

Why security data science matters and how it's different: pitfalls and promises of data science based breach detection and threat intelligence

Joshua Saxe, Invincea Labs

Work presented in this talk contains significant contributions from Alex Long, David Slater, Giacomo Bergamo, Konstantin Berlin, and Robert Gove

Background on Invincea

- ⦿ Invincea
 - ⦿ Security startup with ~80 staff focused on creating endpoint security solutions
- ⦿ Invincea Labs
 - ⦿ ~40 researchers working on a dozen or so research programs, some of which generate new products for Invincea as a whole
- ⦿ Invincea Labs-Data Science team
 - ⦿ 8 researchers working on a handful machine learning, visualization, and data mining projects mostly applied to security



Presentation Overview

- Session 1:
 - What is data science and why does it matter for defending our networks?
 - Data science at a glance – machine learning and visualization
 - Unique problems in security data science
- Session 2:
 - Three case studies from Invincea Labs
 - Machine learning based malware detection – going beyond signatures
 - Machine learning based prediction of malware family “hockey stick” growth – predicting malware’s future
 - Visualization and machine learning for threat intelligence – malware clustering and automated profiling

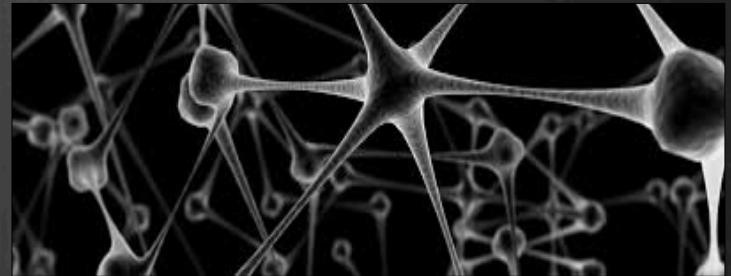
What is data science?

Data science consists of three technical areas ...

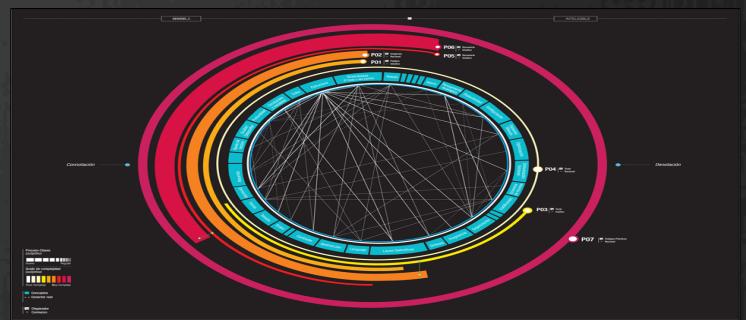
1. Scalable data storage technologies



2. Machine learning / scalable analytics

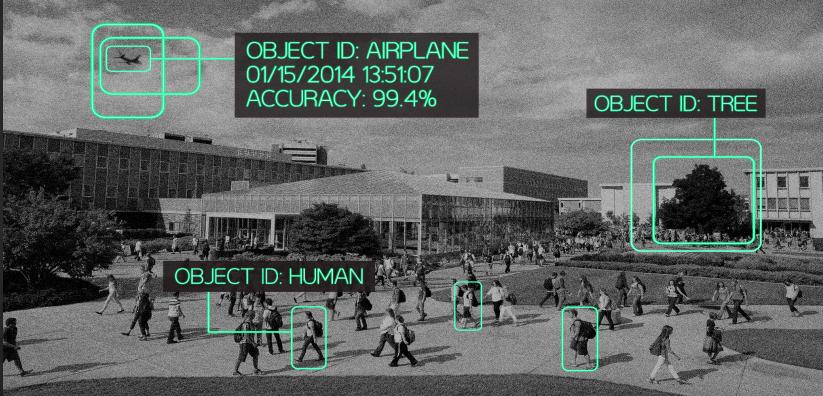


3. Data visualization / decision support



... and involves potentially limitless applications

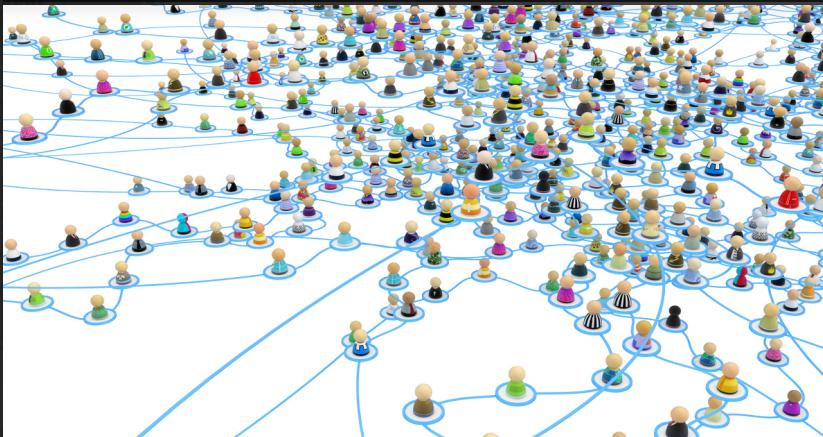
Computer vision



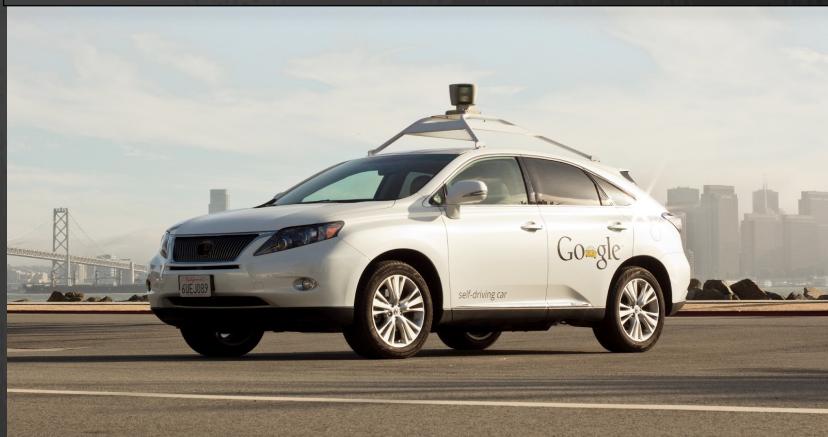
Voice recognition / natural language processing



Social network analysis

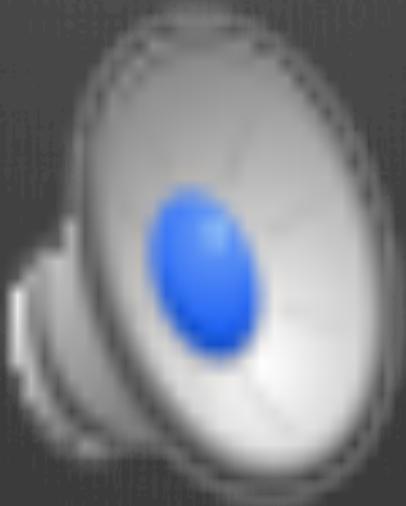


Robotics



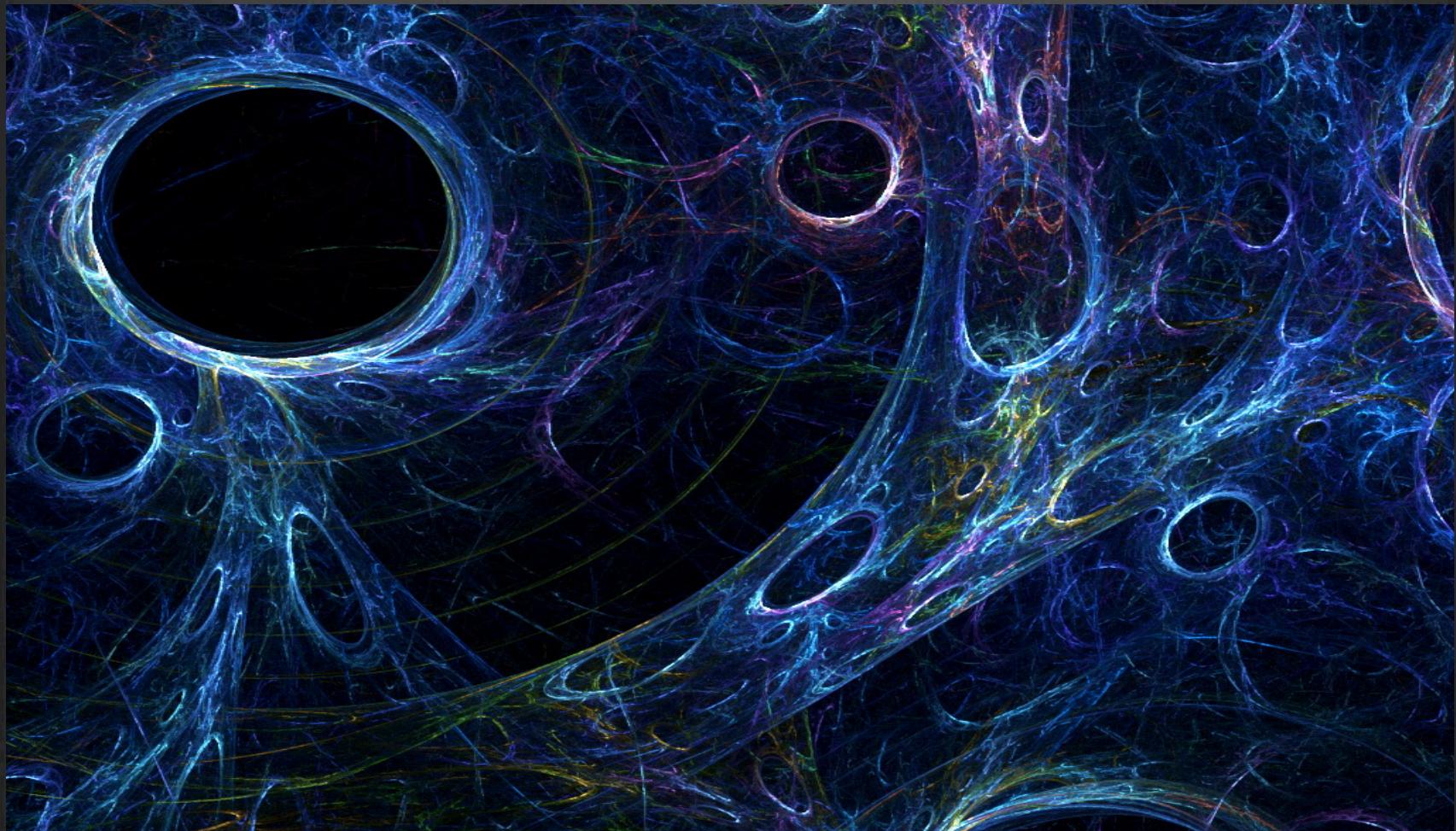
...

Data science projects often involve
all three data science areas



Why security needs data science

We have access to the signals needed to detect attackers,
but they are currently the “dark matter” of our field

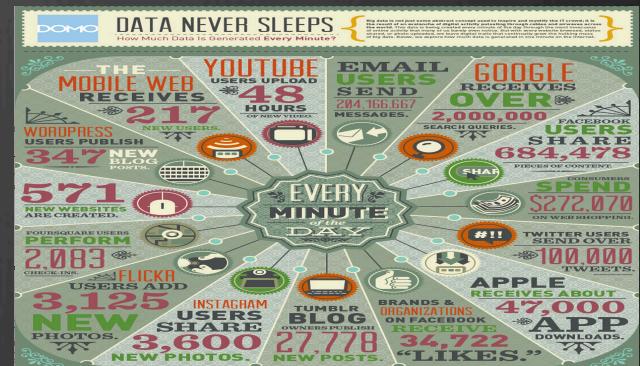


The “dark matter” of information security is where attackers hide

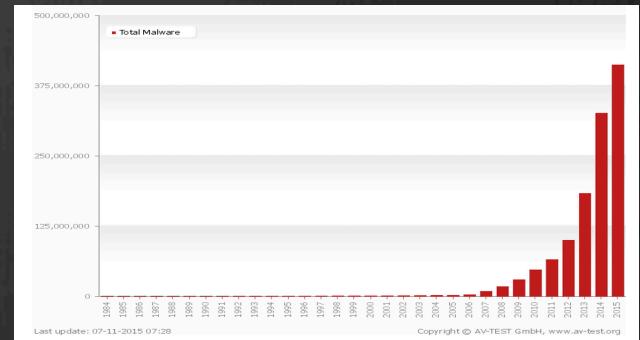
Terabytes of logs generated by enterprise networks that are never examined ...

```
[  9.039944] type=1400 audit(1436890728.649:9): apparmor="STATUS" operation="profile_replace" script/dhcclient-script" pid=1048 comm="apparmor_parser"
[  9.039954] type=1400 audit(1436890728.649:10): apparmor="STATUS" operation="profile_replace" Manager/mm-dhcp-client.action" pid=1047 comm="apparmor_parser"
[  9.415727] ISOFs: Unable to identify CD-ROM format.
[  9.665743] init: fail-safe main process (1273) killed by TERM signal
[  9.771719] Bluetooth: Core ver 2.17
[  9.771755] NET: Registered protocol family 31
[  9.771757] Bluetooth: HCI device and connection manager initialized
[  9.771767] Bluetooth: HCI socket layer initialized
[  9.771768] Bluetooth: L2CAP socket layer initialized
[  9.771773] Bluetooth: SCO socket layer initialized
[  9.780319] init: avahi-cups-reload main process (1404) terminated with status 1
[  9.793605] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[  9.793608] Bluetooth: BNEP filters: protocol multicast
[  9.793619] Bluetooth: BNEP socket layer initialized
[  9.793709] Bluetooth: RFCOMM TTY layer initialized
[  9.793749] Bluetooth: RFCOMM socket layer initialized
[  9.793754] Bluetooth: RFCOMM ver 1.11
[ 10.271274] init: plymouth-upstart-bridge main process ended, respawning
[ 10.339397] init: pollinate main process (1510) terminated with status 1
[ 13.128016] init: lightdm main process (1555) terminated with status 1
[ 13.131751] init: plymouth-ready (startup) main process (680) terminated with status 1
[ 61.949085] nvidia_uvm: Loaded the UVM driver, major device number 251
```

Reams of user behavioral data that could hold information about threats ...

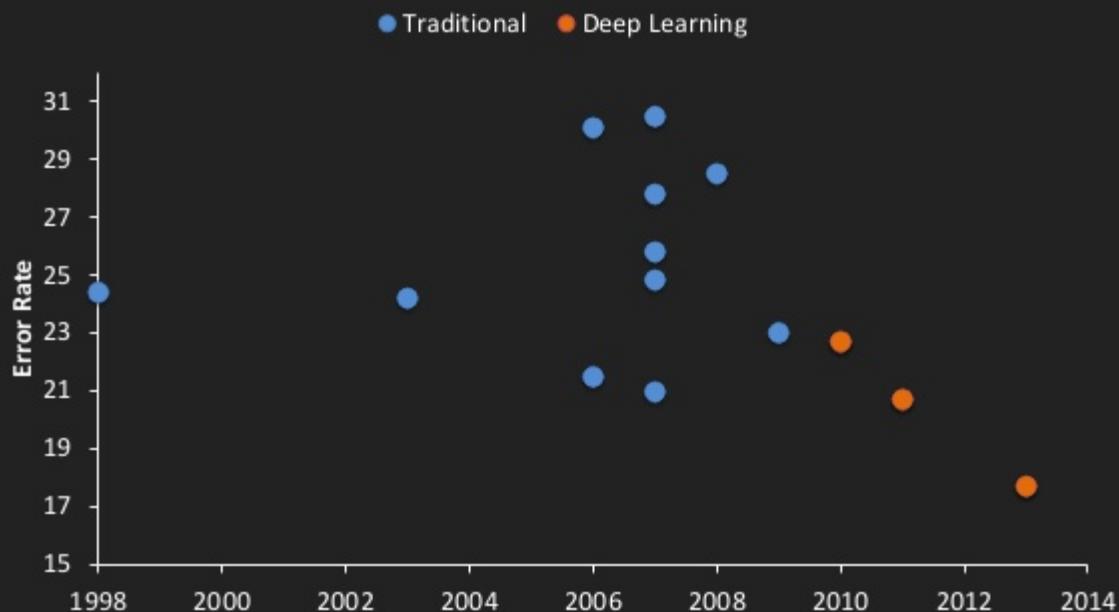


... the 400 million+ malware samples seen in the Internet, most of which have never been analyzed



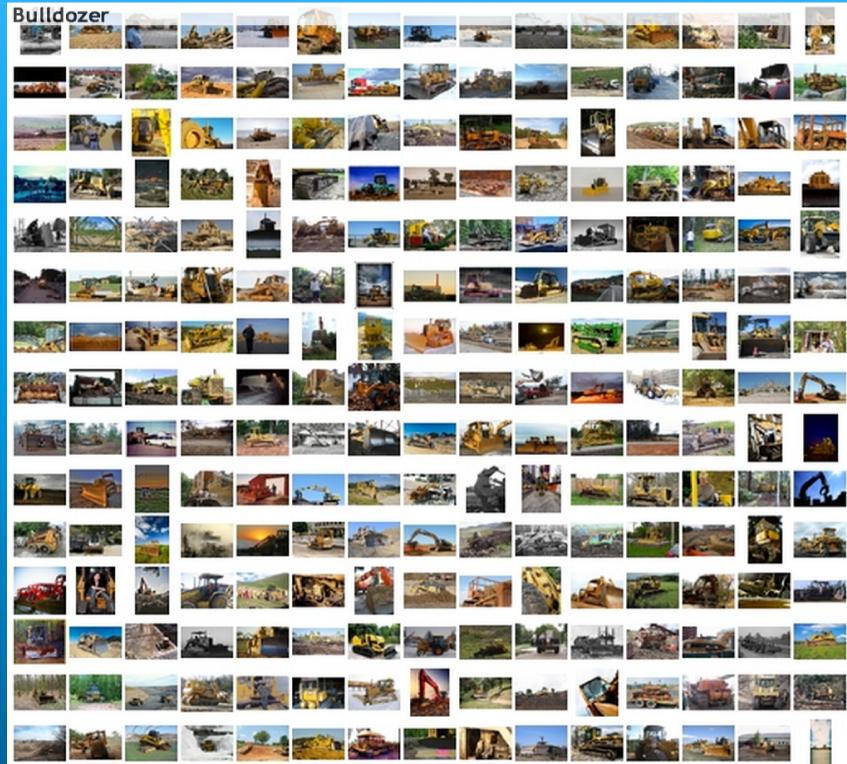
Evidence we can be successful: data science breakthroughs in adjacent domains

TIMIT Speech Recognition

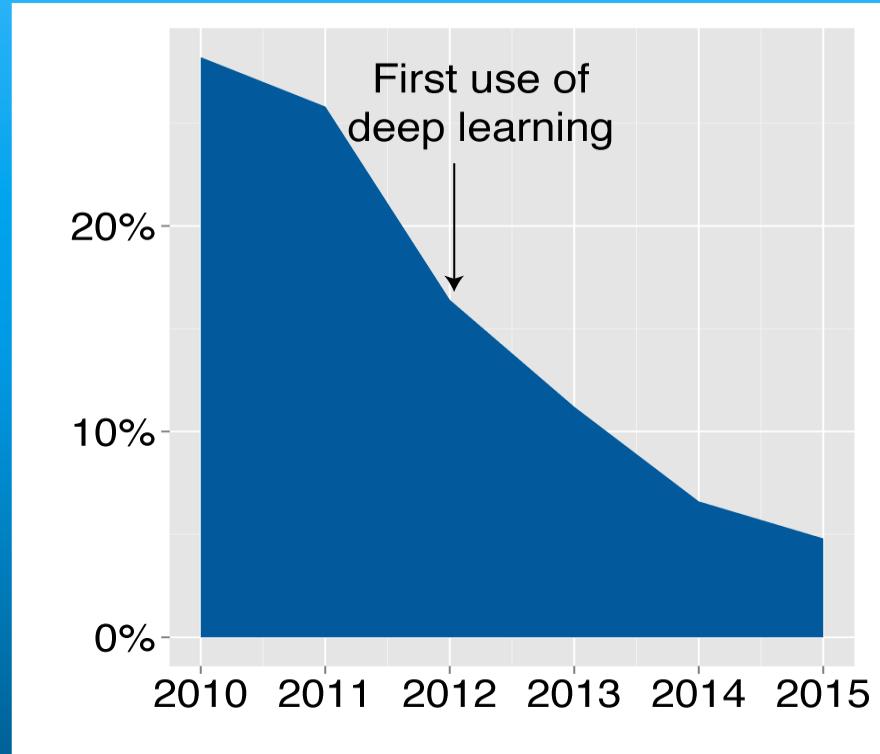


Evidence we can be successful: data science breakthroughs in adjacent domains

ImageNet examples



Objection classification error rate



The big question:
Can data science produce
analogous breakthroughs to
problems that seem intractable
in the security space?

A lightning overview of machine learning

Some definitions: training data, test data, prediction, generalization

- ➊ Training data: data that we show to machine learning algorithms to “teach them”
- ➋ Labeled training data: data that we show to machine learning algorithms that alongside *answers to questions*
- ➌ Unlabeled training data: data that we show to machine learning algorithms without answers to questions
- ➍ Test data: data that we show to machine learning algorithms to evaluate their accuracy

Some definitions: types of machine learning

- Supervised learning: algorithms that require *labeled training data*
- Unsupervised learning: algorithms that do not require *labeled training data*

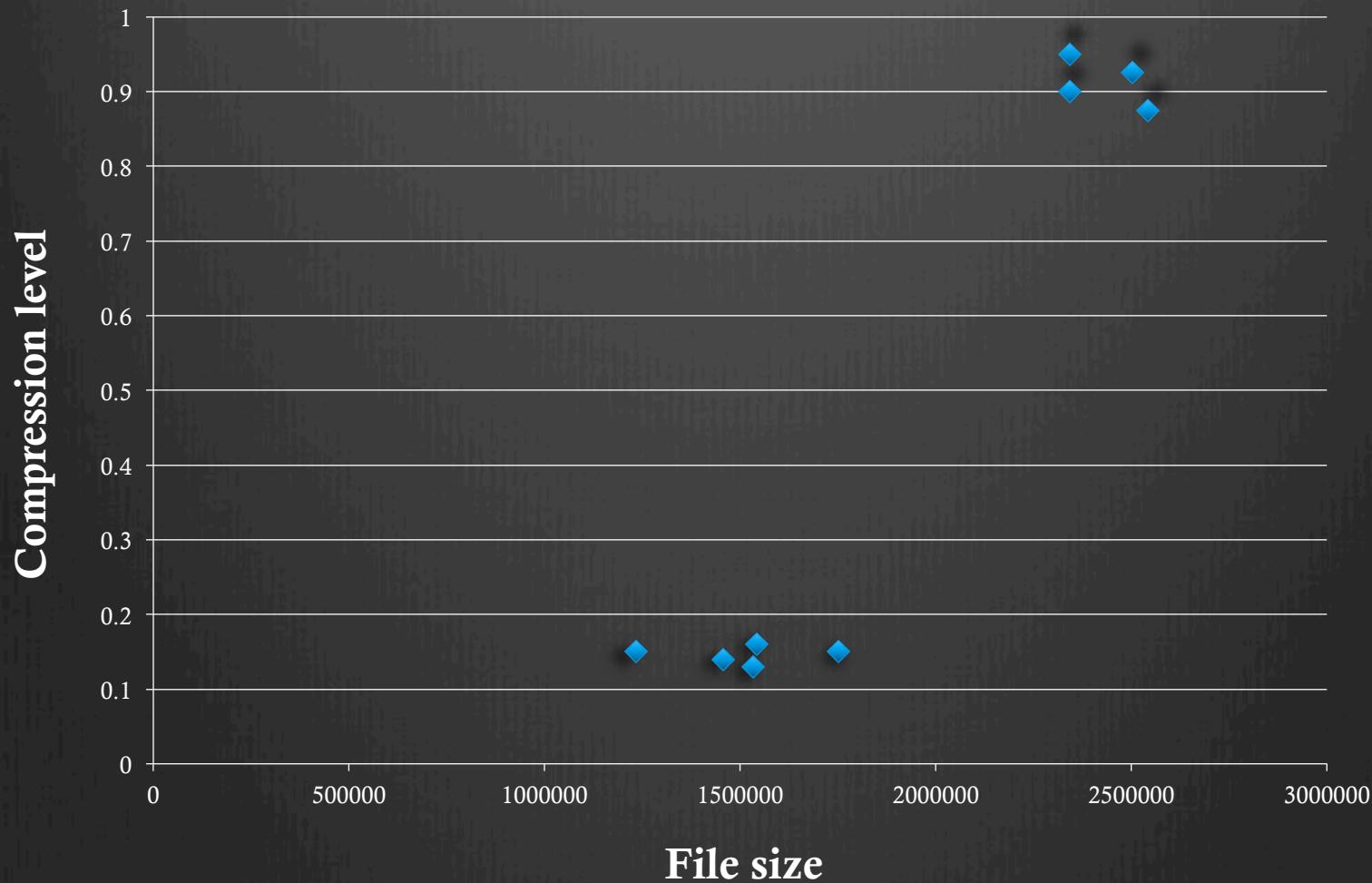
Definition of a “feature *space*”

Raw files ...
/malware/file1
/malware/file2
/malware/file3
...

