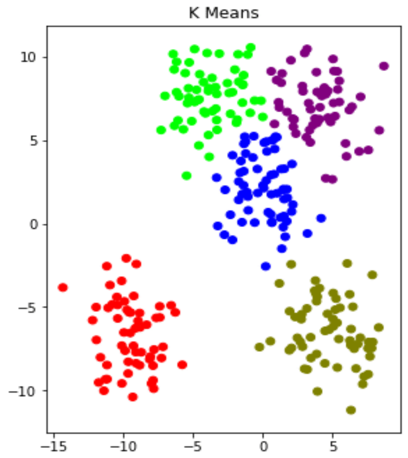
- **Count of Login During Office Hour**
- **Count of Login During Non-Office Hour**
- **Count of Source IP Address**
- **Count of Total Login**
- **Count of Logon Type**

User access anomaly – Unsupervised

Anomaly Detection - Unsupervised

Step 3: Model Fitting

- Cluster Training Data **using KMeans**
- **One model each for Source IP, LogonType and Host, Office hour login and non-office hour login**
- TimeFrame: last [30:1] days of features on a daily basis. This is served as a training.
- Highlight any users with changes in **Kmeans** Membership



user	_time	c_host	c_src_ip	c_Logon
	Day 1	2	2	2
	Day 2	2	2	2

*Assumption: the Login pattern of each user will be similar on a daily basis. Therefore, the group membership clustered is expected to be the same/similar.