Feature extraction code

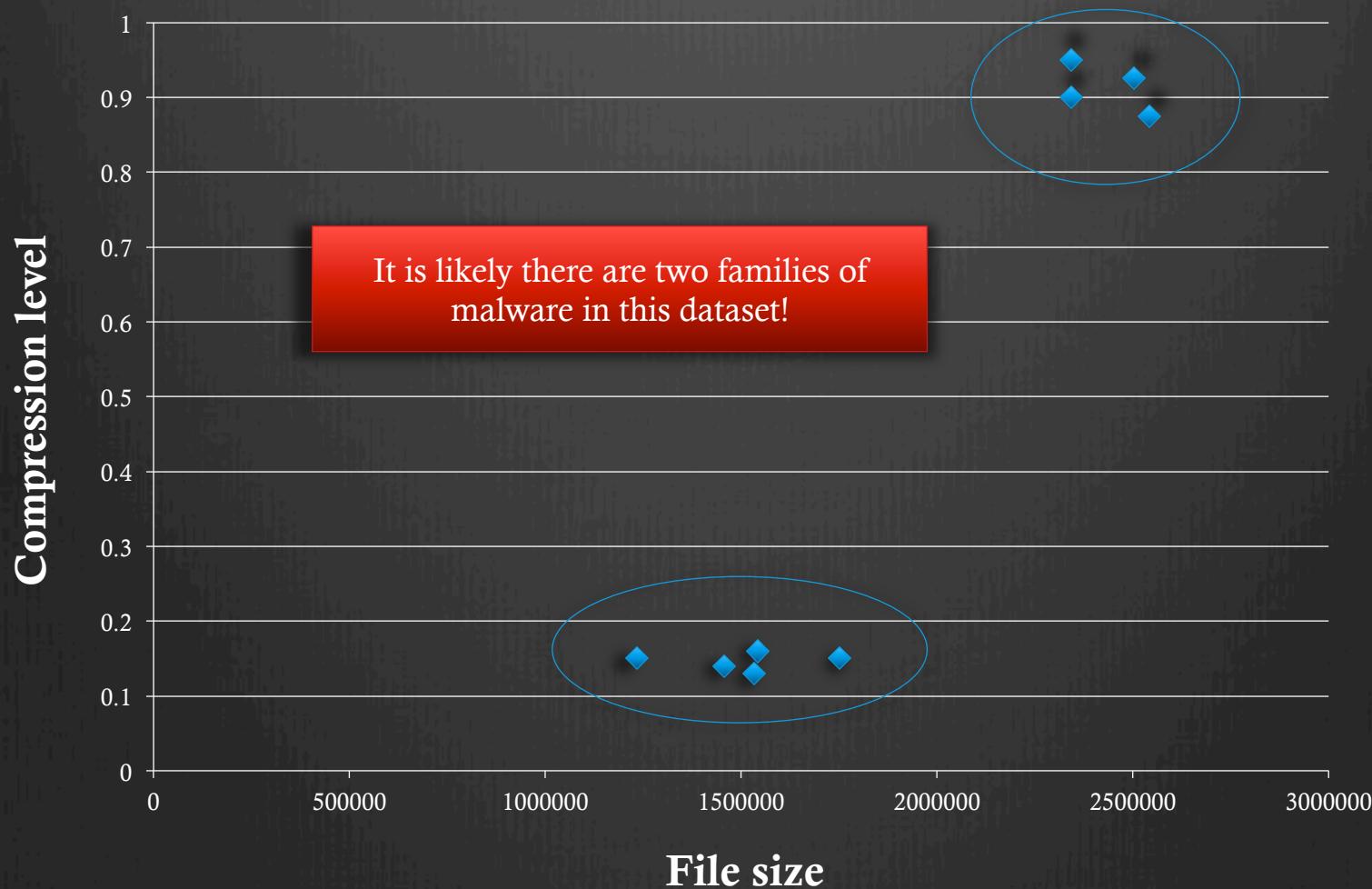
Feature space		
	File size	File compression level
Adversary group 1	2342315	0.95
	2343909	0.9
	2504321	0.925
	2541234	0.875
	1234145	0.15
	1534145	0.13
	1542145	0.16
	1751145	0.15
	1456145	0.14

Visualization of the two-dimensional “file” feature space



Clustering

Clustering algorithms focus on identifying like-groups without any prior knowledge about the data

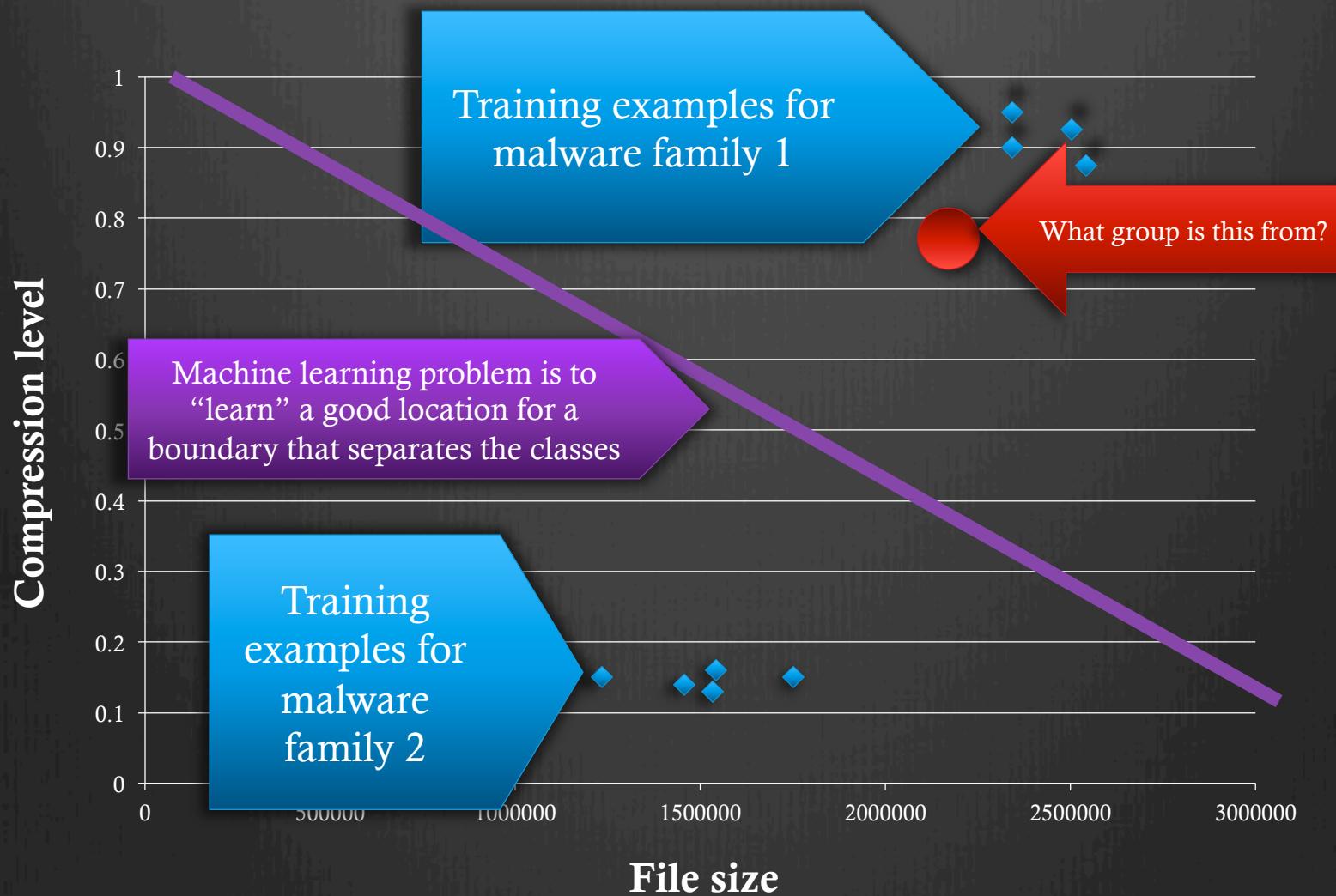


Why clustering matters

- ➊ Group together similar security events
- ➋ Group together similar malware
- ➌ Group together similar network streams
- ➍ Group together similar (malicious) domain names



Classification



A real-world classification algorithm in action (credit: Andrej Karpathy)

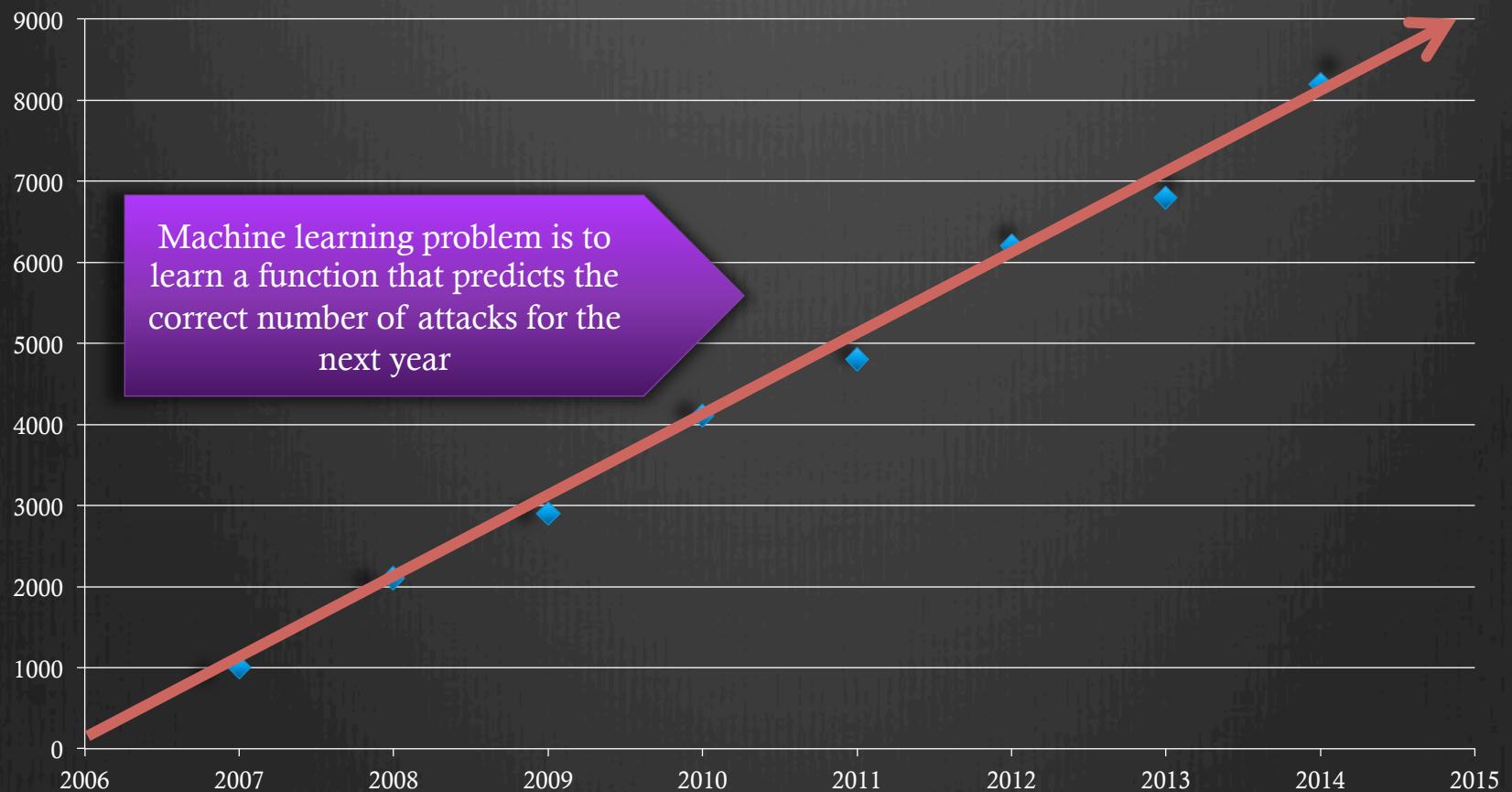


Why classification matters

- Detect malware and malicious behavior by classifying good vs. bad
- Detect suspicious network streams by classifying good vs. bad
- Detect *types* of network streams based on their contents
- ...

Regression

Number of attacks over time



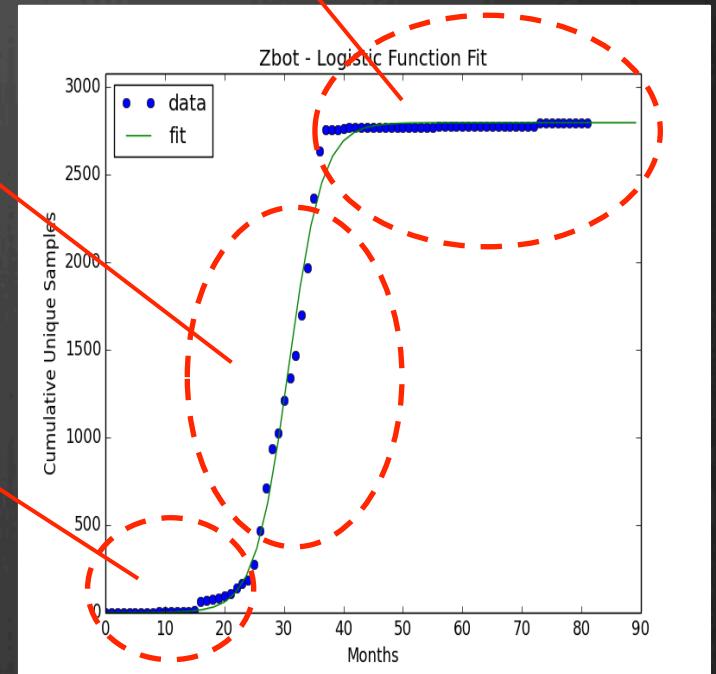
Why regression matters

- Predict future growth of malware families
- Identify security incident trends
- Identify growth of botnets
- ...

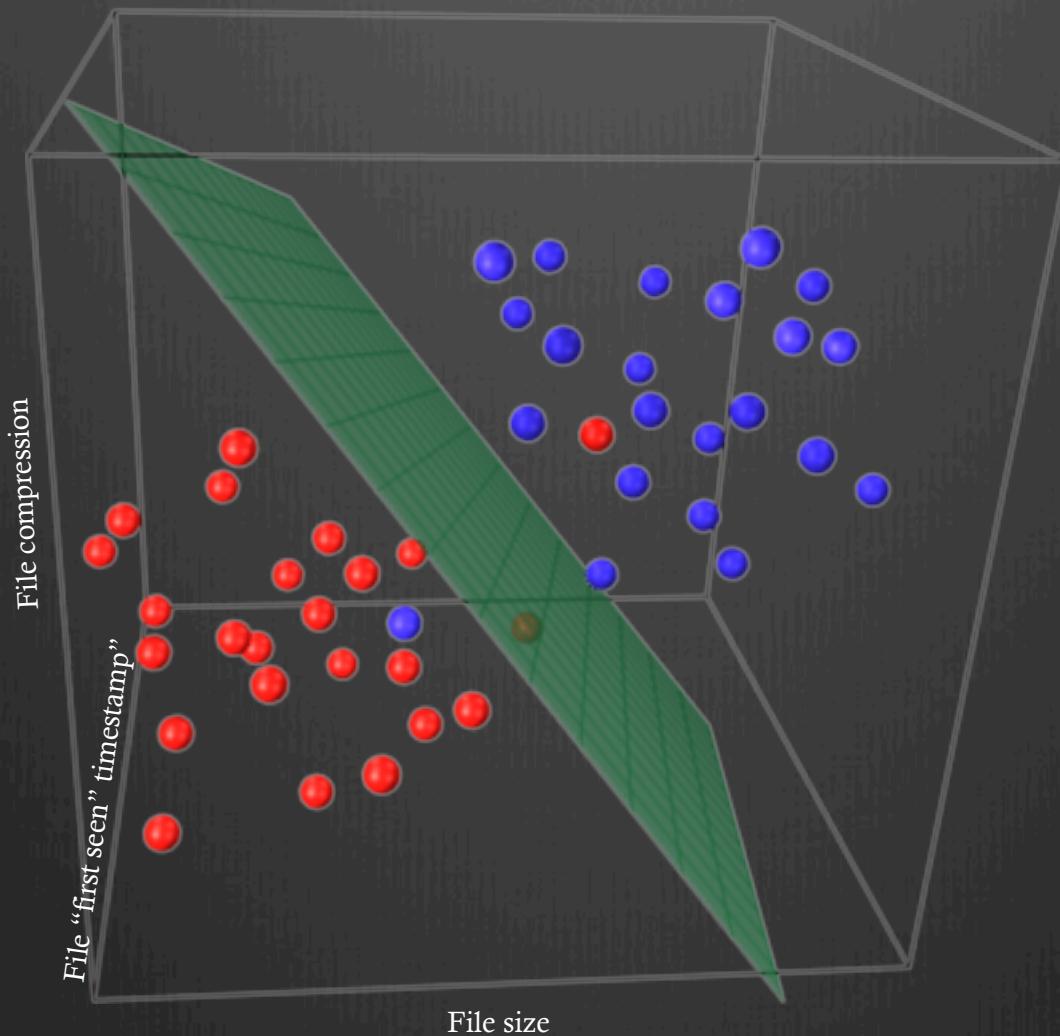
Rapid growth

R&D

Late adopters

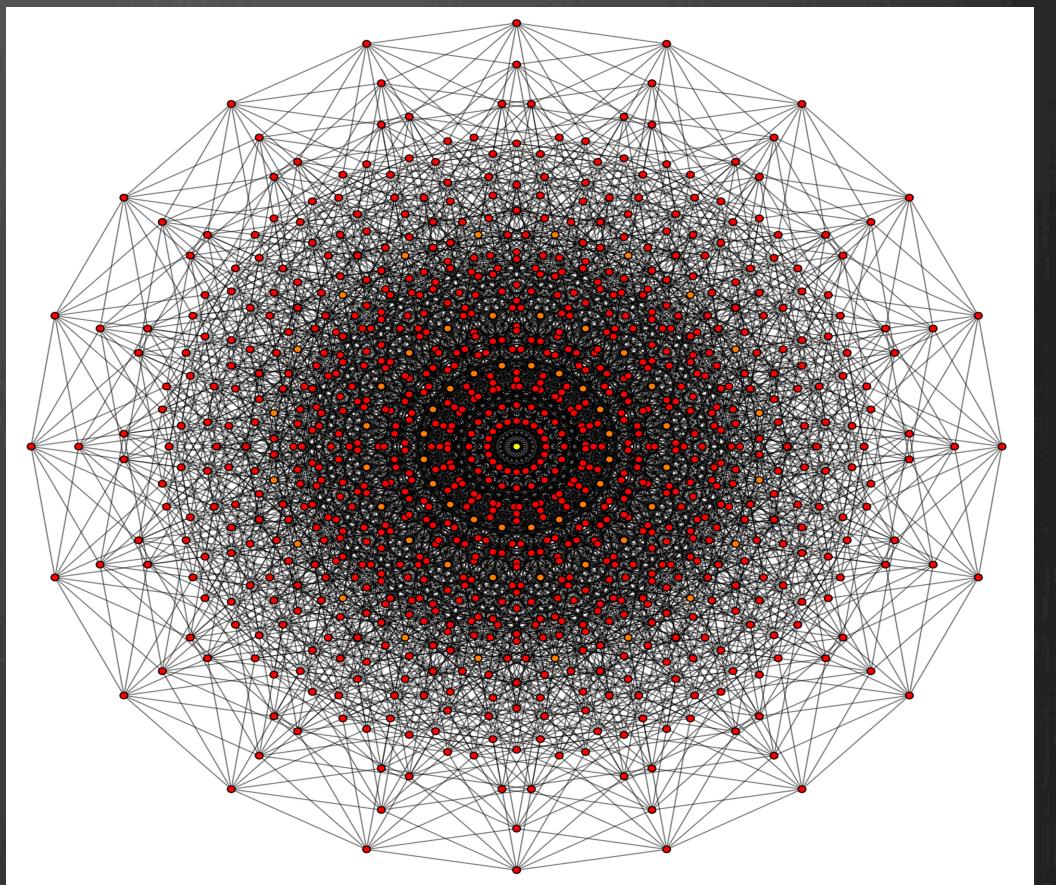


Adding dimensions: A classifier in three dimensions



What would ten dimensions look like?

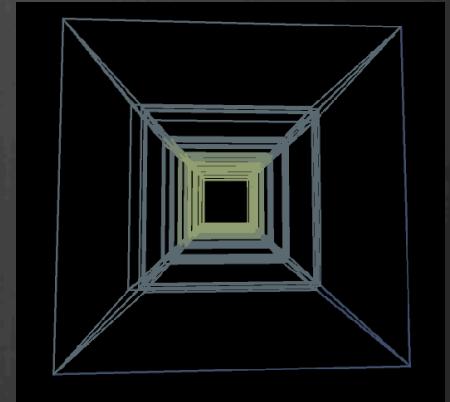
- Impossible to truly visualize
- If we could actually see this ten dimensional cube, all angles would be 90 degrees
- And, all lines would be equal length
- *And yet, machine learning algorithms operate in such high-dimensional spaces*



A ten dimensional cube!

Thinking about dimensionality

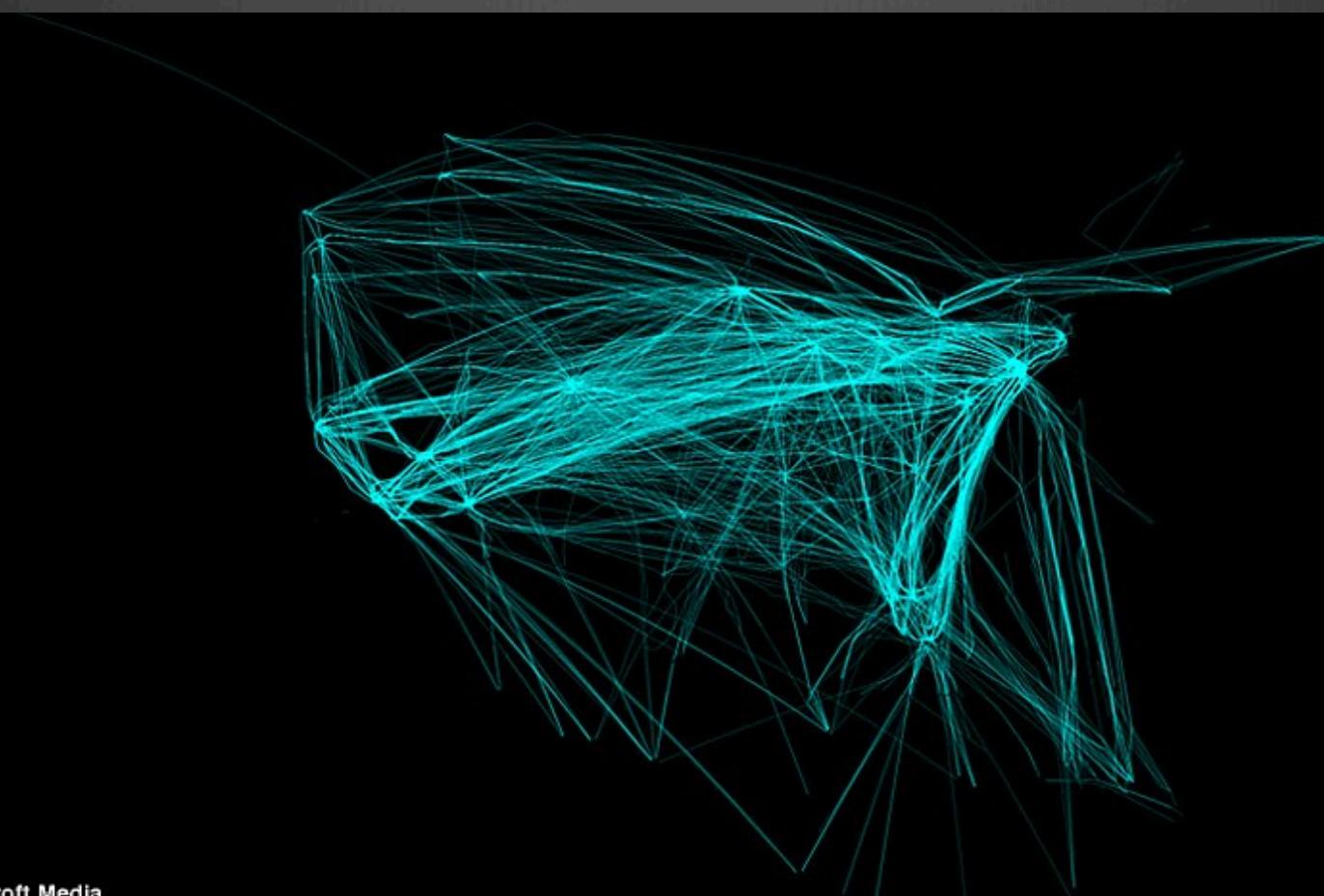
- ➊ In security data science, for most problems we care about, data have not 2-dimensions, but *thousands* or *millions* of dimensions
- ➋ High dimensional data make physical intuition more difficult, but it nonetheless is how most data scientists reason about their models



Questions?

Data visualization

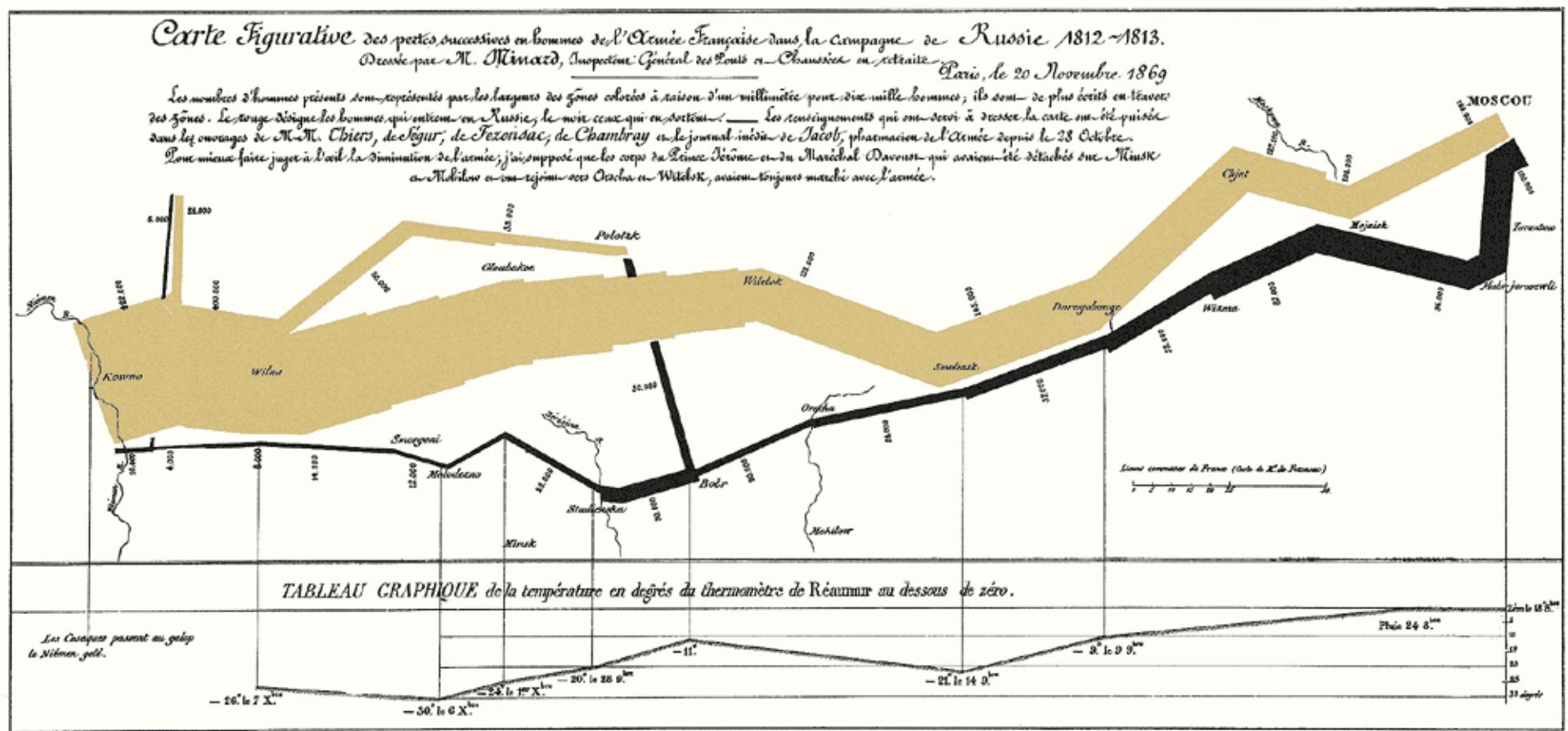
Real world data visualization: Some visualizations reveal
“unknown unknowns” about data



© Aaron Koblin/Barcroft Media

Credit: Aaron Koblin

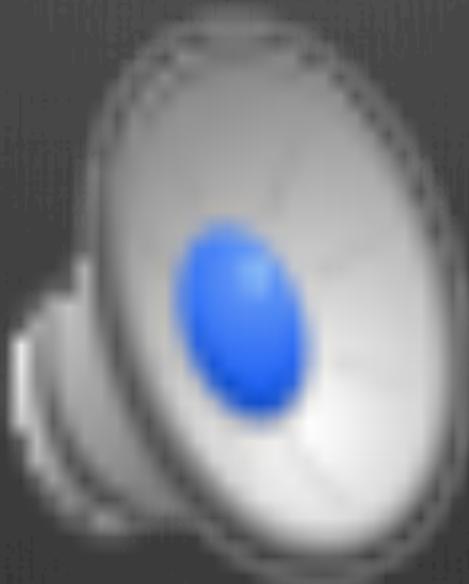
Others answer hard questions in a visually intuitive way



Credit: Charles Joseph Minard

Credit: Andrew H. Caudwell / Gource

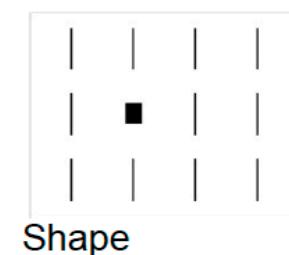
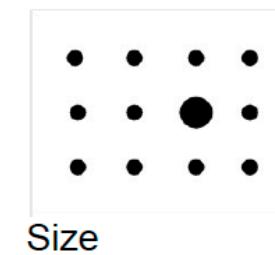
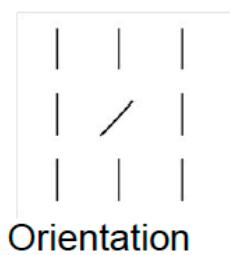
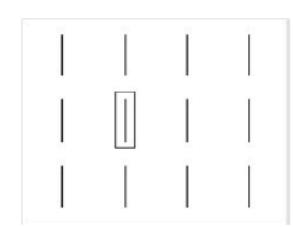
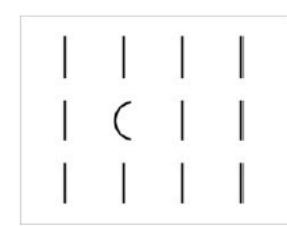
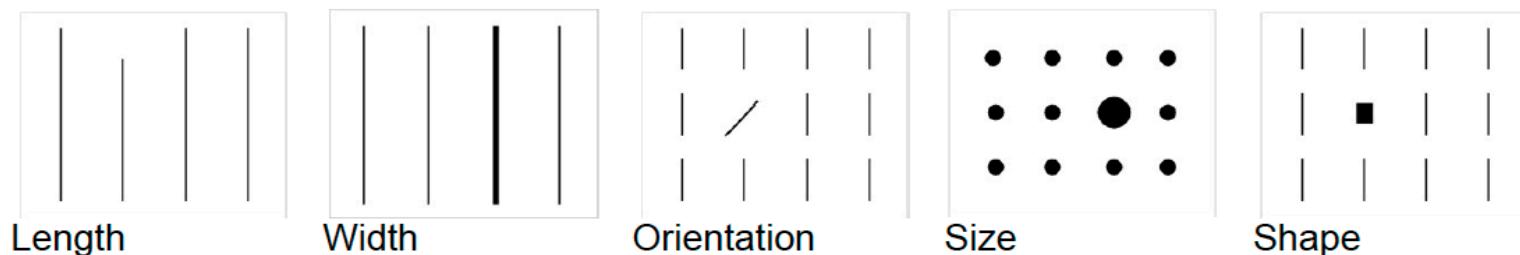
Visualizations are not necessarily static



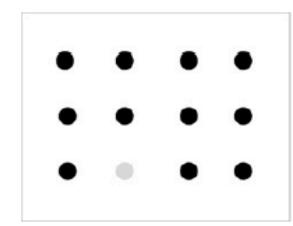
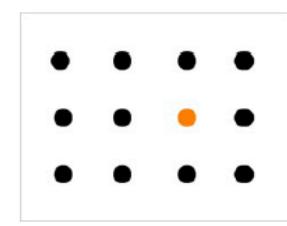
Data processing capabilities of the human visual system

Preattentive attributes of visual perception

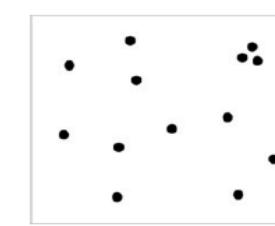
Form



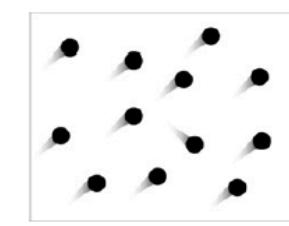
Color



Position

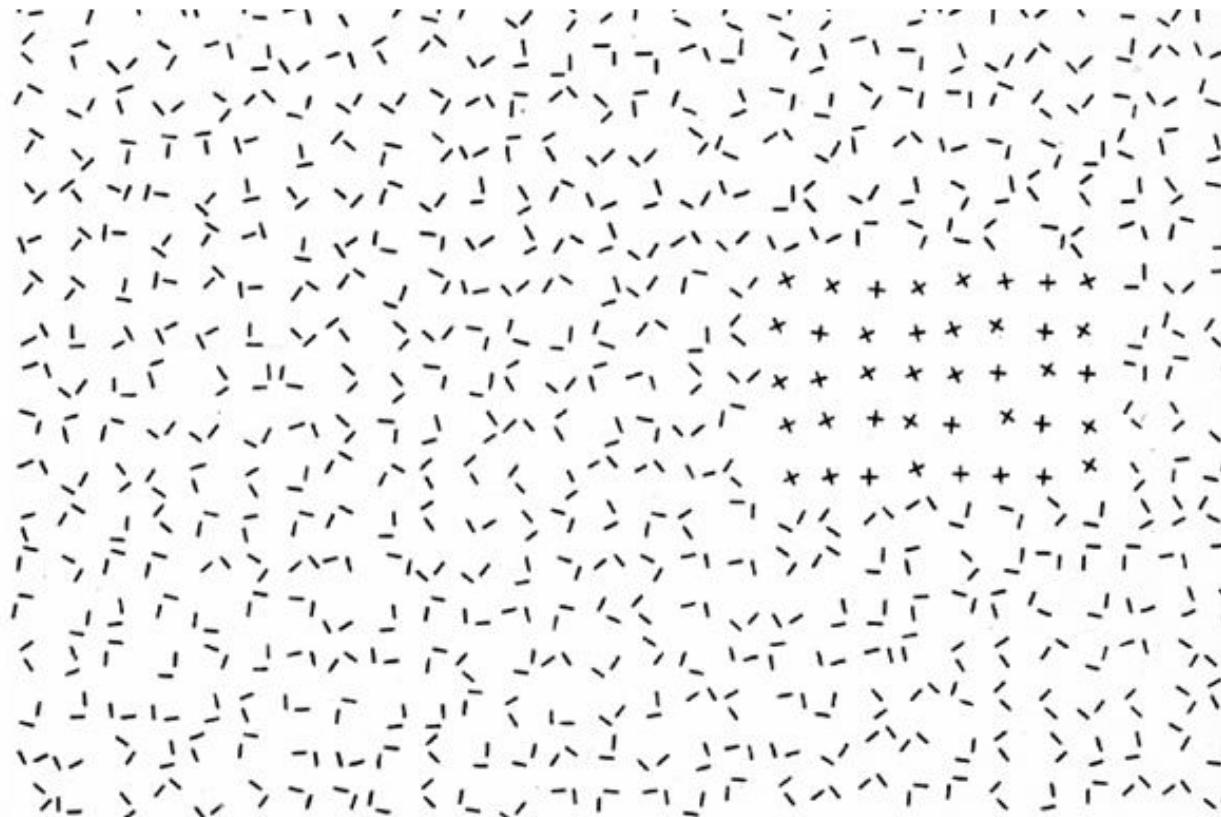


Motion



More preattentive reasoning

Visual search

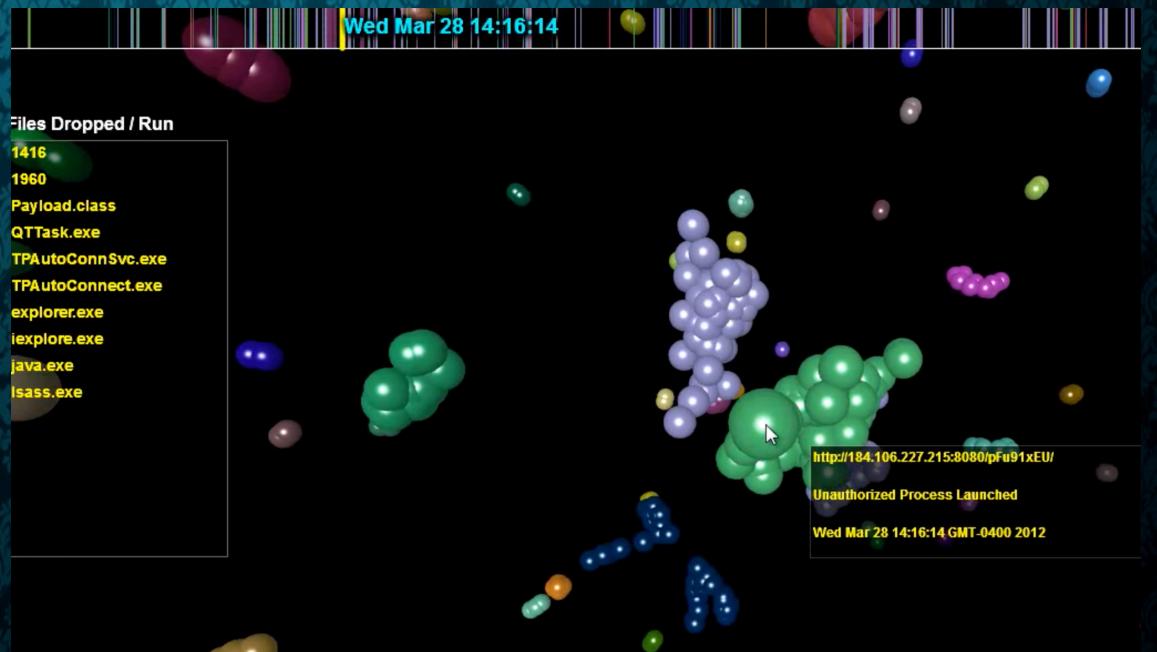


A human visual system “fail”



Visualization design principles

- Overview first, details on demand
- Visualizations should answer questions, or should be explicitly exploratory
- Design with a user in mind, and a user workflow in mind
- Don't overload the user with too many visual channels

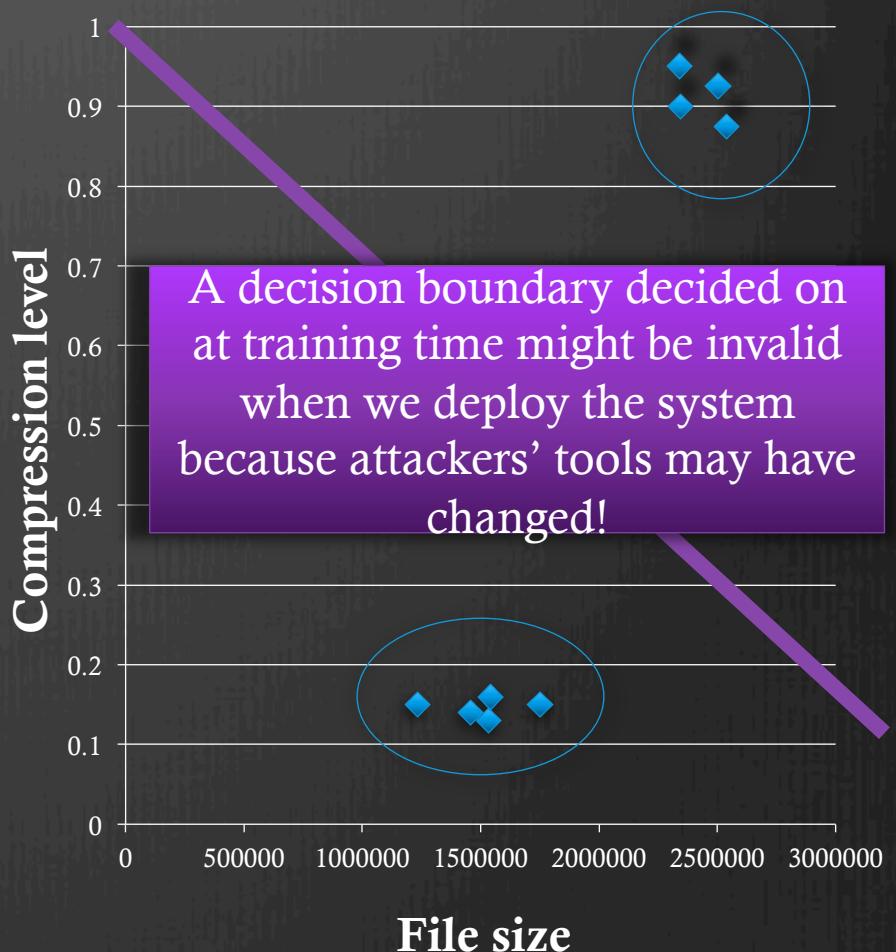


Questions?

What makes security data
science different?

The presence of an adversary

- ⦿ Attackers are deliberately trying to evade detection systems
 - ⦿ This makes applying classification, regression, and clustering a far different problem than, for example, classifying news articles into categories
- ⦿ Attackers techniques are constantly changing
 - ⦿ This makes applying data science methods to security different from, say, voice recognition



Degradation of machine learning based malicious behavior detection due to adversary evolution

Addressing the “evolving adversary” problem:
Adversarial evolution should be addressed by modeling attacker behavior in the way we evaluate our data science tools

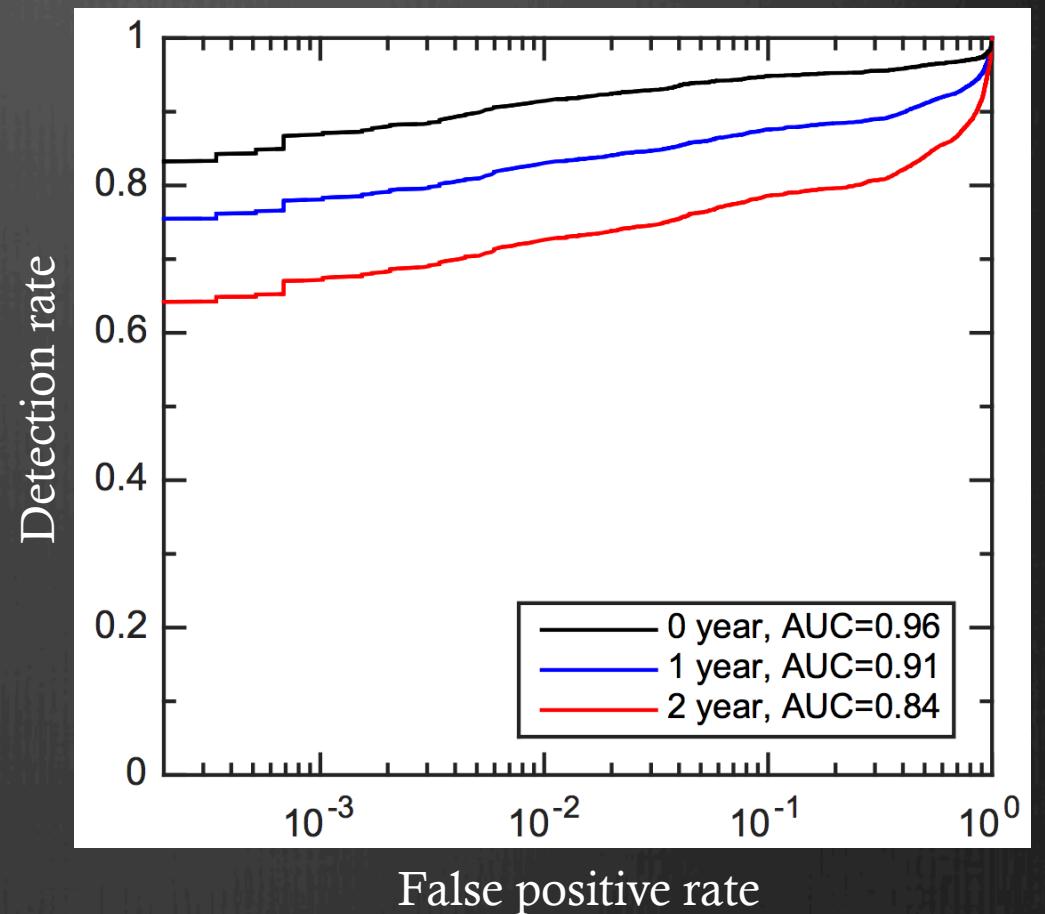


Figure from work by Konstantin Berlin, David Slater, Joshua Saxe

The false positive problem

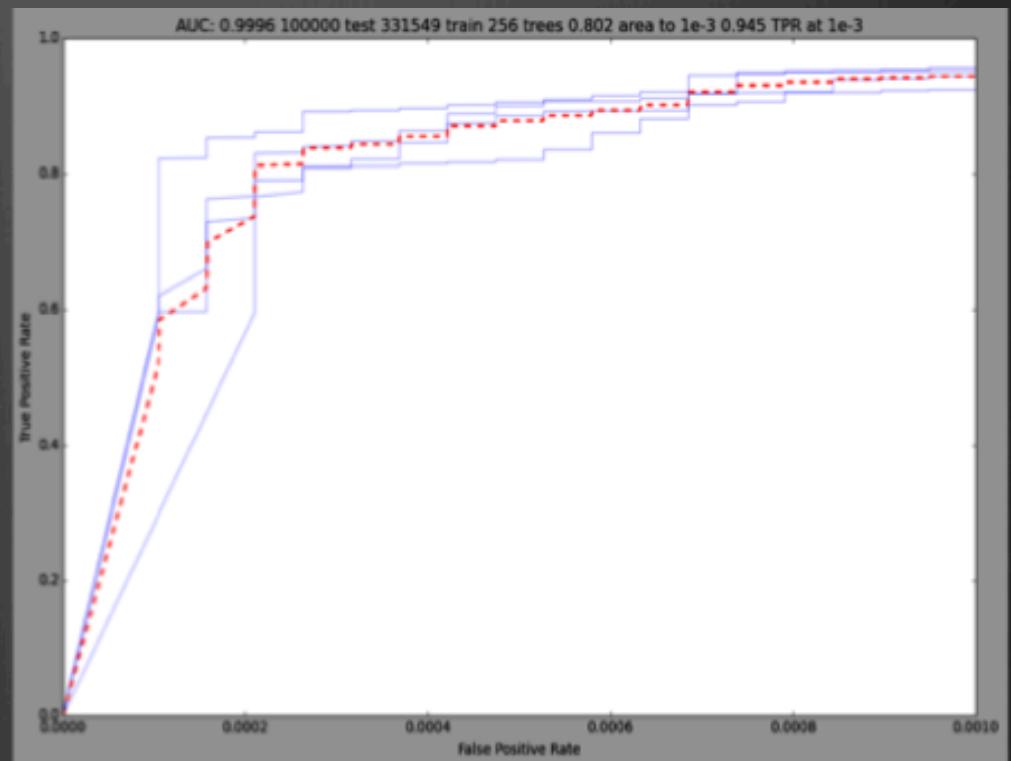
- Many of the things we look for in security are extremely *rare*
 - Exploitable vulnerabilities are rare
 - On an enterprise network, malware is rare relative to benignware
 - Insider threat behavior is rare
 - False positives *rates* (the percentage of false cases mislabeled as true) have to be *extremely* low in security for approaches to be useful

A machine learning based detection “fail” due to false positives

- Our detection system achieved a 75% detection rate at a 0.4% false positive rate
- BUT, when we do the math, we see that:
 - **34% of our alarms will be false positives** if we assume 1/100 malware to benign ratio
 - **84% of our alarms will be false positives** if we assume a 1/1,000 malware to benign ratio
 - **98% of our alarms will be false positives** if we assume a 1/10,000 malware to benign ratio

Addressing the false positive problem

- Focus on measurement of system detection rate in the ultra-low false positive region
- Emphasize collection of huge volumes of benign examples so the false positive rate can be accurately measured in low-FPR region
- Pick models that predict accurate probabilities – calibrate these models
- Learn the dark art of designing systems that operate well at low false positive rates!