User access anomaly – Unsupervised

Anomaly Detection - Unsupervised

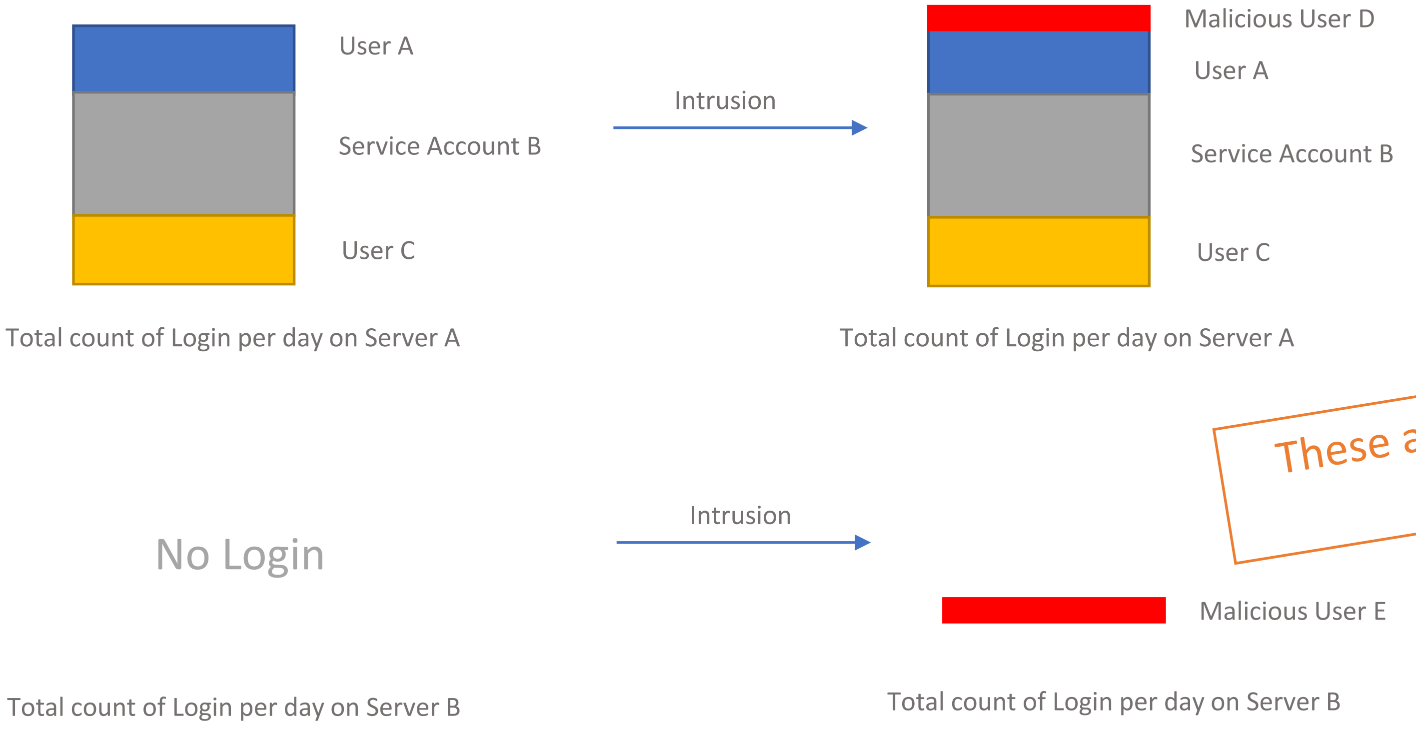
Step 4: Result Delivered by the Model

- Detection of irregular user login - Outliner on:
  - **Non-office hour successful attempts**
  - **Office hour successful attempts**
  - **No. of host logged in**
  - **Logon Type**

_time	reason	user	den_x1	den_x2	den_x3	den_cluster_distance	df_non_office	df_office
	Outliner Non-Office Hour Login							
	Outliner Non-Office Hour Login							
	Outliner host							
	Outliner Non-Office Hour Login							
	Outliner Non-Office Hour Login							

User access anomaly – Supervised

Anomaly Detection - Supervised



IMPACT:

1. New User, New IP address to the server
2. Decrease in the percentage contribution of total count of login of existing user

These activities could not be captured by the previous unsupervised model

User access anomaly – Supervised

Anomaly Detection - Supervised

Step 1: Collection of Events

- Collection of Windows Event Log 4624 per 24 hours into a central repository
- The following features are collected:
  - Timestamp
  - Host
  - User
  - LogonType
  - Source IP address
- Generation of a daily lookup table that contains IP address and users that have been logged into each server in the past 30 days to formulate a baseline

Step 2: Feature Generation

Transform the collected events into the following features every 24 hours

By User/24 hour

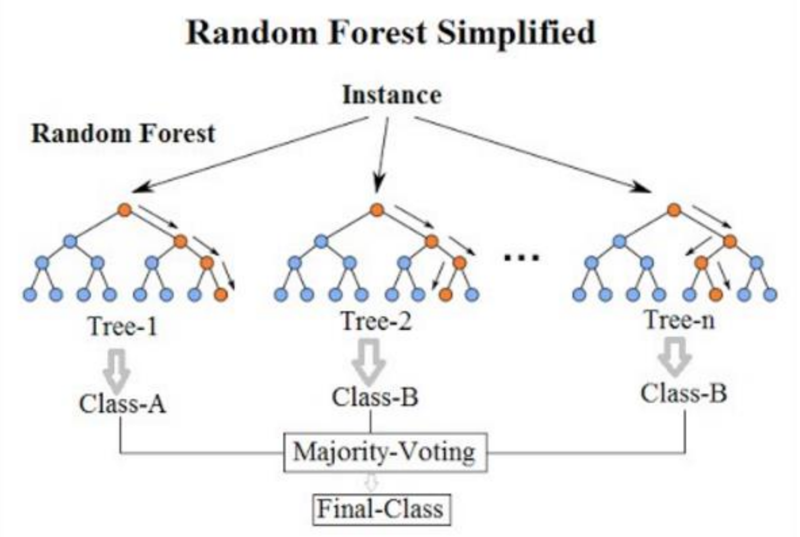
- **Isnewip [Whether the login IP address to the host is a newip by comparing with the baseline table]**
- **Isnewuser [Whether the user to the host is a newip by comparing with the baseline table]**
- **Percentage = Percentage of total count of each user login to the server on each day /Total number of login on the server on each day**
- **Total Count of login from the same source IP address**

User access anomaly – Supervised

Anomaly Detection - Supervised

Step 3: Model Fitting

- **Fit RandomForest Classifer**
- Use the previous red team exercise as a training/test dataset
- Labelling dataset [1= Malicious, 0 = Benign]



Putting the models together

Step 4: Result Delivered by the Model

- **Detection of irregular user login - Outliner on:**
  - **Non-office hour successful attempts**
  - **Office hour successful attempts**
  - **No. of host logged in**
  - **Logon Type**
  - **New User Login on a specific host**
  - **Irregular login proportion on the specific host**

- Result Dashboard - Identify users with high anomaly – For SOC analyst to review anything malicious

user	count	values(reason)
		Outliner host newip/newuser
		newip/newuser
		newip/newuser
		newip/newuser
		newip/newuser