The need for interpretability

- In other areas of data science, all we care about are correct detections
- In security correct detections are only part of the story, we also need explanations
(if a data science tool tells me I've been compromised, I want to see all of the evidence it is using to make that claim so I can follow up)

- gui captures mouse movement 2

TOKEN	PROBABILITY OF CAPABILITY GIVEN TOKEN	SOURCE POST TITLE
SetCursorPos	0.69	Move mouse with c#
GetMessagePos	0.65	Mouse state winapi
DrawIcon	0.60	Draw mouse pointer icon?
TrackMouseEvent	0.53	Mouse Move Capture (Mouse leave & Mouse Enter)
MousePos	0.47	TListView and mouse wheel scrolling
SetCapture	0.46	Activate windows under mouse through mouse hook
WindowFromPoint	0.43	Activate windows under mouse through mouse hook
WheelDelta	0.40	Trigger 'dummy' mouse wheel event
OnMouseMove	0.33	mouse movements in wpf
OnMouseEnter	0.32	C# mouse picking XNA
SetCursor	0.32	draw mouse cursor in win32
GetCursorPos	0.31	Mouse wheel event
ReleaseCapture	0.29	Activate windows under mouse through mouse hook
GetIconInfo	0.26	Mouse cursor bitmap

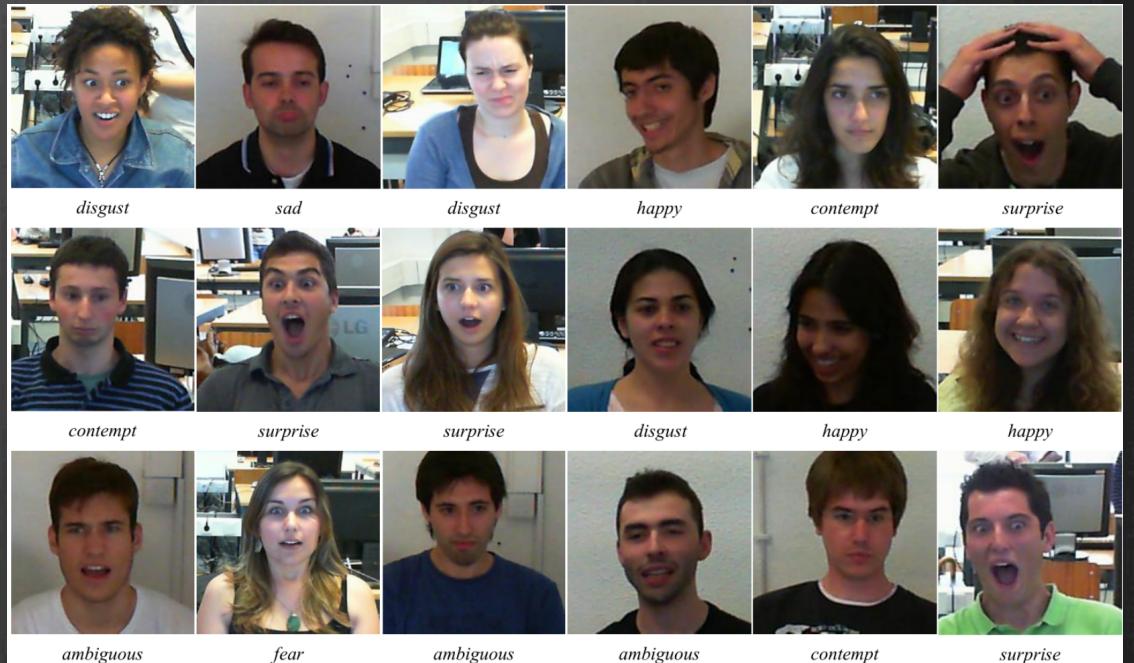
Building interpretability into the data science workflow

- ⦿ Throughout the data science workflow, think about how you're building interpretability into your data science product
- ⦿ Presentation is often just as important as model accuracy, because it means people can actually *use* your results

The lack of labeled data

In other data science areas, the research community has millions of labeled examples to train their models on

Not so when it comes to zero-day attack data!



One potential mitigation

- Use crowdsourced data in auxiliary domains to build data science approaches

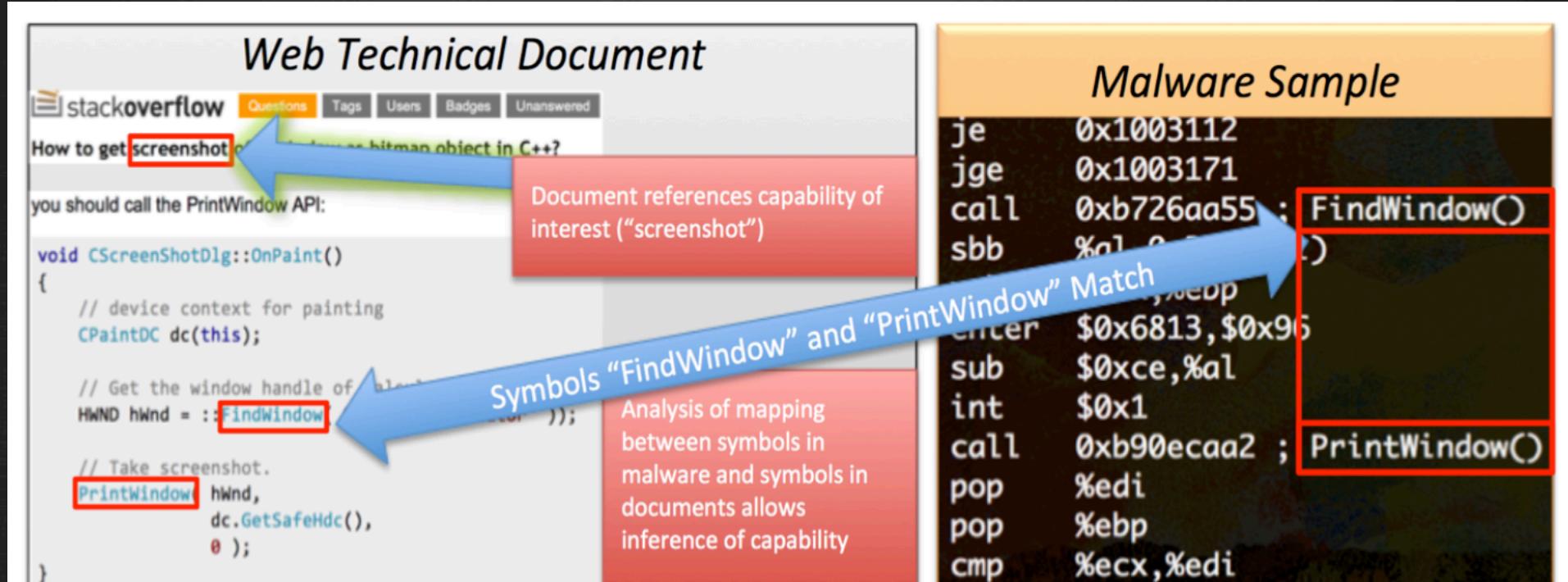
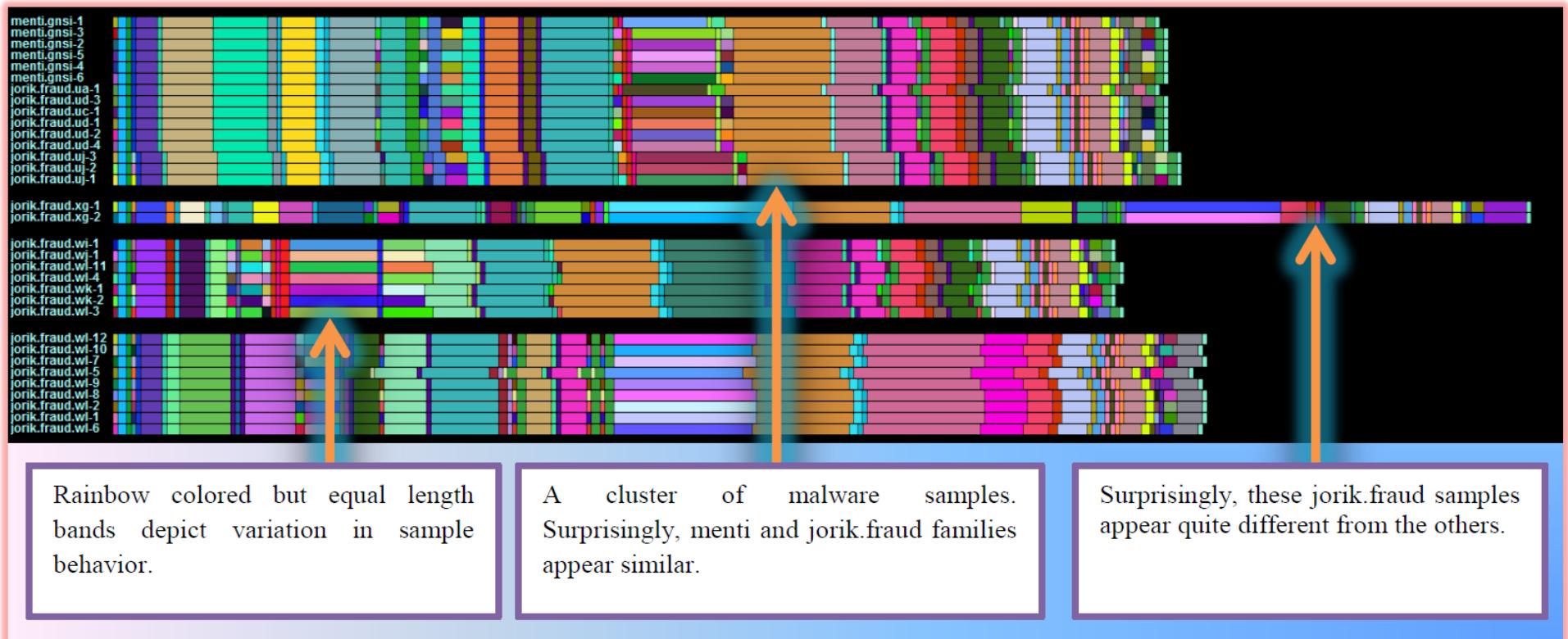


Figure 1. An illustration of the intuition that we can correlate terms found in malware binaries and terms found in StackExchange documents to identify high-level software capabilities within malware

Another potential mitigation

- Build approaches that don't require any ground truth



Questions, discussion?

- Plug for next half of the talk:
- Deep learning based malware detection producing unprecedented accuracy!
- Predicting which malware families are going to go “viral” using autoregression!
- Automatically detecting malware capabilities and “social network” relationships using a suite of machine learning, data storage and visualization tools!

Three security machine learning cases studies

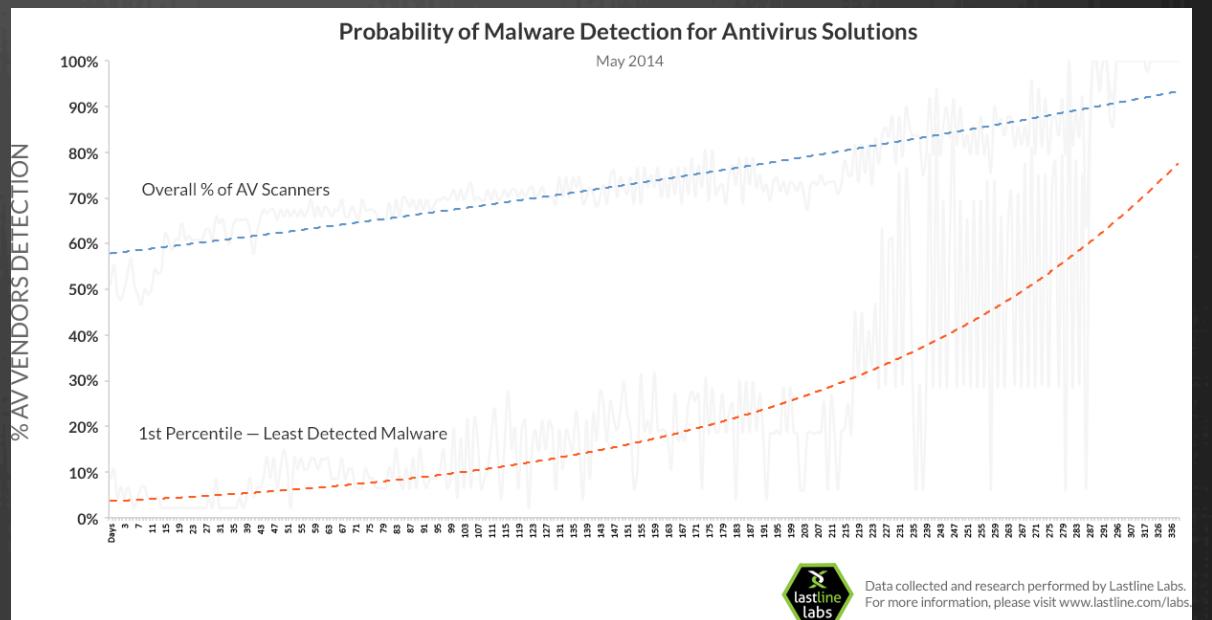
Agenda

- ⦿ Deep neural network based malware detection
(Joshua Saxe, Konstantin Berlin)
- ⦿ Predicting the explosive growth of malware families
(Giacomo Bergamo, Joshua Saxe)
- ⦿ Cynomix: A machine learning driven workbench for malware analysis and malware relationship analysis
(Alex Long, Giacomo Bergamo, Joshua Saxe, Robert Gove, Sigfried Gold)

Case study 1:
Deep neural network-based malware detection
Joint work by Joshua Saxe and Konstantin Berlin

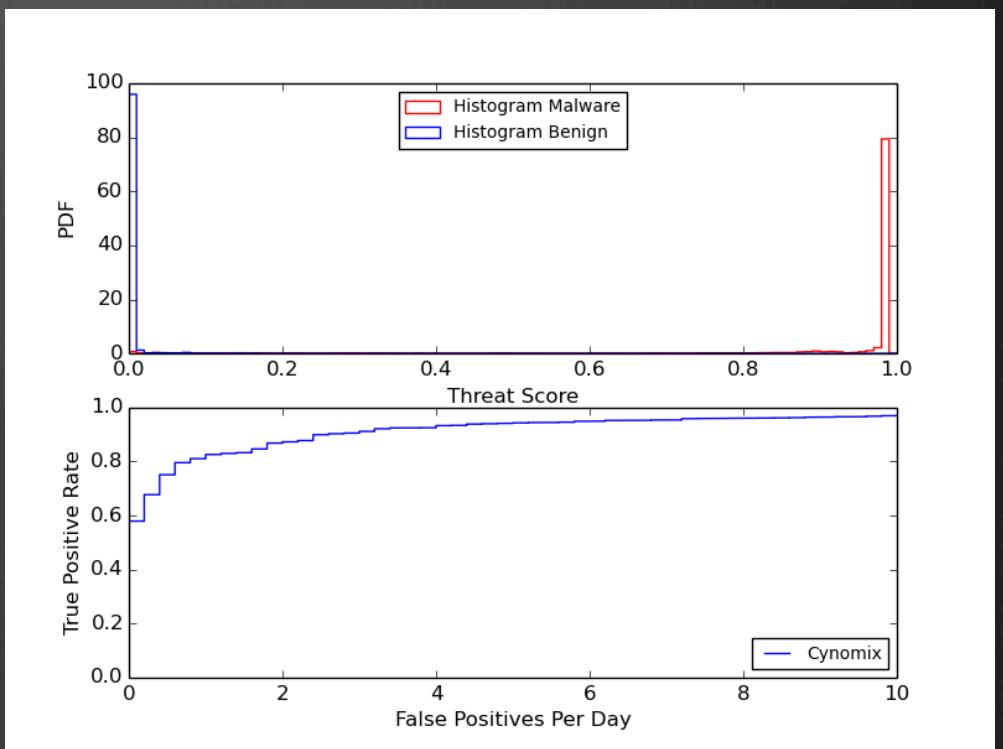
Motivation

- ➊ On average, anti-virus sensors have a 40% chance of missing zero-day malware
- ➋ The data seem to suggest that it takes almost a year before anti-virus sensors start detecting the *hardest-to-detect* zero-day malware
- ➌ Machine learning holds the promise of providing an orthogonal detection methodology that can significantly increase our overall detection rate



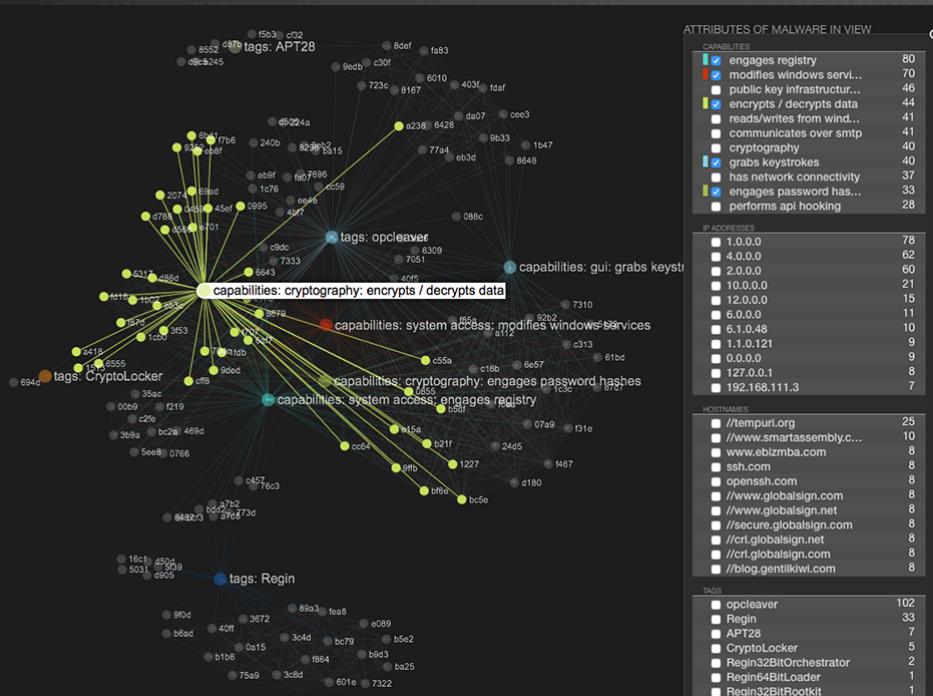
Results in an operational setting

- 60% detection rate on new malware as false positives converge on 0 (in contrast to anti-virus engines' 40%)
- 95% detection rate on new malware as false positives approach five per day
- *Test performed on feed of new binaries obtained from multiple customer networks compromising on the order of thousands of individual machines*



How we accomplished this,
starting with intuition behind
our system

Intuition about inputs: Exploit shared code relationships



If we train on a malware sample with code component A, and then test on a sample with component A and previously unseen component B, and component A occurs rarely or never in benignware, we should classify the sample as malware

Intuition about inputs

Exploit malware componentization

If we **train** on a malware sample with data component A, and then **test** on a sample with data component A and previously unseen data component B, and component A rarely or never occurs in benignware, we should classify the test sample as malware



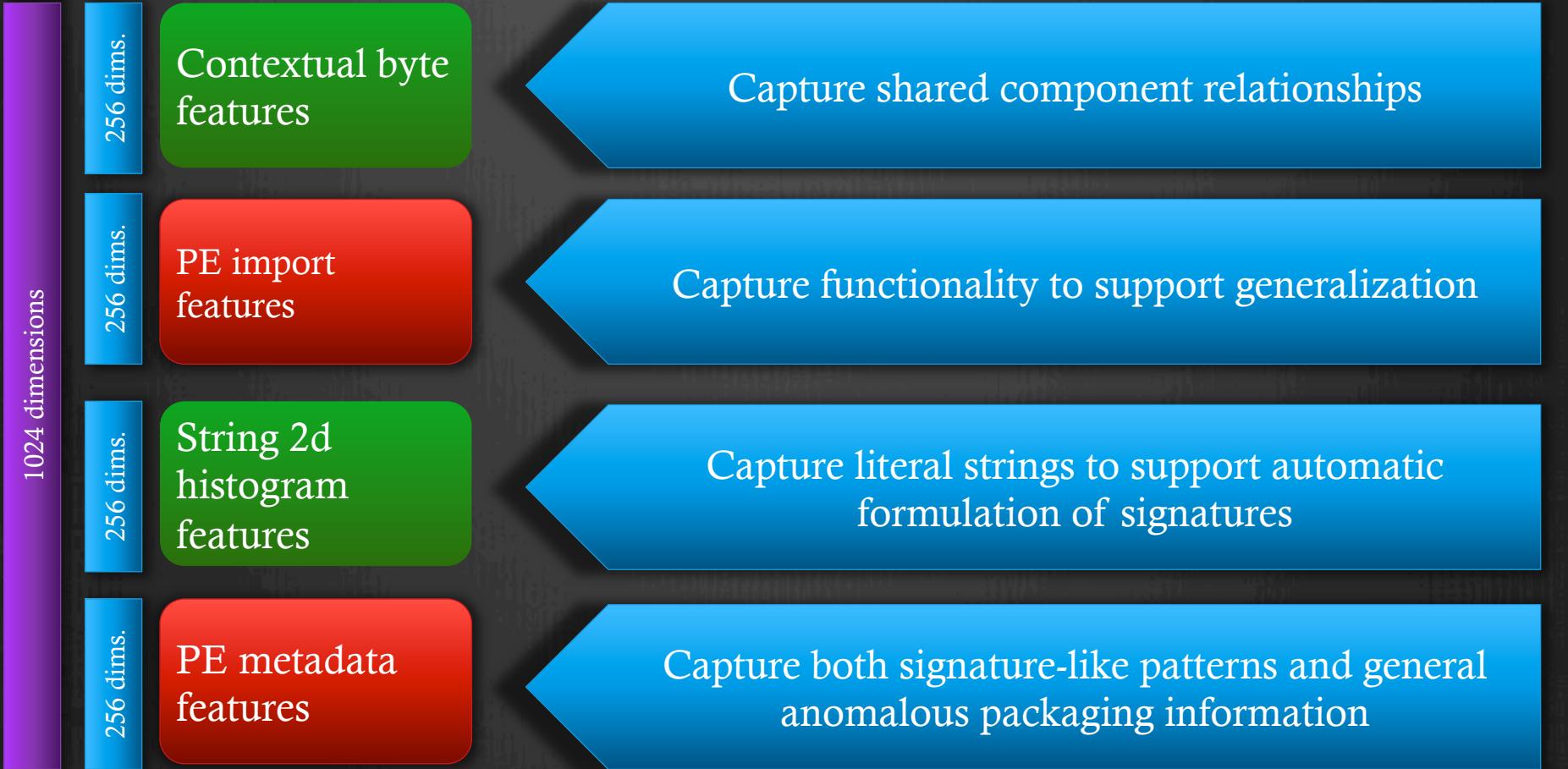
Learn general heuristics about what constitutes malware

Anti-virus analysts identify PE metadata that's abnormal or suspicious and use heuristics defined over these data to detect suspicious files