Correlation to other threat events collected in SIEM

_time	search_name
	Permission Groups Discovery - Rule
	Permission Groups Discovery - Rule
	Detection of Command Line obfuscation - Rule

Extension to Application Access Log

- Same concept could be extended to Customised Application Access Log

Data Collection → Feature Generation → Model Fitting → Results

Data Collection	Feature Generation	Model Fitting	Results
Timestamp	*Count of total count of login compared to yesterday*	**Apply Probability Density Functions on each features**	Detection of non − office hour transaction
Host	*Count of total count of non-office hour login compared to yesterday*		$\textit{Anyone being} \frac{approver}{maker} \textit{at the same time}$
User	*Source IP address*		*New user to initiate the transaction*
User Role	*User role/activity performed*		*Action that have not been performed in 30 days*
Source IP address			

Extension to Application Access Log

- Transaction Monitoring - Anyone initiating transactions in irregular hours [Density Function]

Transaction Anomaly

_time	Address	EventCode	EventName	Merged	To_Status	Operator	reason
							Outliner Transaction Time

user	anomaly_score	values(reason)
	1	unusal browser IP address
	1	unusal browser IP address

Use Case 2 – User Activity Profiling

User activity anomaly – Concept

Concept: Develop a point-based evaluation system to evaluate any abnormal activities performed by a user

Anomaly Score for each workstations process activities → Correlate with other SIEM alerts

Malicious Software Installation

- Identify any outliner software installation

Obfuscated Commands

- Identify obfuscated commands called by workstations

Malicious Process Call

- Identify process created related to the use of LOLBAS bin
- Baseline past 30 days LOLBAS bin usage
- Identify outliner usage

Malicious software installation

Step 1: Collection of Events

- A script has been used to scan each software installed on each of the workstations
- The following features are collected:
  - Installation Date
  - Software Name
  - Publisher
  - User

Problems we met

- We do not have a centralized authorized lists.
- There are users with local administrators right and could have installed software on their own.
- Software name are inconsistent depends on the version name and the local language of computers

Malicious software installation

Step 2: Feature Generations – Word Embedding Techniques

- Use of **Term Frequency – Inverse Document Frequency (TFIDF)** on the software name

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$$

	Doc 1	Doc 2	...	Doc *n*
Term(s) 1	12	2	...	1
Term(s) 2	0	1	...	0
...	...	...	...	
Term(s) *n*	0	6	...	3

account_tfidf_12__	account_tfidf_13_a	account_tfidf_14_b	account_tfidf_15_c	account_tfidf_16_d
0.0	0.0	0.0	0.27282225077971983	0.0
0.0	0.0	0.0	0.27282225077971983	0.0
0.0	0.0	0.0	0.6065348164026177	0.0

TFIDF score of each of the top 100 characters identified.

Malicious software installation

Step 2: Feature Generations

- Character features extraction:
  - Shannon Entropy $H = -\sum p_i \log_b p_i$

    (how random the software name is)
  - Length of Software Name
  - Digit Ratio – No. of digit/Length of Software Name
  - Space Ratio – No. of Space/Length of Software Name
  - Vowel Ratio – No. of Vowel characters/Length of Software Name
  - Consonant Ratio – (1- digit ratio – vowel ratio)
  - Meaning Ratio – len(dictionary word)/len(SoftwareName)

    (to identify how likely the software name is dictionary word)

```
|`ut_shannon(CommandLine)`
| `ut_meaning(CommandLine)`
| eval ut_digit_ratio = 0.0
| eval ut_vowel_ratio = 0.0
| eval ut_command_length = max(1,len(CommandLine))
| rex field=CommandLine max_match=0 "(?<digits>\d)"
| rex field=CommandLine max_match=0 "(?<vowels>[aeiou])"
| rex field=CommandLine max_match=0 "(?<space>\s)"
| eval ut_digit_ratio=if(isnull(digits),0.0,mvcount(digits) / ut_command_length)
| eval ut_vowel_ratio=if(isnull(vowels),0.0,mvcount(vowels) / ut_command_length)
| eval ut_space_ratio=if(isnull(space),0.0,mvcount(space) / ut_command_length)
| eval ut_consonant_ratio = max(0.0, 1.000000 - ut_digit_ratio - ut_vowel_ratio) |  fields - digits - vowels - space |
```