A machine learning method should automatically learn such rules as well

```
'export:directories/image_directory_entry_export/Size': 1,
'export:directories/image_directory_entry_export/VirtualAddress': 1,
'file_header/image_file_header/Characteristics': 258,
'file_header/image_file_header/Flags:IMAGE_FILE_32BIT_MACHINE': 1,
'file_header/image_file_header/Flags:IMAGE_FILE_EXECUTABLE_IMAGE': 1,
'file_header/image_file_header/Machine': 332,
'file_header/image_file_header/NumberOfSections': 4,
'file_header/image_file_header/NumberOfSymbols': 0,
'file_header/image_file_header/PointerToSymbolTable': 0,
'file_header/image_file_header/SizeOfOptionalHeader': 224,
'file_header/image_file_header/TimeDateStamp': 1332235473,
'imported_symbols/image_import_descriptor/TimeDateStamp': 0,
'load_config/image_load_config_directory/CSDVersion': 0,
'load_config/image_load_config_directory/CriticalSectionDefaultTimeout': 0,
'load_config/image_load_config_directory/DeCommitFreeBlockThreshold': 0,
'load_config/image_load_config_directory/DeCommitTotalFreeThreshold': 0,
'load_config/image_load_config_directory/EditList': 0,
'load_config/image_load_config_directory/GlobalFlagsClear': 0,
'load_config/image_load_config_directory/GlobalFlagsSet': 0,
'load_config/image_load_config_directory/LockPrefixTable': 0,
'load_config/image_load_config_directory/MajorVersion': 0,
'load_config/image_load_config_directory/MaximumAllocationSize': 0,
'load_config/image_load_config_directory/MinorVersion': 0,
'load_config/image_load_config_directory/ProcessAffinityMask': 0,
'load_config/image_load_config_directory/ProcessHeapFlags': 0,
'load_config/image_load_config_directory/Reserved1': 0,
'load_config/image_load_config_directory/SEHandlerCount': 48,
'load_config/image_load_config_directory/SEHandlerTable': 4344544,
'load_config/image_load_config_directory/SecurityCookie': 4354112,
'load_config/image_load_config_directory/Size': 72,
'load_config/image_load_config_directory/TimeDateStamp': 0,
'load_config/image_load_config_directory/VirtualMemoryThreshold': 0,
'nt_headers/image_nt_headers/Signature': 17744,
'optional_header/image_optional_header/AddressOfEntryPoint': 67186,
'optional_header/image_optional_header/BaseOfCode': 4096,
'optional_header/image_optional_header/BaseOfData': 139264,
'optional_header/image_optional_header/CheckSum': 0,
'optional_header/image_optional_header/DllCharacteristics:IMAGE_DLL_CHARACTERISTICS_DYNAMIC_BASE': 1,
'optional_header/image_optional_header/DllCharacteristics:IMAGE_DLL_CHARACTERISTICS_NX_COMPAT': 1,
'optional_header/image_optional_header/DllCharacteristics:IMAGE_DLL_CHARACTERISTICS_TERMINAL_SERVER_AWARE': 1,
```

The overall design of our feature space and how it supports these modeling goals



Our overall architecture

Feature extraction (1-2 seconds) populates a 1024 *fixed-dimensional* feature space via the *feature hashing trick*

Contextual byte
features

PE import
features

String 2d
histogram
features

PE metadata
features

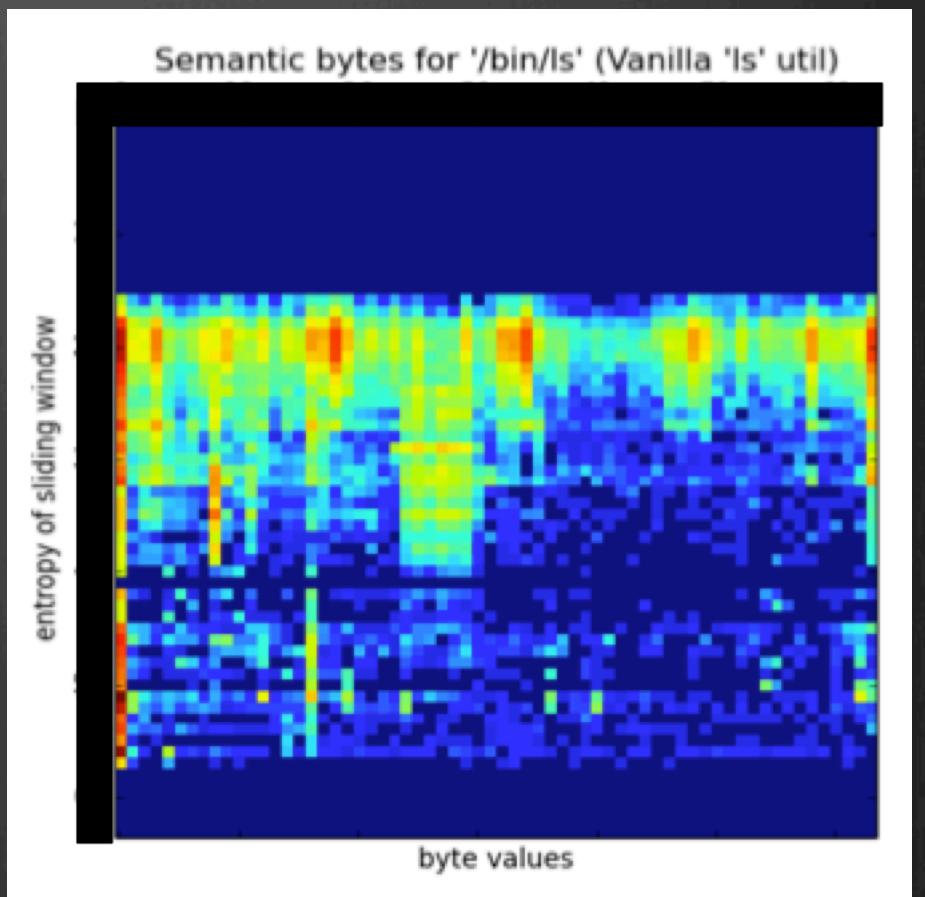
Neural network (30 minutes to train on 400k files), can be trained in streaming fashion

Bayesian calibration model: convert neural network output to principled *probability that a given file is malware*

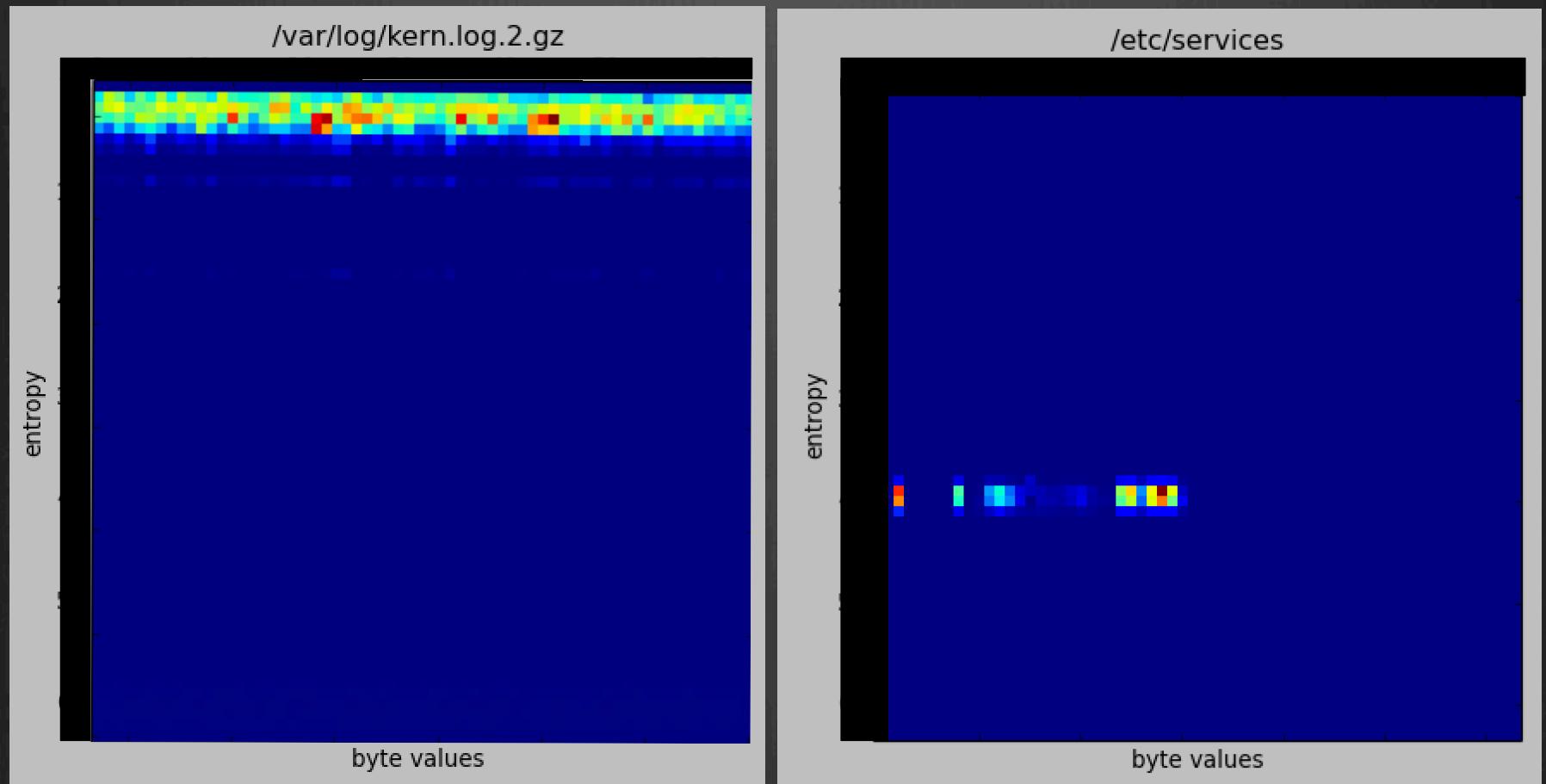
Model operationalization: 80MB model data stored in memory representing 400k training samples, supporting streaming, parallel evaluation of files' P(malware)

Contextual byte histogram features: a key component of our feature space

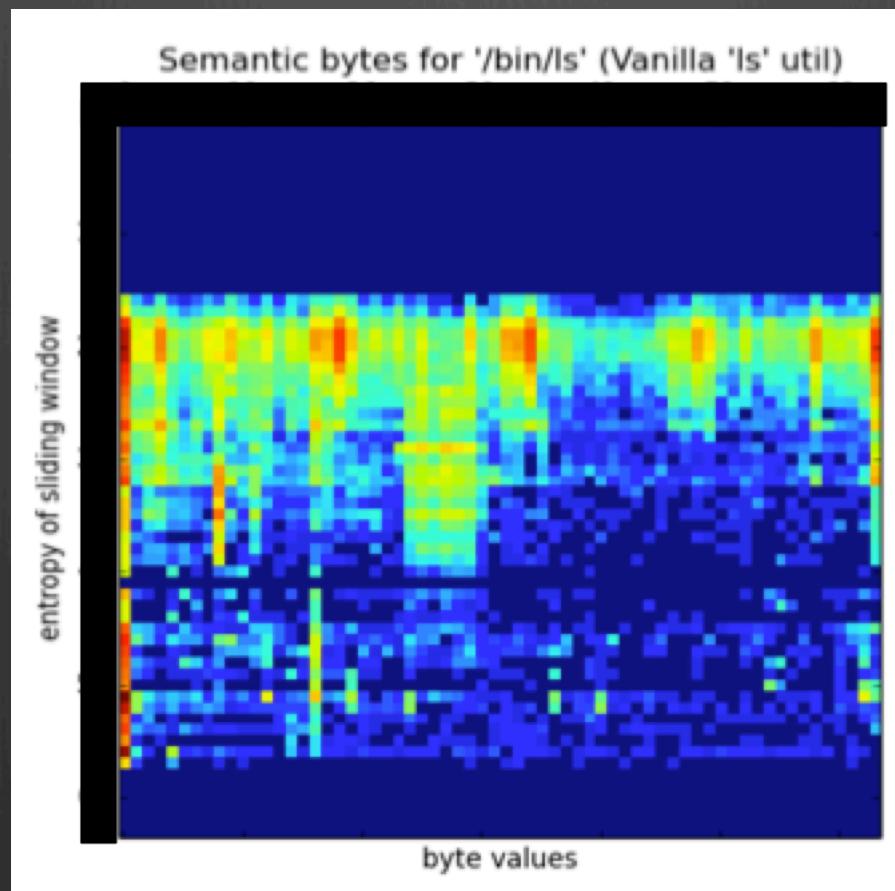
- Feature extraction algorithm:
 - Slide a 1024 byte window over the target binary, taking 256 byte steps
 - Compute the entropy of each 1024 byte window
 - For each byte in the window, store a tuple (byte, entropy)
 - Create a 2d histogram with byte values on one axis and entropy on another axis



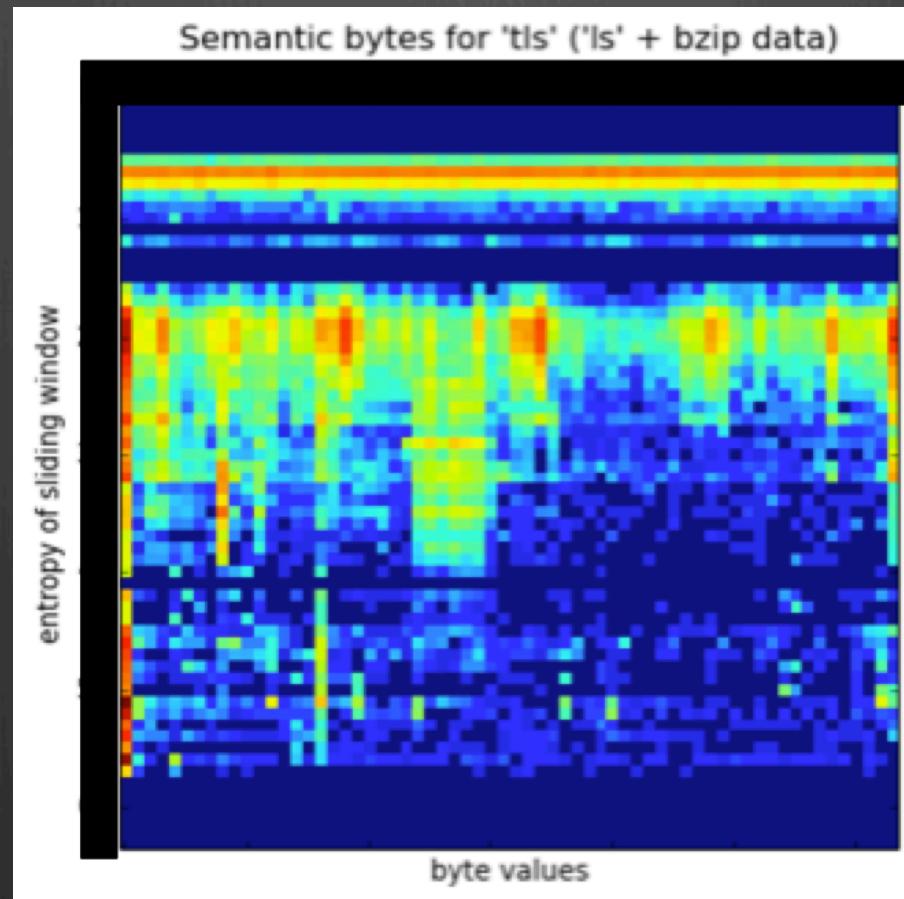
Byte/entropy histograms: a key component of our feature space

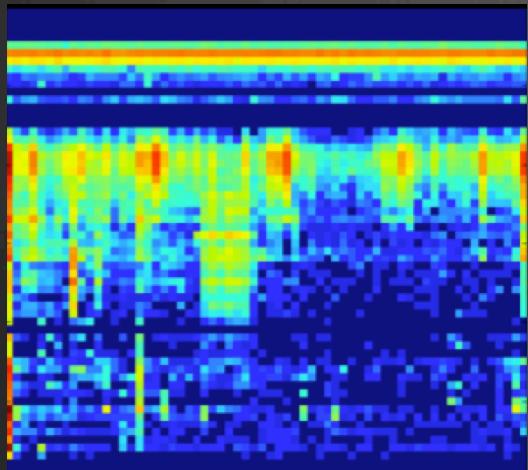


Byte/entropy representation of a binary file (benign in this example)



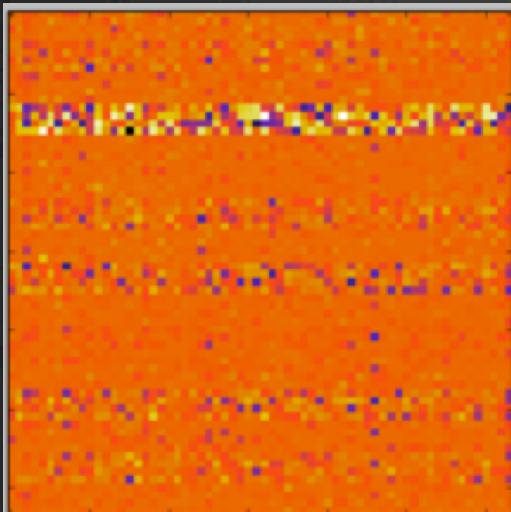
Byte/entropy representation of a binary file with a simulated component added



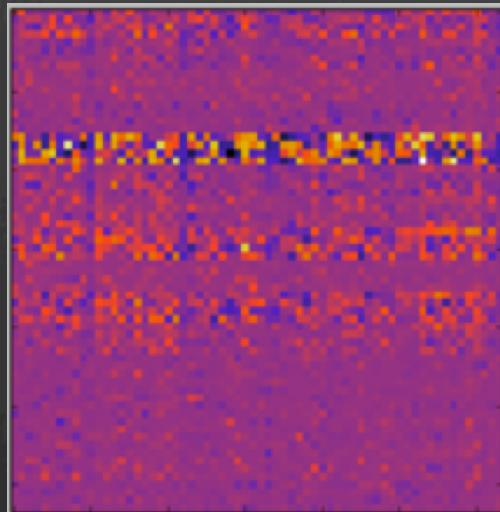


Visualizing what our neurons “learn” about the byte/entropy features

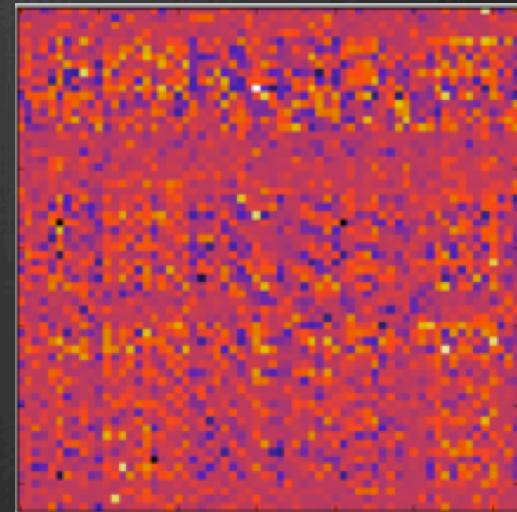
Neuron 1



Neuron 2



Neuron 3



Building a neural network that uses our features to detect malware

Feature extraction (1-2 seconds) populates a 1024 *fixed-dimensional* feature space via the *feature hashing trick*

Contextual byte features

PE import features

String 2d histogram features

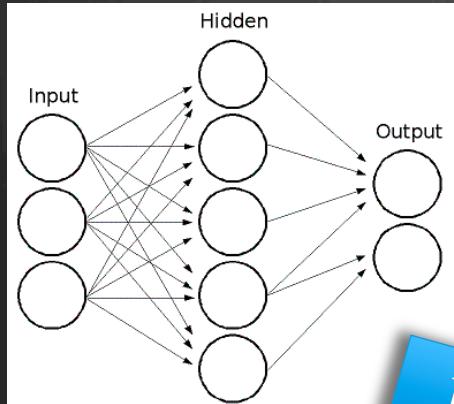
PE metadata features

Neural network (30 minutes to train on 400k files), can be trained in streaming fashion

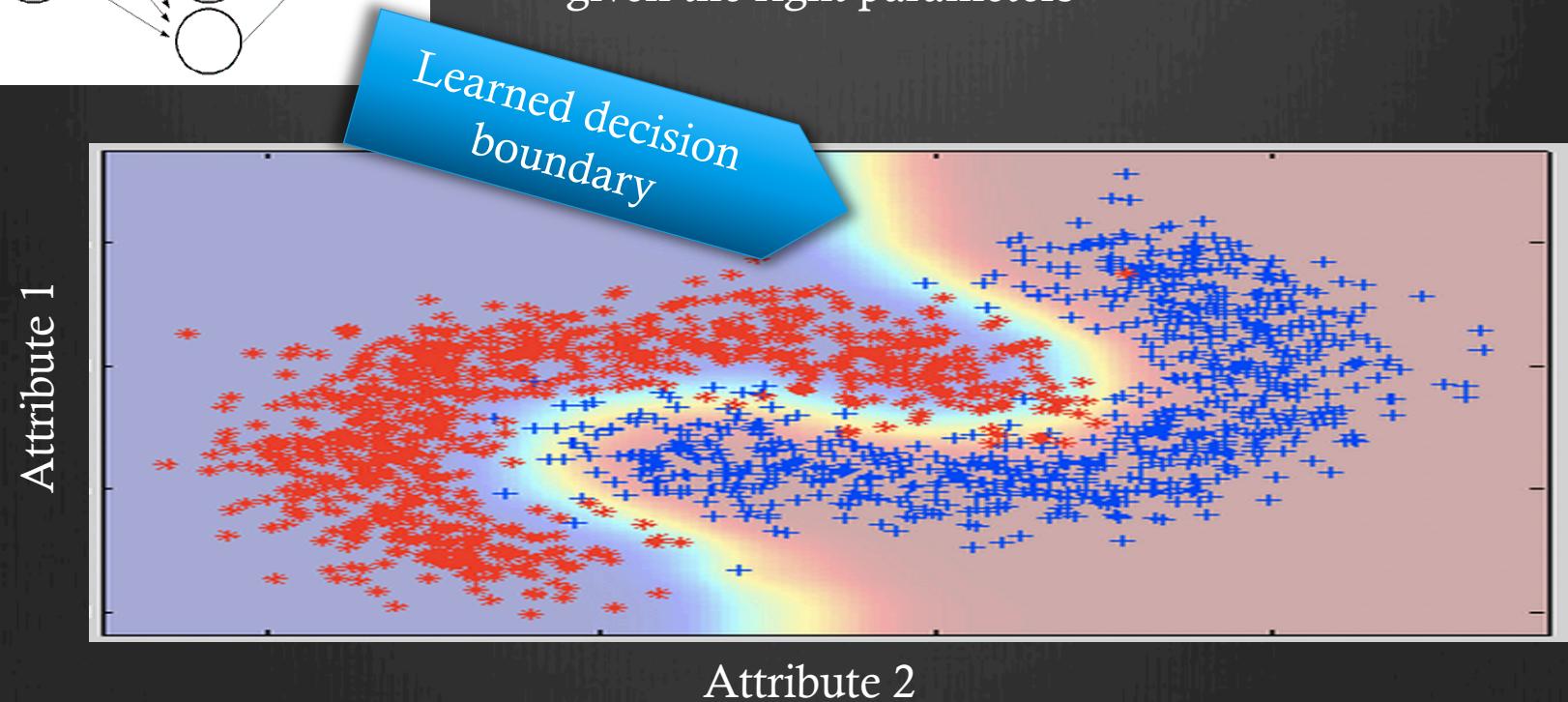
Bayesian calibration model: convert neural network output to principled *probability that a given file is malware*

Model operationalization: 80MB model data stored in memory representing 400k training samples, supporting streaming, parallel evaluation of files' P(malware)

What are neural networks?

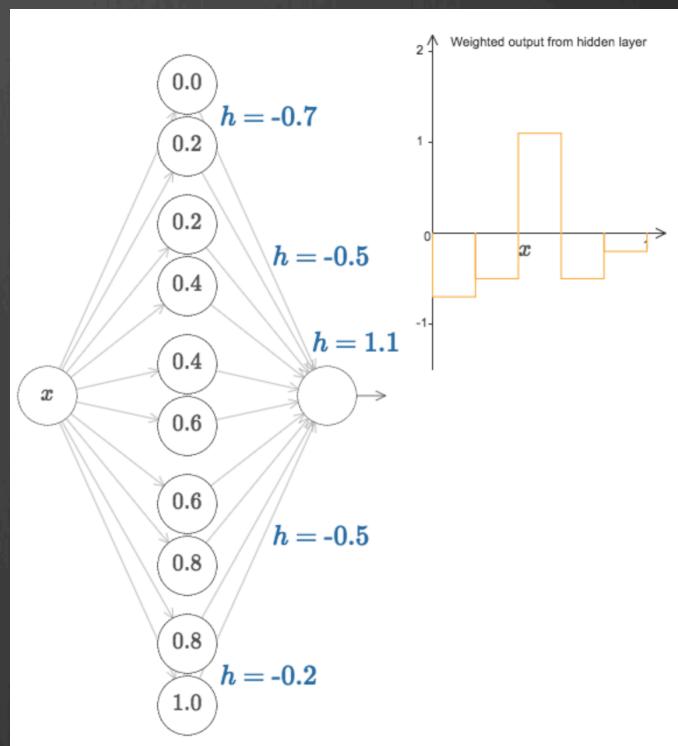


- A set of inputs
- A set of nonlinear transforms to those inputs
- A set of outputs
- This simple setup can approximate any function, given the right parameters



What kinds of functions can neural networks represent?