ut_command_length	ut_consonant_ratio	ut_digit_ratio	ut_meaning_ratio	ut_shannon	ut_space_ratio	ut_vowel_ratio
120	0.625000	0.21666666666666667	0.35833333333333334	5.233496795710055	0.025	0.15833333333333333
81	0.654321	0.25925925925925924	0.30864197530864196	5.0962814108698	0.012345679012345678	0.08641975308641975
61	0.803279	0.03278688524590164	0.45901639344262296	4.8077292479888225	0.08196721311475409	0.16393442622950818
56	0.857143	0.03571428571428571	0.39285714285714285	4.5683535294637805	0.08928571428571429	0.10714285714285714

Malicious software installation

Step 3: Train the Model

- Collected a list of installed software for the last 3 months as a training dataset. Conduct an Eyeball checking to identify legit/malicious data and label the dataset.
- Train it with **Random Forest Classifer :**
  - Result from TFIDF
  - Character Feature Extraction

Step 4: Result Delivered by the Model

_time	software	publisher	hostname	user	reason
					Software outliner
					Software outliner

Obfuscated commands

Step 1: Collection of Events

- Collection of commandline generated from Windows Event Log 4688 – Benign data
- Use of Invoke-DOSfuscation dataset as the malicious data https://github.com/danielbohannon/Invoke-DOSfuscation

Step 2 and 3 : Feature Generations and model training

- Same approach as previous example

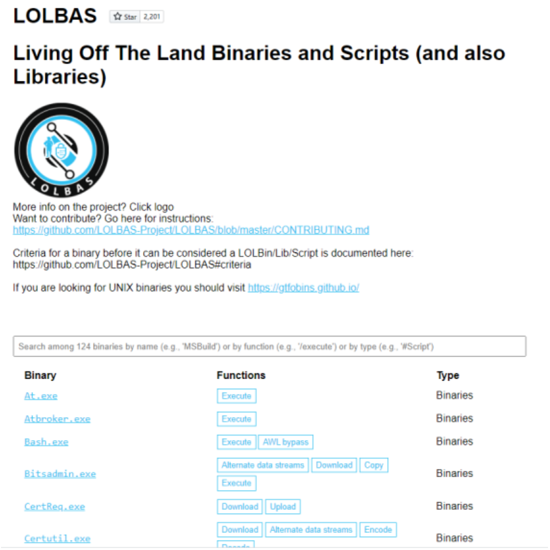
Step 4: Result Delivered by the Model

orig_host	user	ParentProcessName	NewProcessName	CommandLine	reason
	-	C:\Windows\System32\cmd.exe		PsExec PsExec64	Outliner Command Line - possible obfuscated command line used

Malicious LOLBAS usage

What is LOLBAS ….

Living Off the Land Binaries and Scripts (LOLBAS), i.e. scripts and binaries normally installed by default in Microsoft Windows. Attackers and pen testers could leverage LOLBAS functionalities potentially allowing for compromise of the target system.



LOLBAS Star 2,201

Living Off The Land Binaries and Scripts (and also Libraries)

More info on the project? Click logo
Want to contribute? Go here for instructions:
https://github.com/LOLBAS-Project/LOLBAS/blob/master/CONTRIBUTING.md

Criteria for a binary before it can be considered a LOLBin/Lib/Script is documented here:
https://github.com/LOLBAS-Project/LOLBAS#criteria

If you are looking for UNIX binaries you should visit https://gtfobins.github.io/

Search among 124 binaries by name (e.g., 'MSBuild') or by function (e.g., '/execute') or by type (e.g., '#Script')

Binary	Functions	Type
At.exe	Execute	Binaries
Atbroker.exe	Execute	Binaries
Bash.exe	Execute, AWL bypass	Binaries
Bitsadmin.exe	Alternate data streams, Download, Copy, Execute	Binaries
CertReq.exe	Download, Upload	Binaries
Certutil.exe	Download, Alternate data streams, Encode	Binaries

https://lolbas-project.github.io/

Malicious LOLBAS usage

Step 1: Collection of Events

- Windows Event Log 4688 – Process Creation
- Filter out process related to LOLBAS Bin and script

_time	user	hostname	ParentProcessName	NewProcessName	CommandLine
			regedit	regedit	
			regedit	regedit	
			regedit	regedit	