- Anything!
- This is even true for a simple three-layer feed forward neural network
- To get an intuition for this, think about what you can do when you add up the trigonometric wave forms

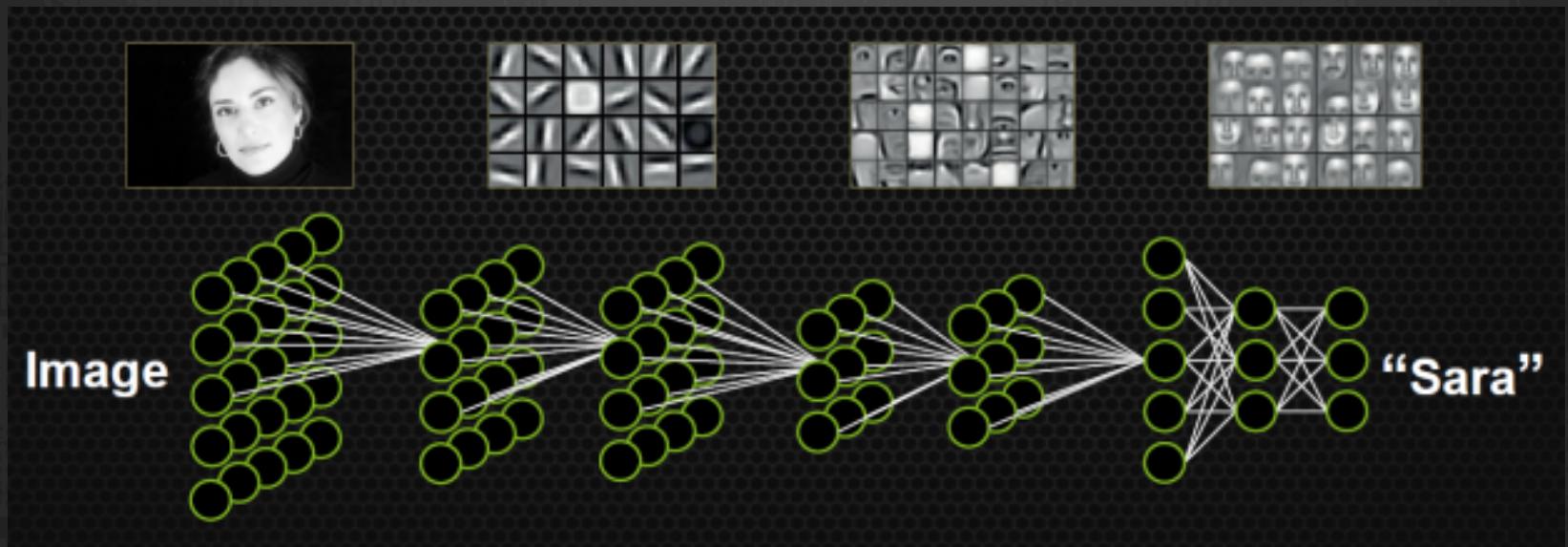


Training a neural network on 2-dimensional inputs (credit: Andrej Karpathy)

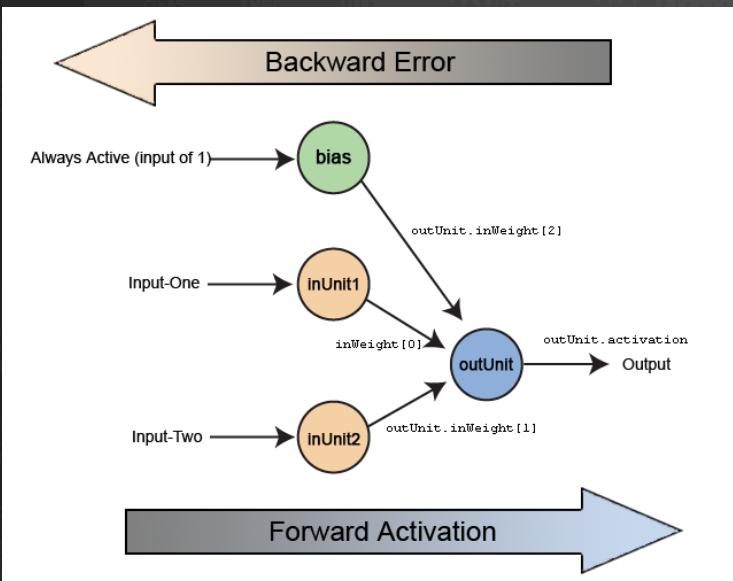


Deep networks

- ⦿ In theory these are no more expressive than a three-layer network
- ⦿ However, they seem to provide better models of the world in terms of layered abstractions



How do we train neural networks?



- “Backpropagation”: errors at the output propagate backward through all weights and biases in the network
- In practice more modern techniques are layered on top of backpropagation now (stochastic gradient descent)
- The beauty of these algorithms is that millions of algorithms can be estimated within hours

Our neural network architecture

Unbounded number of features (millions) compressed to 1024

Contextual byte
features

PE import
features

String 2d
histogram
features

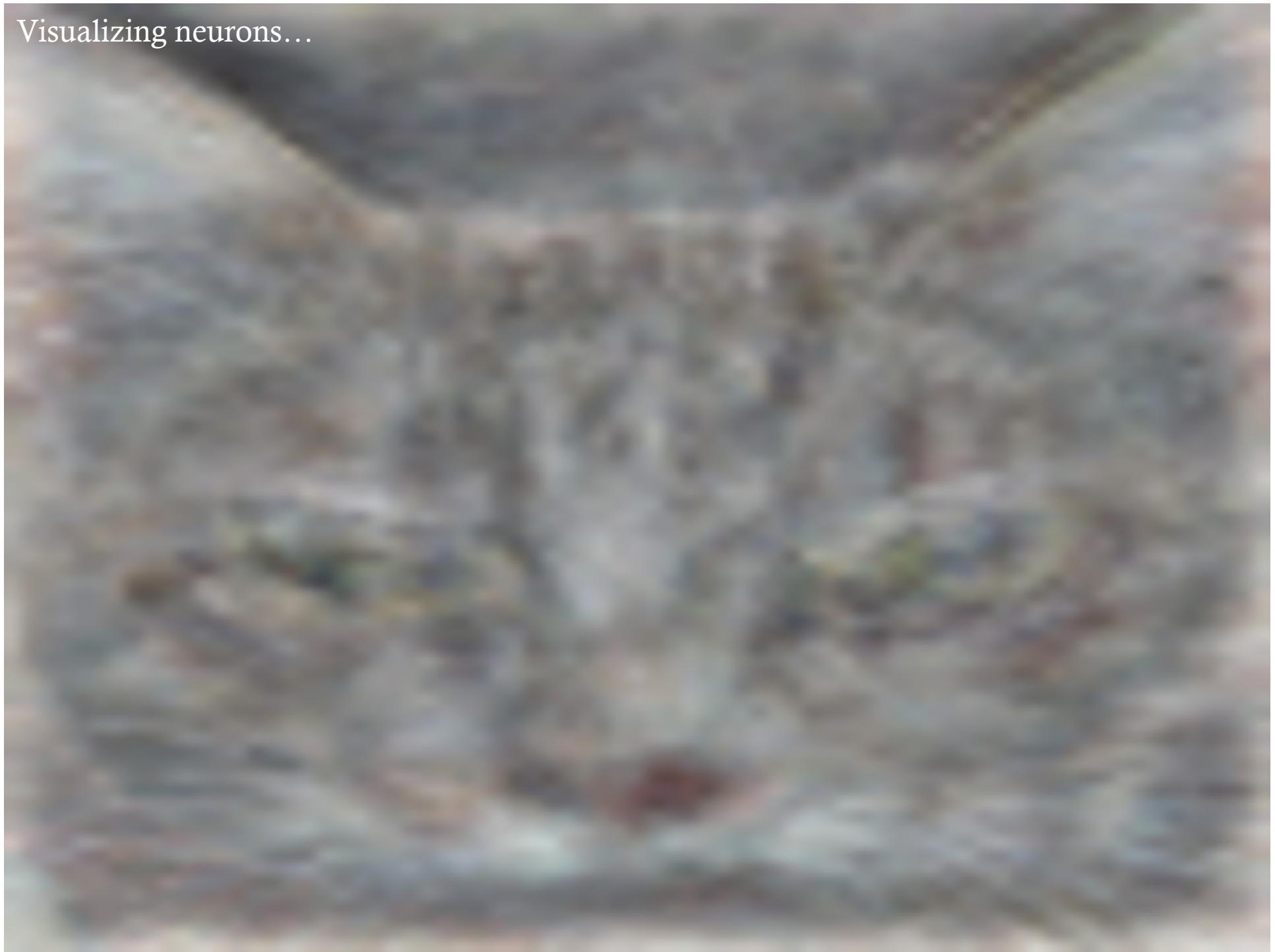
PE metadata
features

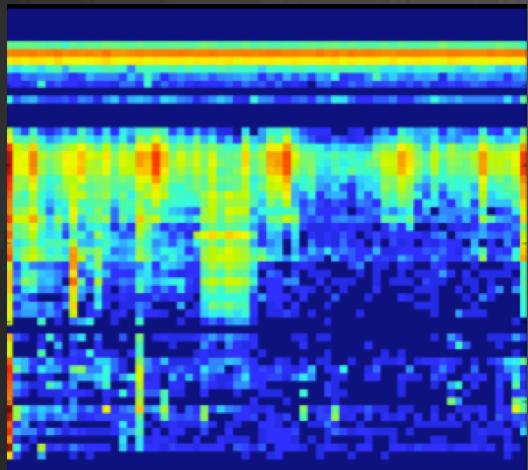
1024 neurons

1024 neurons

1 neuron

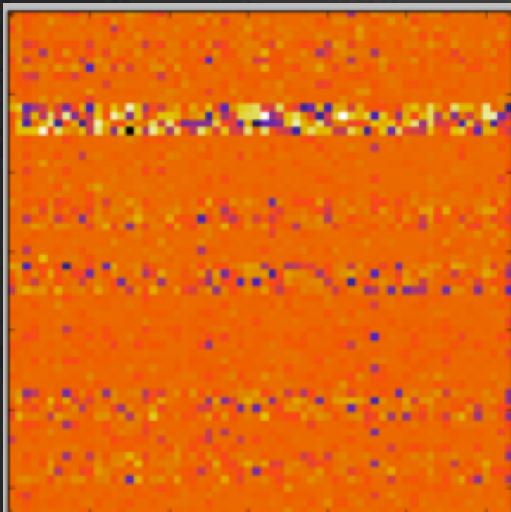
Visualizing neurons...



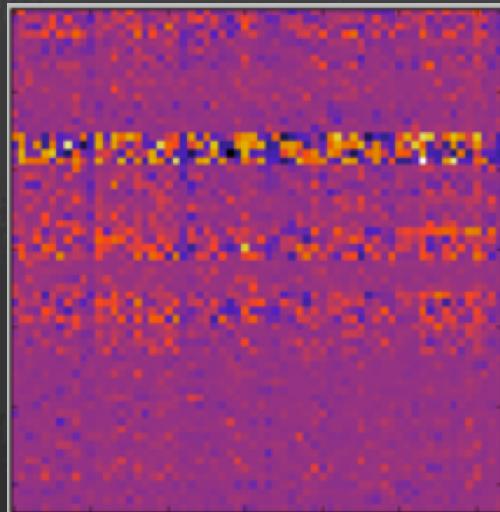


Visualizing what our neurons “learn” about the byte/entropy features

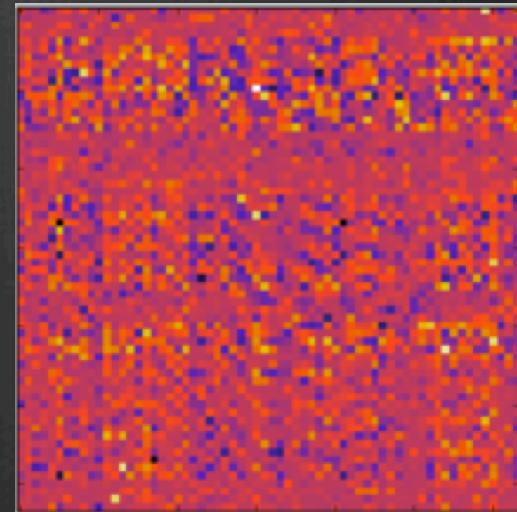
Neuron 1



Neuron 2



Neuron 3

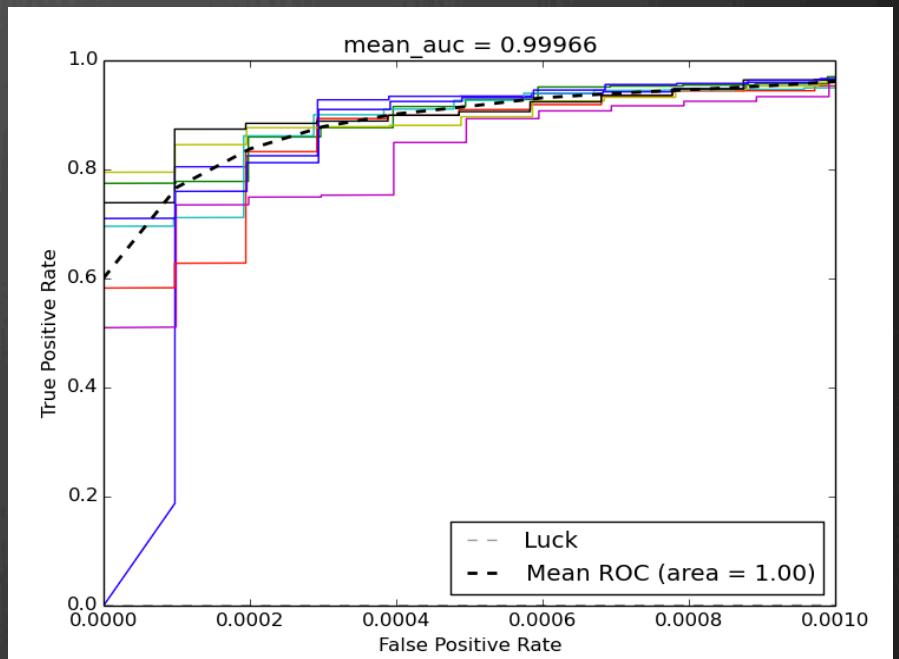


In-lab accuracy evaluation on 400k files

We can detect about 75% of malware samples our neural network has *never seen before* at a 0.01% false positive rate

We can detect 95% of malware samples that our neural network has *never seen before* at a 0.1% false positive rate

ROC curve zoomed to low FPR range



Takeaway: these results model the zero-day malware detection problem (showing detection of previously unseen malware) and are the best publicized results we know of

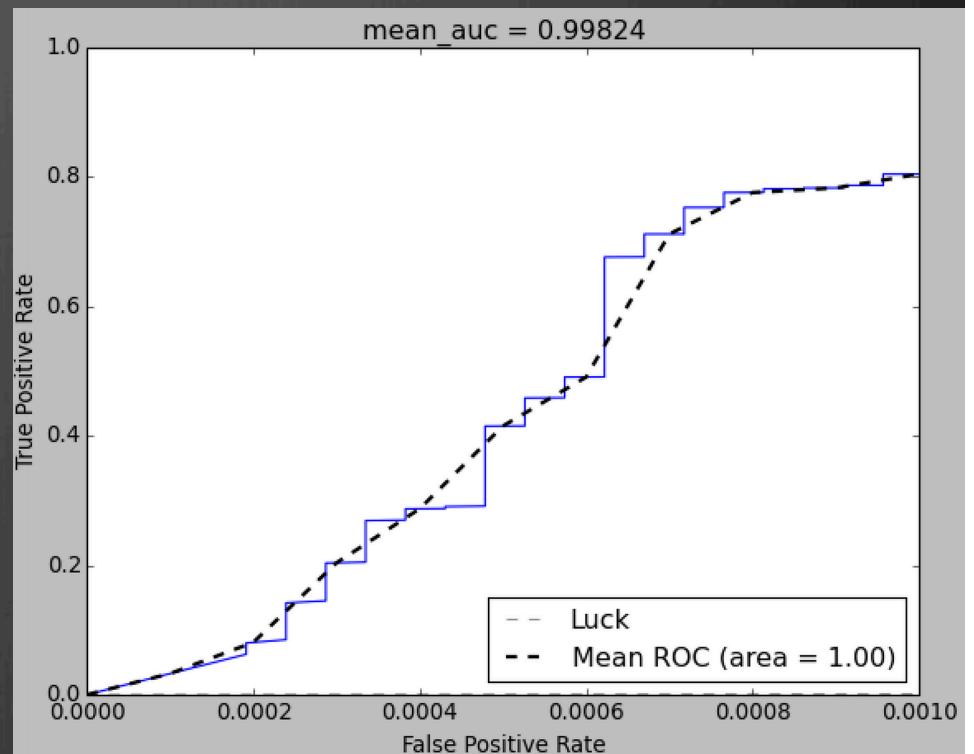
Simulating concept drift

On this test we *train* on malware with compile timestamps between 2000 and July 31st 2014

Then we evaluate our ability to *detect* malware received in our lab over the last year

The results, as you'd expect, are noticeably worse, but still pretty good!

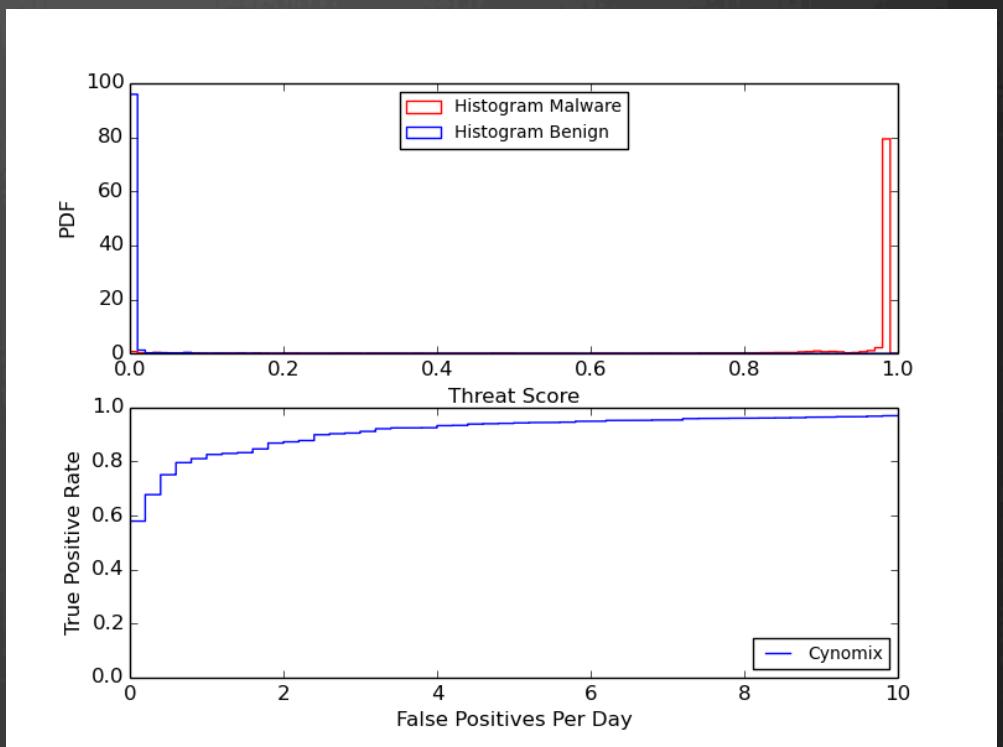
Detection rate



False positive rate

Results in an operational setting

- 60% detection rate on new malware as false positives converge on 0 (in contrast to anti-virus engines' 40%)
- 95% detection rate on new malware as false positives approach five per day
- *Test performed on feed of new binaries obtained from multiple customer networks compromising on the order of thousands of individual machines*



Case study 2. Predicting the future “success” of malware families

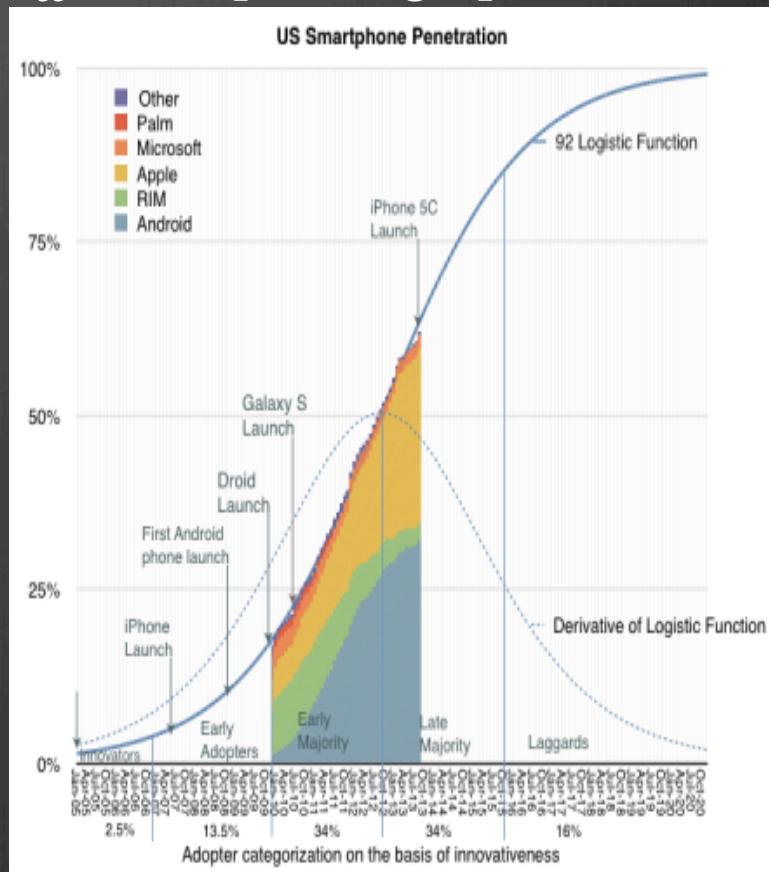
Joint work, Joshua Saxe and Giacomo Bergamo

Project overview

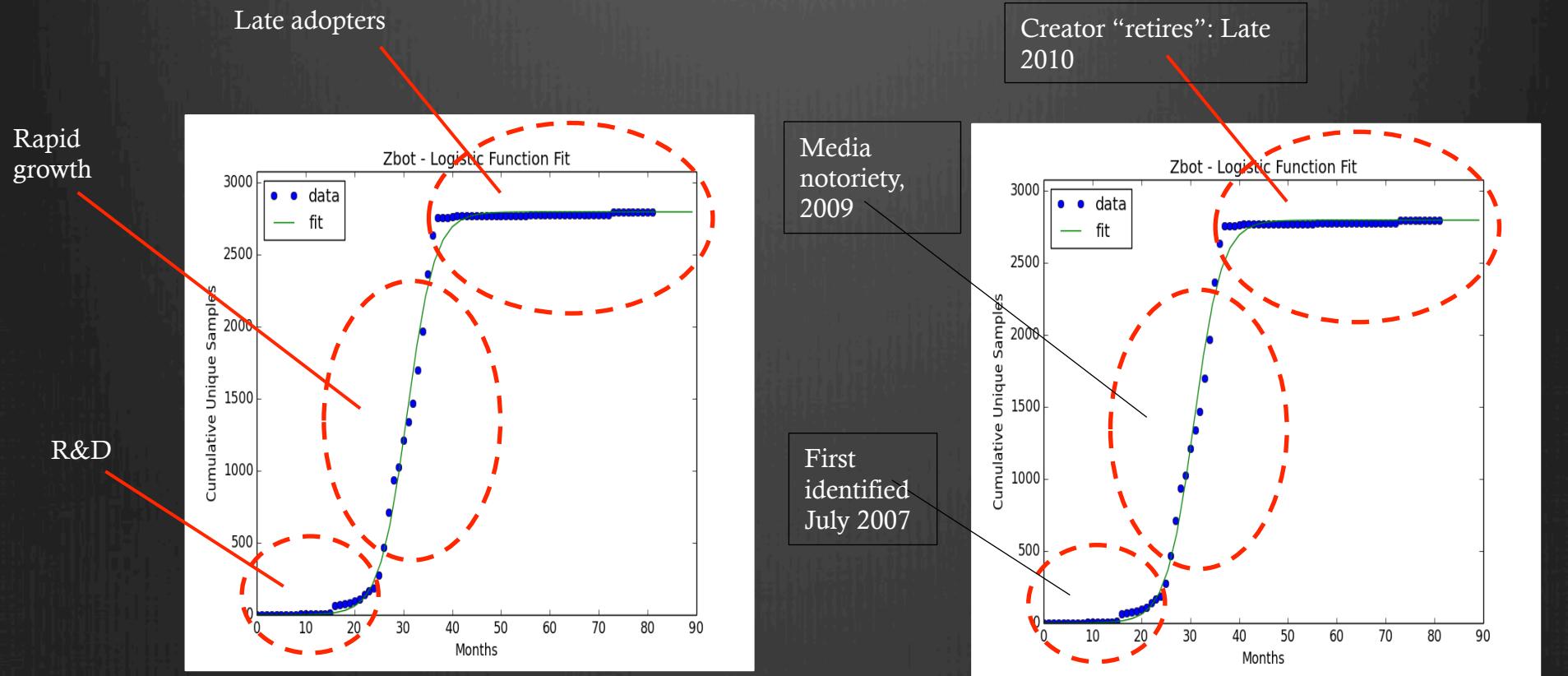
The problem: predict which malware families will grow explosively

- ⦿ New malware families (such as Koobface, Storm and Zeus) become targets for robust detection and disruption after they have achieved widespread criminal adoption
- ⦿ Can we predict this *in advance*?
- ⦿ Payoff:
 - ⦿ Focus our reversing and countermeasure efforts on malware likely to achieve breakaway success

Our approach: adapt models of technology diffusion to predicting explosive malware

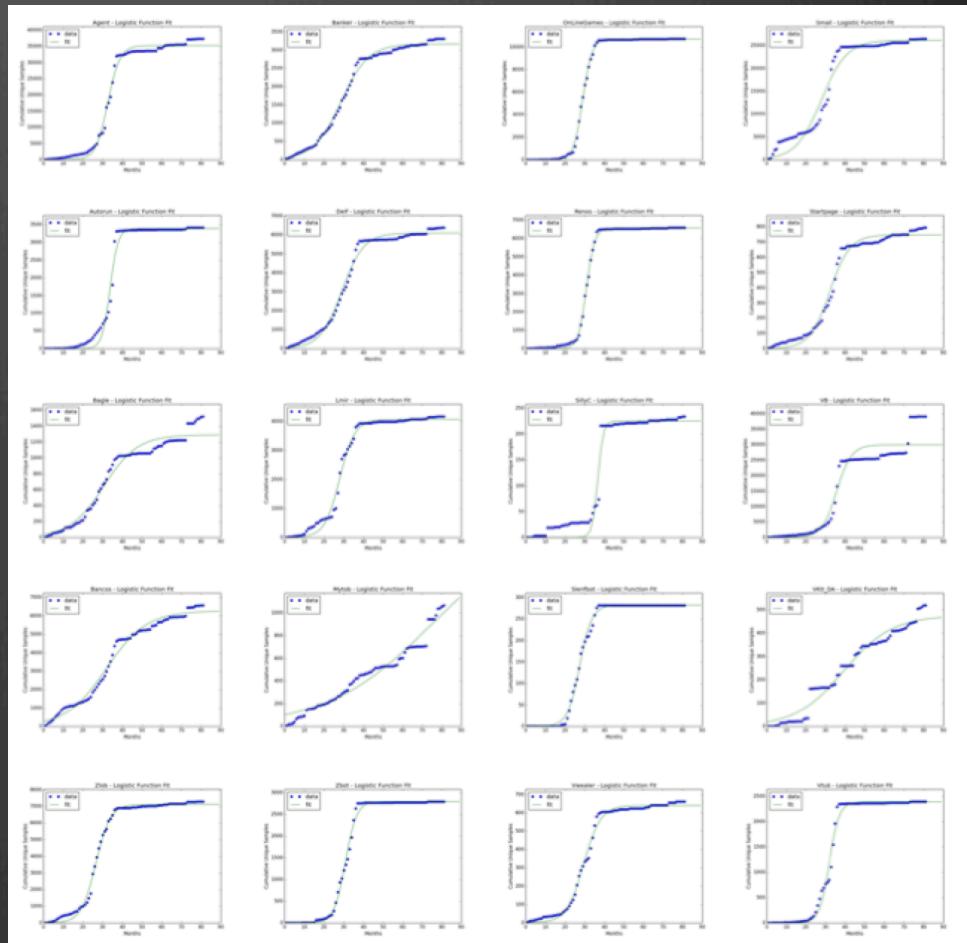


Exemplar malware family: Zbot / Zeus



Bird's eye view of 20 malware families 'goodness of fit'

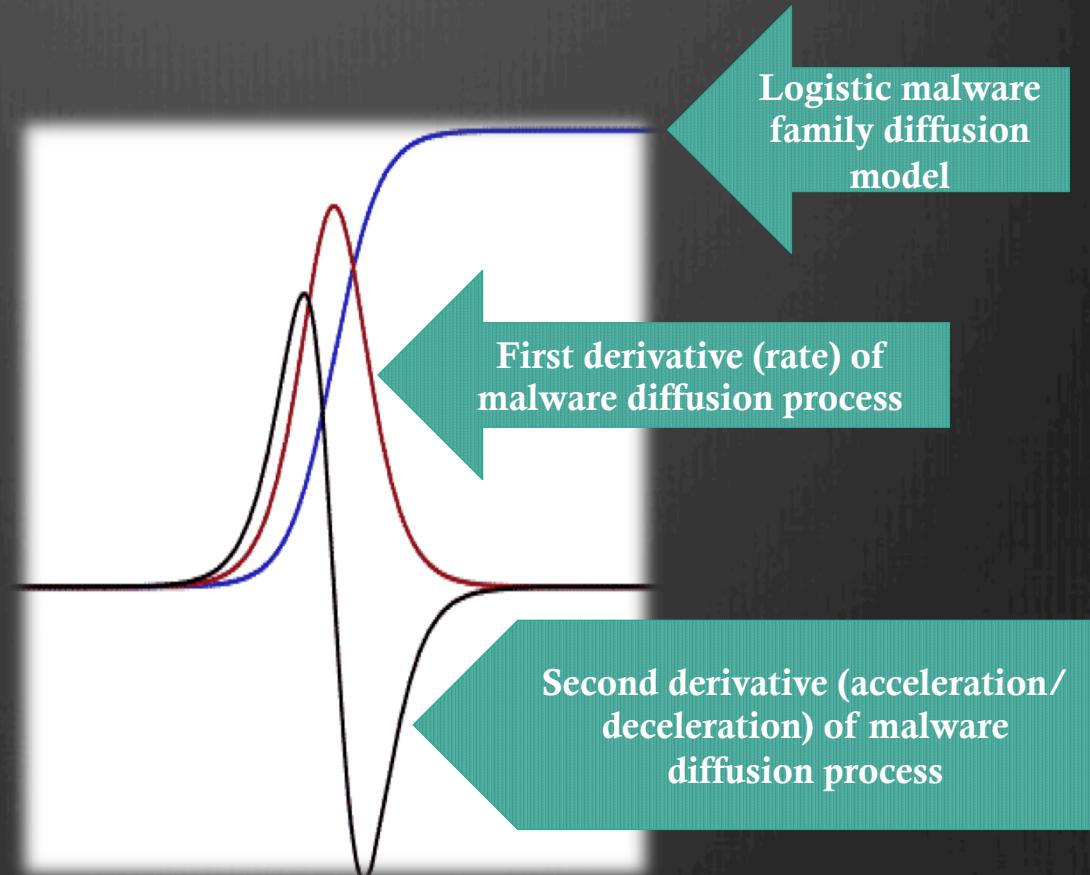
- Considerable variation in shape of data
 - Still tends to fit a parametric ‘S-shape’
 - Given this variation how well can we predict explosion and taper?



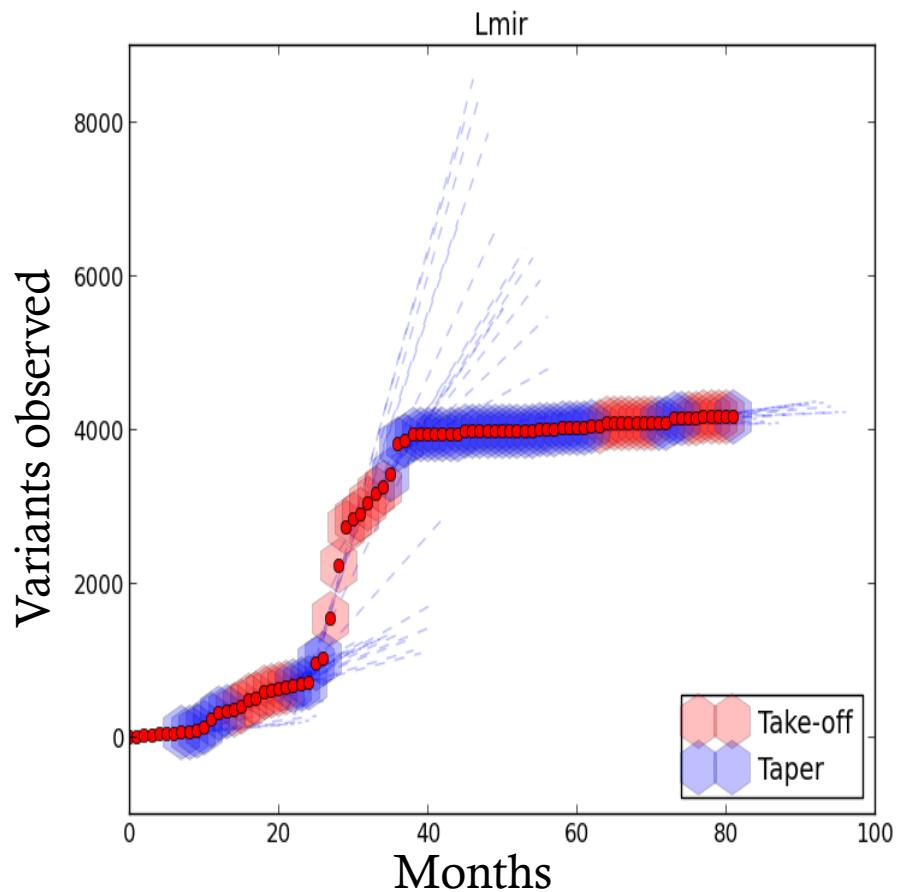
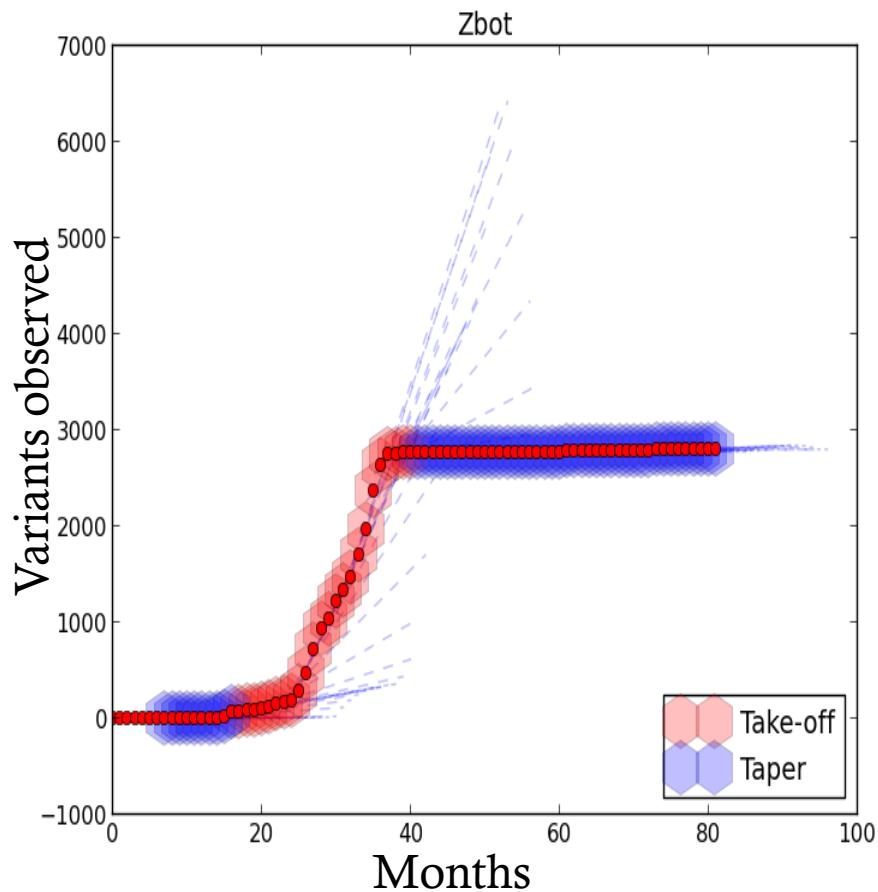
Predictive modeling approach

Key modeling points:

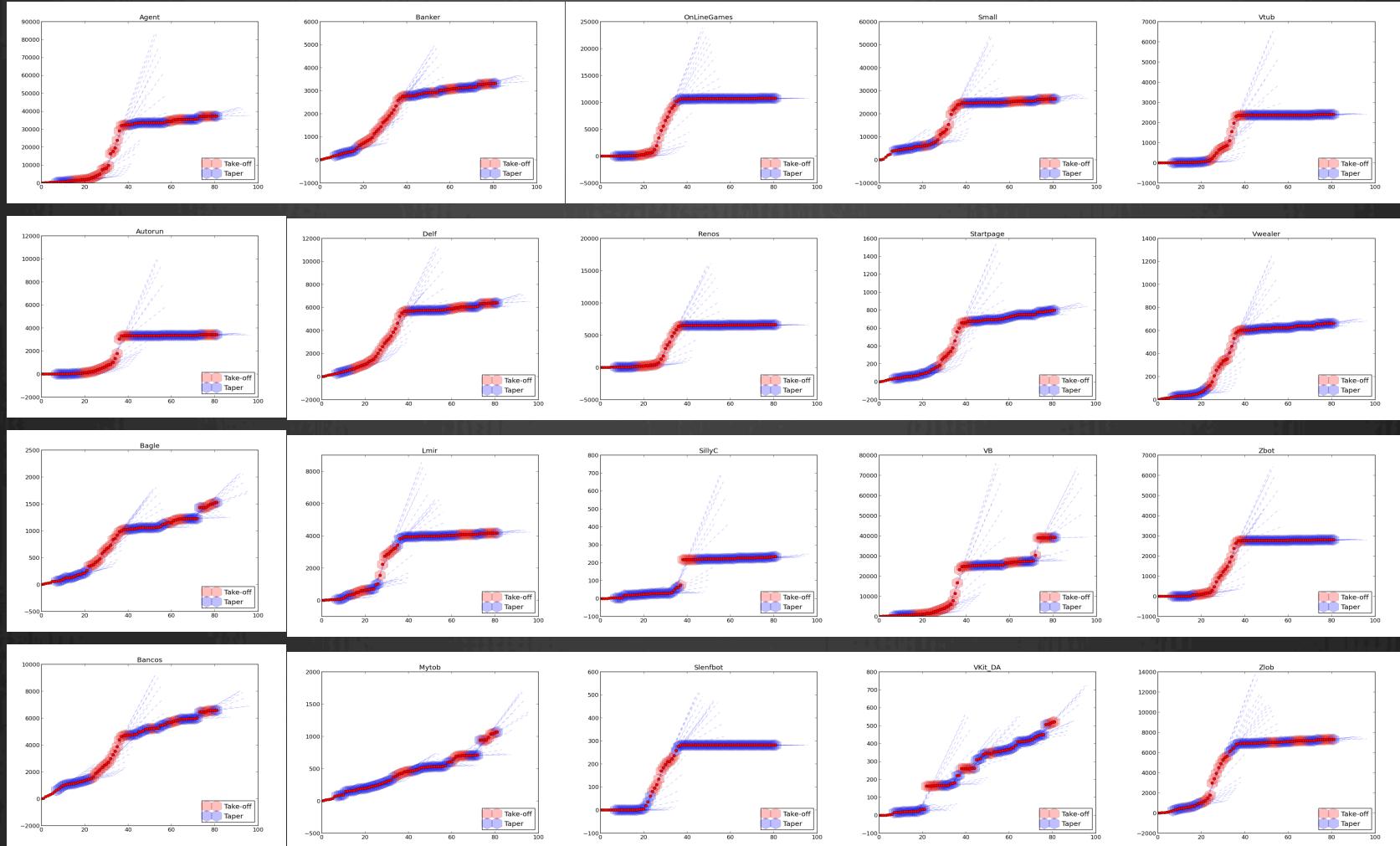
- We want to observe how many new variants we're observing for each family in each discrete time bin (say, per-month)
- Use linear regression to calculate month by month speed of new variant production
- Second order regression measures acceleration / deceleration in this process
- *Threshold on acceleration determines if takeoff has been entered*



A good case and a bad case: Lmir and Zbot

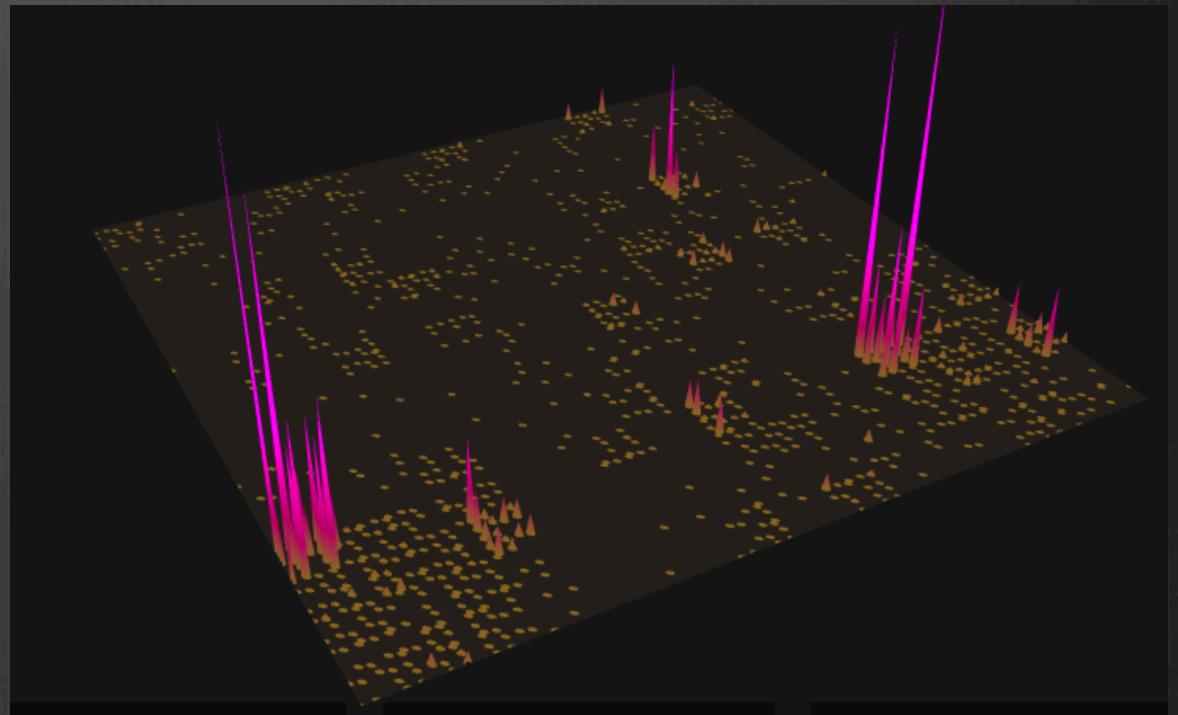


Results on all 20 of our malware families



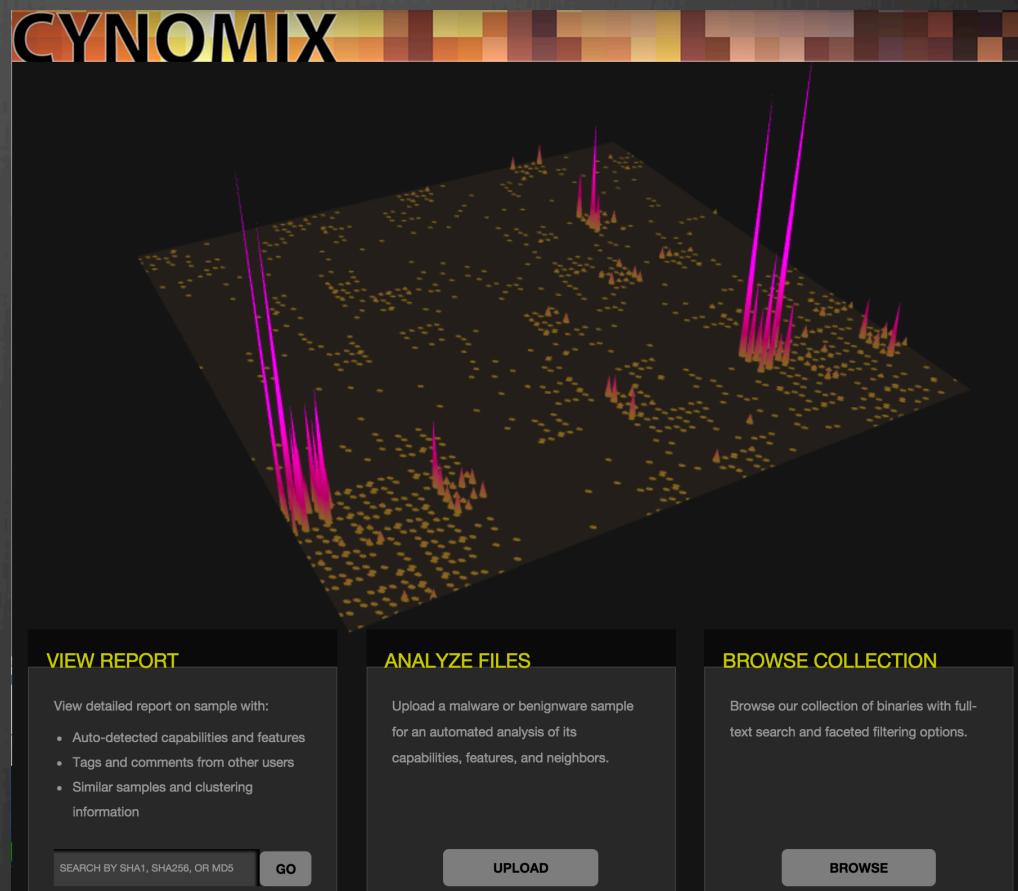
Case study 3: A machine learning- aided malware analysis workbench

*Alex Long, Robert
Gove, Joshua Saxe,
Sigfried Gold, Giacomo
Bergamo, Aaron Liu*



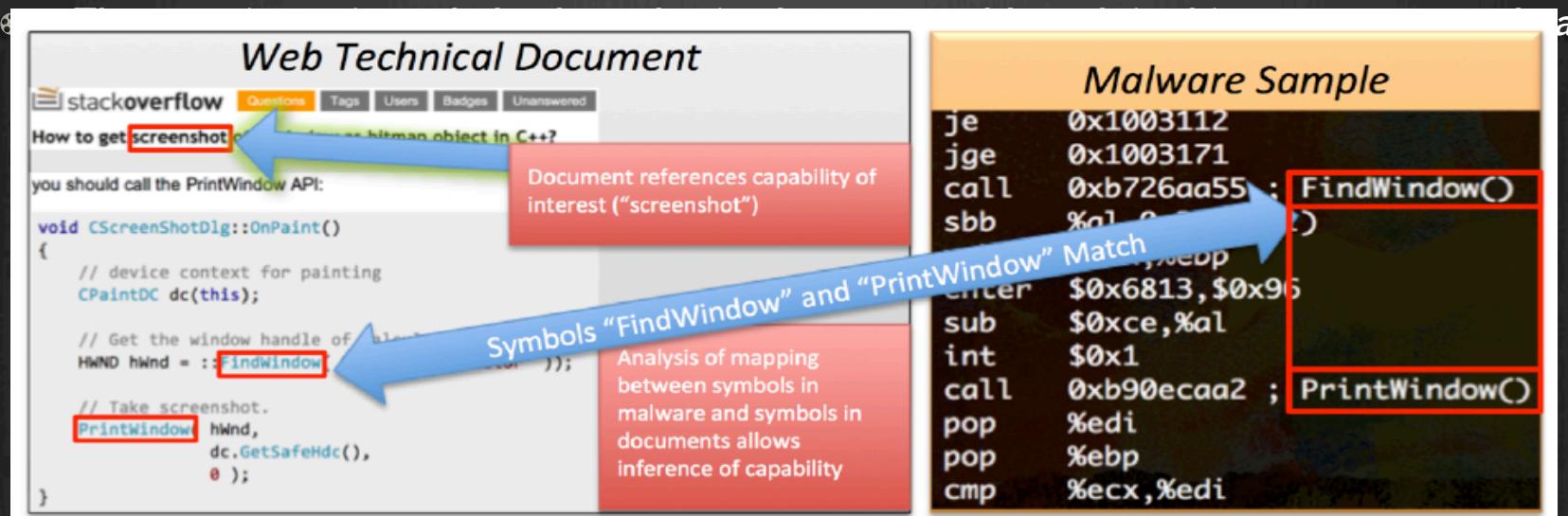
Capabilities of our research prototype ("Cynomix")