- Collection of 30 days data as a baseline

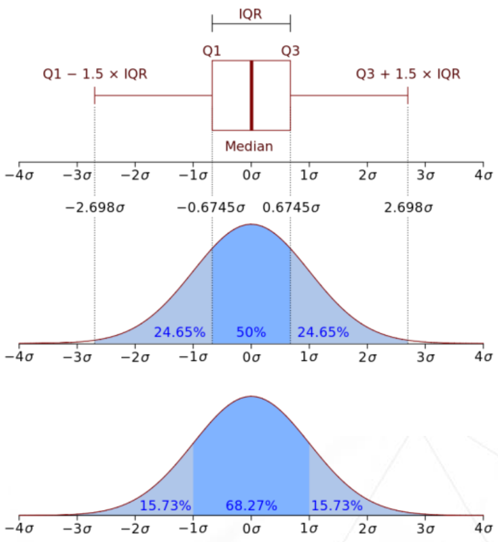
Step 2: Feature generations

- Count of No. of LOLBAS Bin called in a day per hostname

Malicious LOLBAS usage

Step 3: Model Fitting

- **apply Probability Density Functions on** Count of No. of LOLBAS Bin called on each host per 30 days data



Step 4: Result Delivered by the Model

LOLBAS Alert

_time	hostname	lolbas	reason
		csc cscript findstr msiexec netsh reg regsvr32 rundll32 sc wscript	LOLBAS Outliner
		bash control csc cscript gpscript msiexec netsh reg regedit rundll32 sc wscript	LOLBAS Outliner
		bash csc findstr gpscript msiexec netsh reg regedit regsvr32 rundll32 sc	LOLBAS Outliner
		control csc cscript gpscript mavinject netsh reg regsvr32 rundll32 sc wscript	LOLBAS Outliner
		reg	LOLBAS Outliner
		cscript explorer findstr gpscript msiexec netsh reg regsvr32 rundll32 sc wscript	LOLBAS Outliner
		cscript findstr gpscript mavinject msiexec netsh reg regsvr32 rundll32 sc wscript	LOLBAS Outliner
		csc cscript dfsvc findstr gpscript mavinject msiexec netsh reg regsvr32 rundll32 sc wscript	LOLBAS Outliner
		csc cscript findstr gpscript msiexec netsh reg regsvr32 rundll32 sc wscript	LOLBAS Outliner
		rundll32 sc	LOLBAS Outliner

Putting the models together

Dashboard

hostname	anomaly_score	reason
	2	LOLBAS Outliner Obfuscated Command Line
	1	Obfuscated Command Line
	2	LOLBAS Outliner Obfuscated Command Line
	2	LOLBAS Outliner Obfuscated Command Line
	3	LOLBAS Outliner Obfuscated Command Line

Correlation to other threat events collected in SIEM

_time	search_name
	Permission Groups Discovery - Rule
	Permission Groups Discovery - Rule
	Detection of Command Line obfuscation - Rule

Event Drill Down

Detailed Process Events

_time	user	hostname	Obfuscated_Command	ParentProcessName	NewProcessName	CommandLine
			No	regedit	regedit	
			No	regedit	regedit	
			No	regedit	regedit	
			No	regedit	regedit	
			No	regedit	regedit	
			No	regedit	regedit	
			No	regedit	regedit	
			No	regedit	regedit	

Conclusion

Key Benefits

- Enhancing the alerting capabilities
- Remove alerts fatigues and enhance the overall detection rates as we use our env. Data as training dataset
- Ease the operation by shortening 20% of analysis time

Another Red Team Scenarios…..

Insider Threat – An malicious employee with privileged access

Malicious Commands used

System compromise – login to application to retrieve information

- **Irregular login proportion on the specific host**

hostnamesrc	orig_host	user	reason
			ı host - Supervisied

- **Alert Triggered by same user**

date	count	values(CommandLine)	values(SubjectUserName)	values(mitre_technique_category)	values(mitre_technique_name)
				Defense_Evasion	Deobfuscate/Decode Files or Information
				Defense_Evasion	Deobfuscate/Decode Files or Information

- **New Source IP address – Not coming from typical operation on a core application**

user		anomaly_score	reason
		4	unusal browser IP address
		4	unusal browser IP address

- **Malicious commands**

	user	ParentProcessName	NewProcessName	CommandLine

Further Work

- Develop deep learning model on malware classification
- Our own machine learning toolkit

Thank you !