- ⦿ Performs automatic capability recognition by “learning” from StackOverflow
- ⦿ Builds a “social network” of malware samples based on shared code relationships



Premise of automatic capability detection work

- The Internet is rife with text that combines example code with natural language description of its functionality
- It might be possible to piggyback on the expert knowledge of the web “crowd,” which holds more knowledge than the mind of any one malware subject matter expert



Overall, we found the approach works

- Detecting malware capabilities based on a text mining approach leveraging crowdsourced data yields good results some of the time
- Requires unpacked malware
- Detections are still somewhat noisy
- But, good enough to be adopted by and benefit practitioners
- *Rest of this talk adds more detail, describing methods and results*

Document datasets we used

- Four textual datasets including ~6 million Q&A threads
- This Q&A data is made publicly available as an XML dump by StackExchange
- Approach could accommodate new datasets such as malware technical reverse engineering reports, malware forums



Ask Ubuntu
Q&A for Ubuntu users and developers

questions 111k
answers 143k
answered 71%
users 131k

"Are OpenID logins compromised with the Ubuntu Forums breach?"
– asked 6 hours ago



Server Fault
Q&A for professional system and network administrators

questions 156k
answers 291k
answered 82%
users 132k

"Should I install vanilla ESXi or the Dell customised version on a new cluster of R620?"
– asked 6 hours ago



Super User
Q&A for computer enthusiasts and power users

questions 181k
answers 307k
answered 77%
users 182k

"What is the exact use of a MAC address?"
– asked 12 hours ago

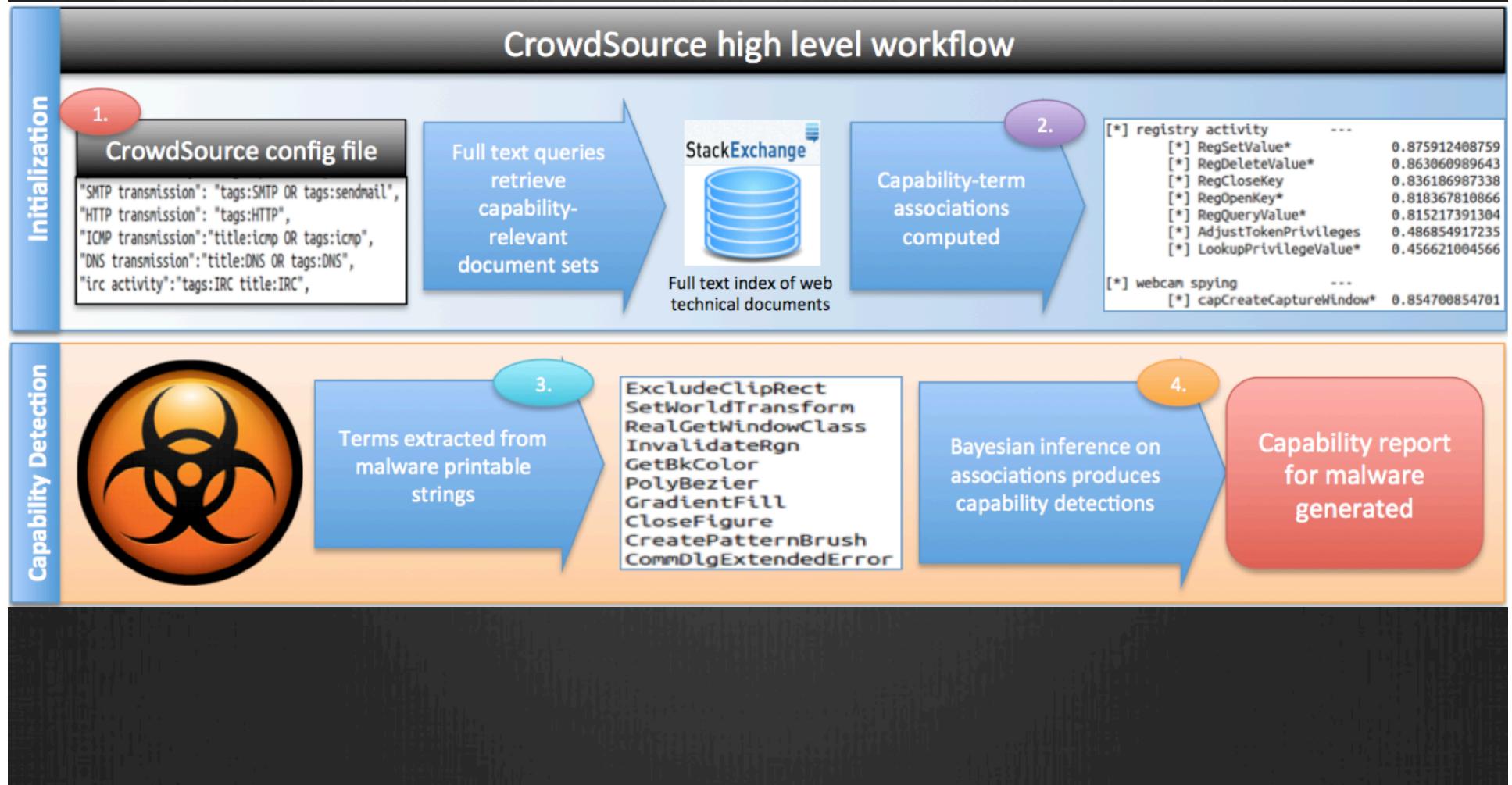


Stack Overflow
Q&A for professional and enthusiast programmers

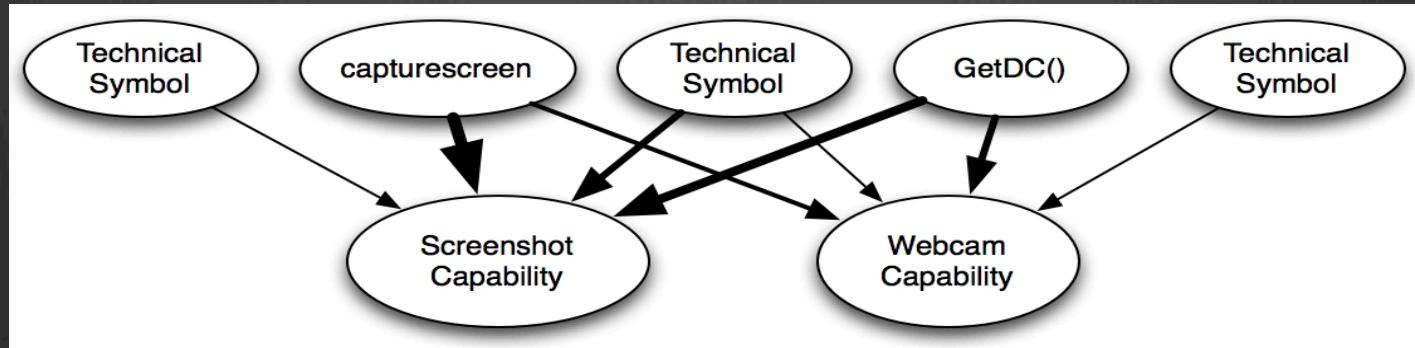
questions 5.4m
answers 9.9m
answered 78%
users 2.2m

"How to Find the Max Integer Value in a Stack without using max() or iterating over it?"
– asked 3 hours ago

Our approach to capability-detection



Using Bayesian networks with “Noisy-OR” to infer capabilities



- Most clearly interpreted as a one-layer Bayesian network with Noisy-OR gate edges
- Parameters can be quickly estimated in closed form using document frequencies combined with prior belief; we use the Beta-Bernoulli model
- For equations, read the paper!

Our model “learns” what symbols are associated with what malware capabilities, using these associations to explain its detections to analysts

```
[|] category: gui
[*] low level GUI calls / screenshot (somewhat likely)
[*] Extracted from sample: Post title on StackOverflow

[-] 'BitBlt' Capturing screenshot
[-] 'CreateCompatibleDC' Capturing screenshot
[-] 'CAPTURE]: Screen' C: take screenshot
[-] 'DeleteDC' c# Code for capturing Full ScreenShot
[-] 'SelectObject' Capturing screenshot
[-] 'screen. [CAPTURE, screen capture, Screen' Capturing screenshot
[-] 'gdi32, Gdi32' C# rectangle drawing and screenshot
[-] 'GetDeviceCaps' c# Code for capturing Full ScreenShot
[-] 'Gdi32.dll, gdi32.dll' Desktop screenshot in WPF
[-] 'CreateDIBSection' Capturing screenshot
[-] 'CloseClipboard' Screenshot program not working
[-] 'frame [CAPTURE' Kinect take screenshot with pictures overlayed
[-] 'GetForegroundWindow' C# - Capture screenshot of active window
[-] 'OpenClipboard' Screenshot program not working
[-] 'GetClipboardData' Screenshot program not working
[-] 'User32' Window screenshot using WinAPI
[-] 'Screen' Sending screenshot via C#
[-] 'capturing' Capturing screenshot
```

Our approach can also explain to analysts where in the web technical documents it found evidence for a prediction

```
[*] low level GUI calls / screenshot (somewhat likely)
[*] Extracted from sample: Post title on StackOverflow

[-] 'capturing screen' Capturing screenshot in iphone?

[|] s code but it not working for me .... it gave me same result. :( upstate - hello RRB:
[|] this code is working into my application for capture the screenshot of a iphone. -
[|] can i give the demo for that - actually i want to create video of game play using
[|] capturing screen shots along with sound. I want functionality like talking
[|] tom application. - you should probably take a look at AVFoundation and specifically
[|] AVAssetWriter as a way of creating videos of your screen content. - Take the
[|] screenshot of yo

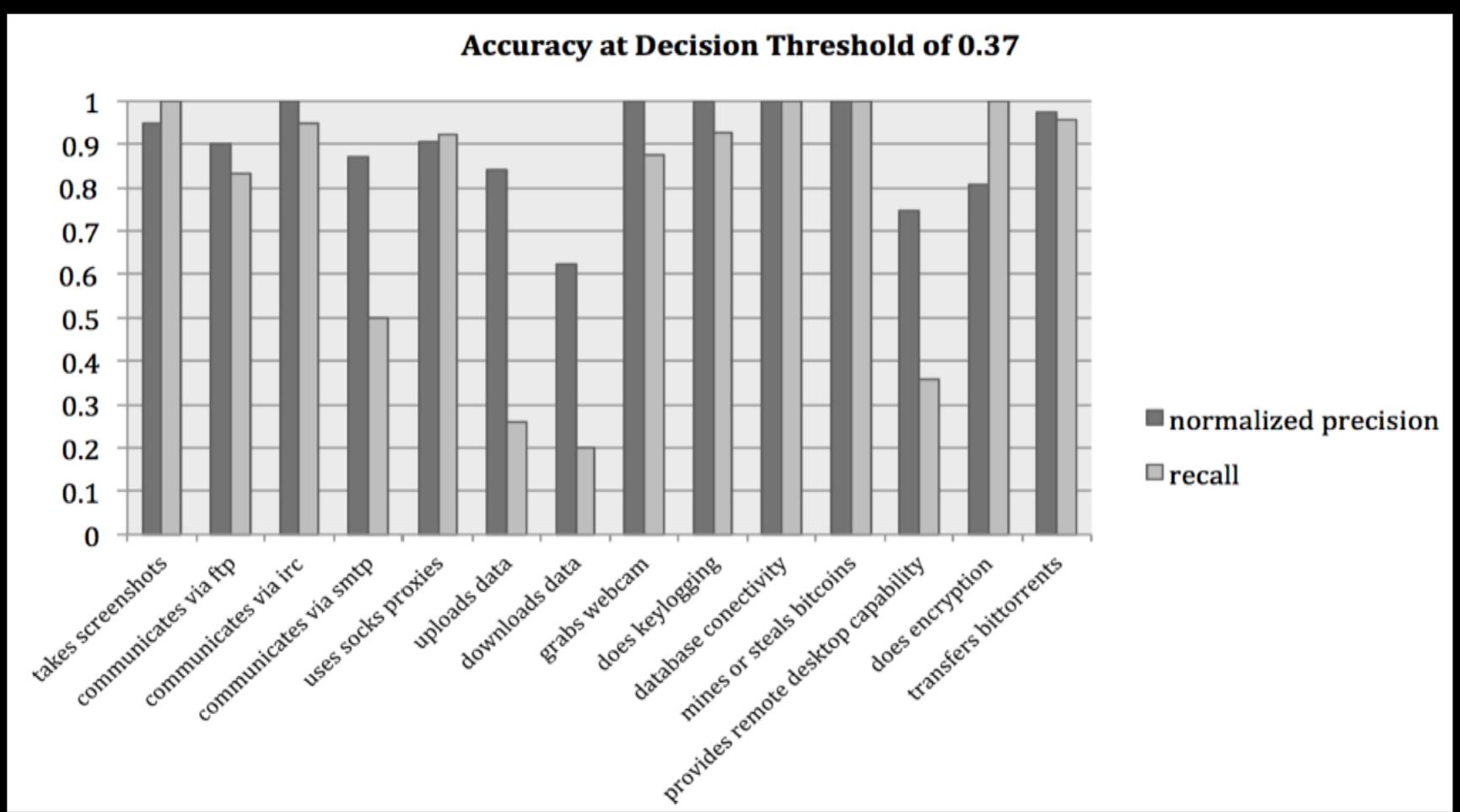
[-] 'BitBlt' Capturing screenshot

[|] // bitmap handle HBITMAP hbitmap = CreateCompatibleBitmap(hdc_screen, bounds.width,
[|] bounds.height); // select the bitmap handle SelectObject(hdc_memory, hbitmap); //
[|] paint onto the bitmap BitBlt(hdc_memory, bounds.x, bounds.y, bounds.width,
[|] bounds.height, hdc_screen, bounds.x, bounds.y, SRCPAINT); // release the screen DC
[|] ReleaseDC(NULL, hdc_screen); // get the pixel data from the bitmap handle and put it

[-] 'CreateCompatibleDC' Capturing screenshot

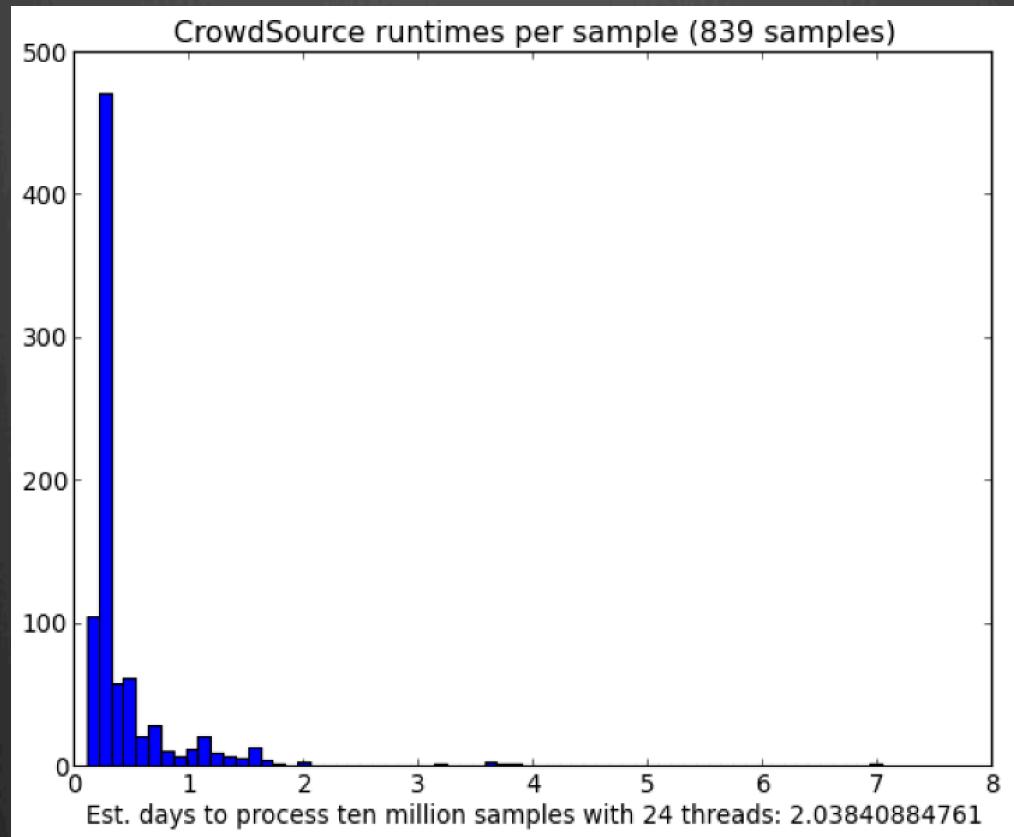
[|] void get_screenshot(COLORREF** img, const Rectangle &bounds) { // get the screen DC
[|] HDC hdc_screen = GetDC(NULL); // memory DC so we don't have to constantly poll the
[|] screen DC HDC hdc_memory = CreateCompatibleDC(hdc_screen); // bitmap
[|] handle HBITMAP hbitmap = CreateCompatibleBitmap(hdc_screen, bounds.width,
```

Results analysis on 14 capabilities



Results analysis on 14 capabilities

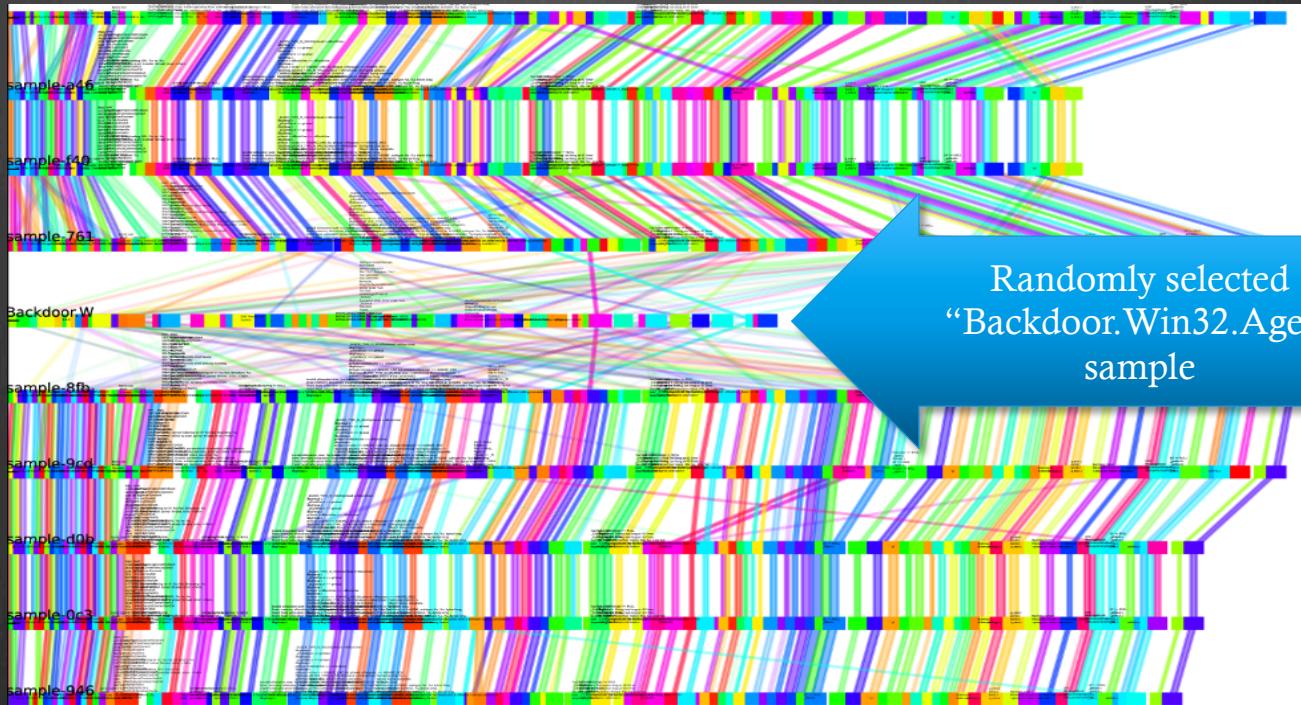
Running time on the order of ~200ms per sample



Building the malware “social network”: identifying shared code between malware samples

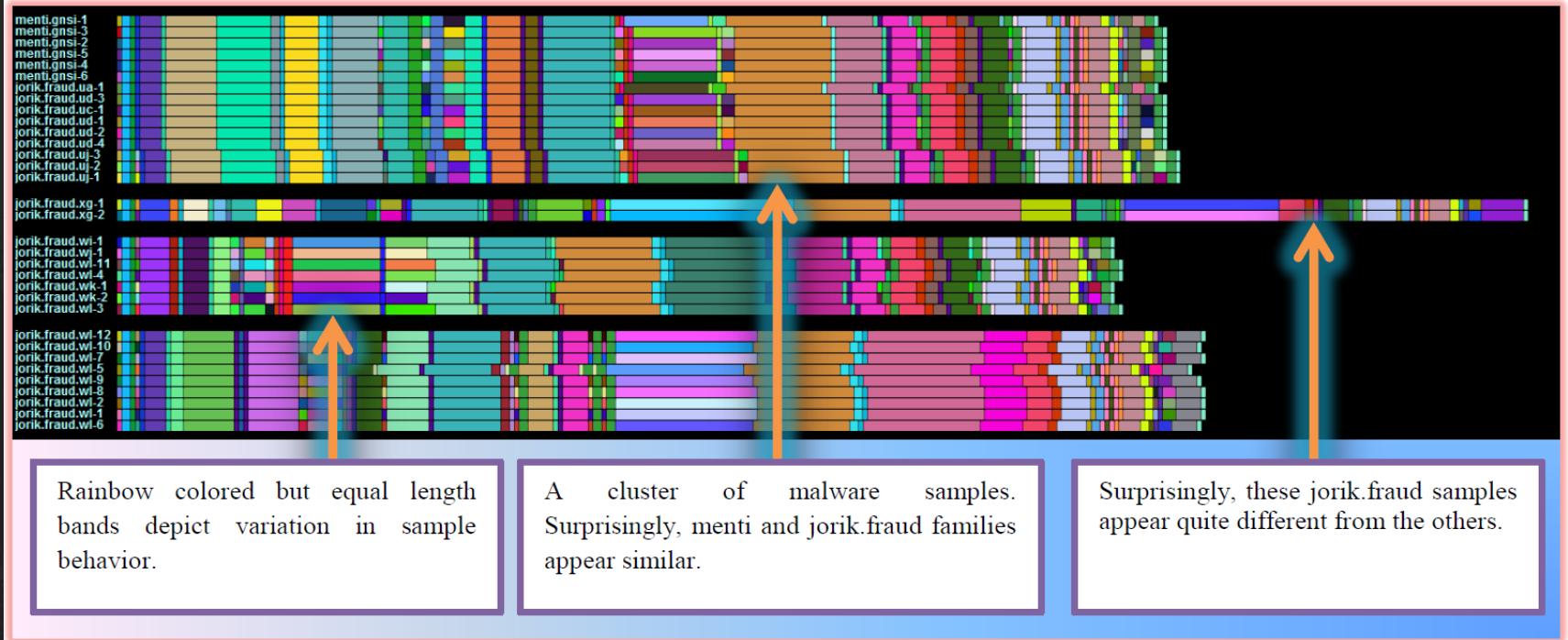
Members of
the Kbot
malware
family

Members
of the
Kbot
malware
family



The approaches we have developed discover genealogical relationships between millions of malware samples based on identification of shared attributes

Dynamic similarity analysis



Demonstration / guided tour of
malware “social network”
visualization and malware capability
recognition interface

Conclusion

Takeaways / Sound Bytes

- ⦿ Security data science holds the promise of shining a spotlight into the reams of “dark matter” security data that we’re currently not exploiting, thereby changing the game in network defense
- ⦿ Security data science is different: it requires ultra-low false positive rates, interpretability, and adversarial modeling
- ⦿ As data science advances, security will not be unaffected – there is every reason to think that data science will disrupt security just as it has advertising, human computer interface design, and computer vision

Security data science: Where we are in the innovation cycle



Credit: Gartner

Get involved!

- ➊ Security data science is a new field in which security professionals of all backgrounds can make an impact
- ➋ If you're new to data science, pick up scikit-learn, networkx, numpy, theano, R, or any of the other open source tools, along with a good data mining textbook, and start hacking
- ➌ If you're new to security, work with security professionals to find data science applications that make sense

