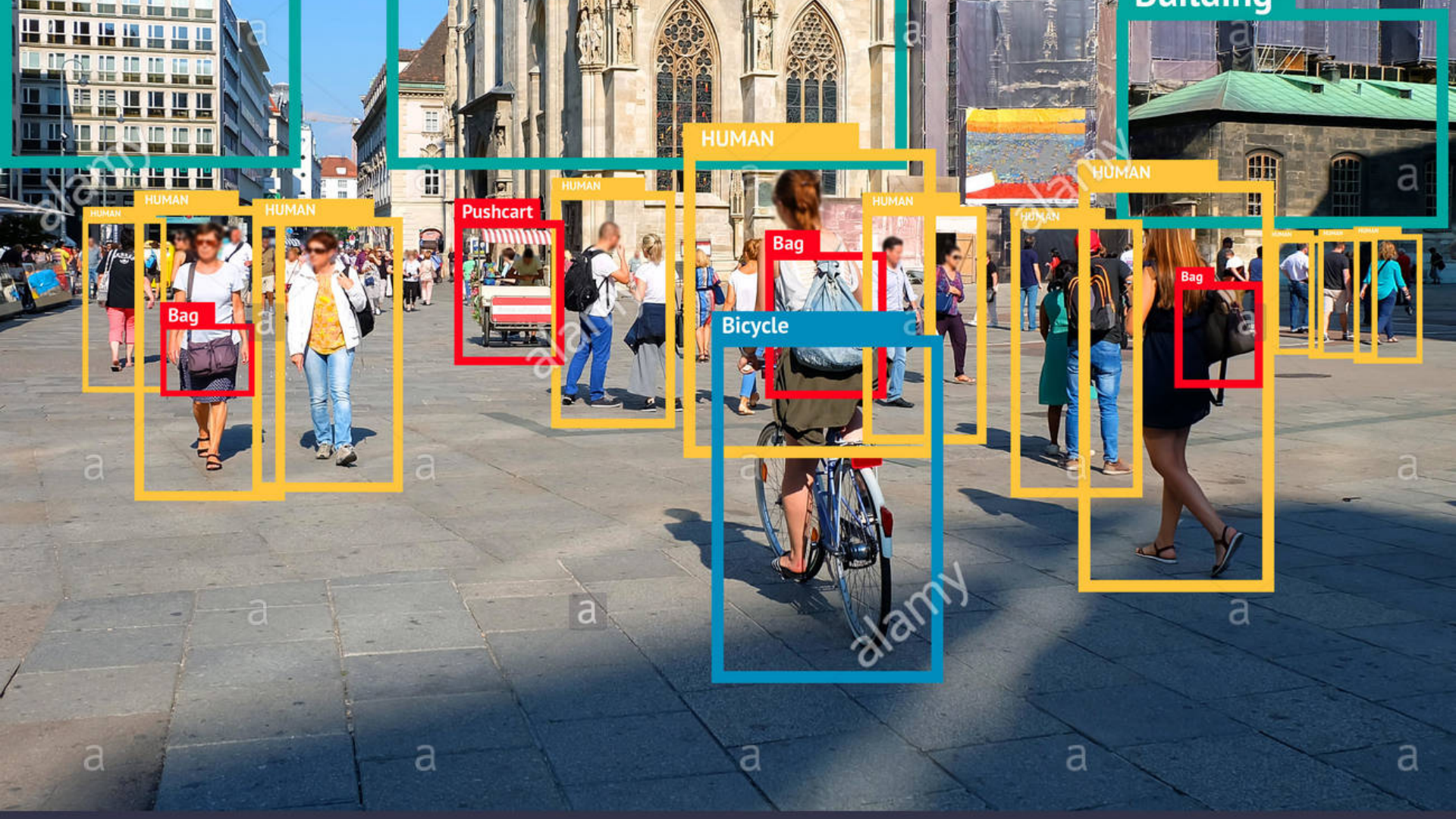
RSAConference2018
San Francisco | April 16 – 20 | Moscone Center

SESSION ID: HTA-F03

WINNING THE OS X MALWARE WAR: NEURALLY FINDING OUTBREAKS

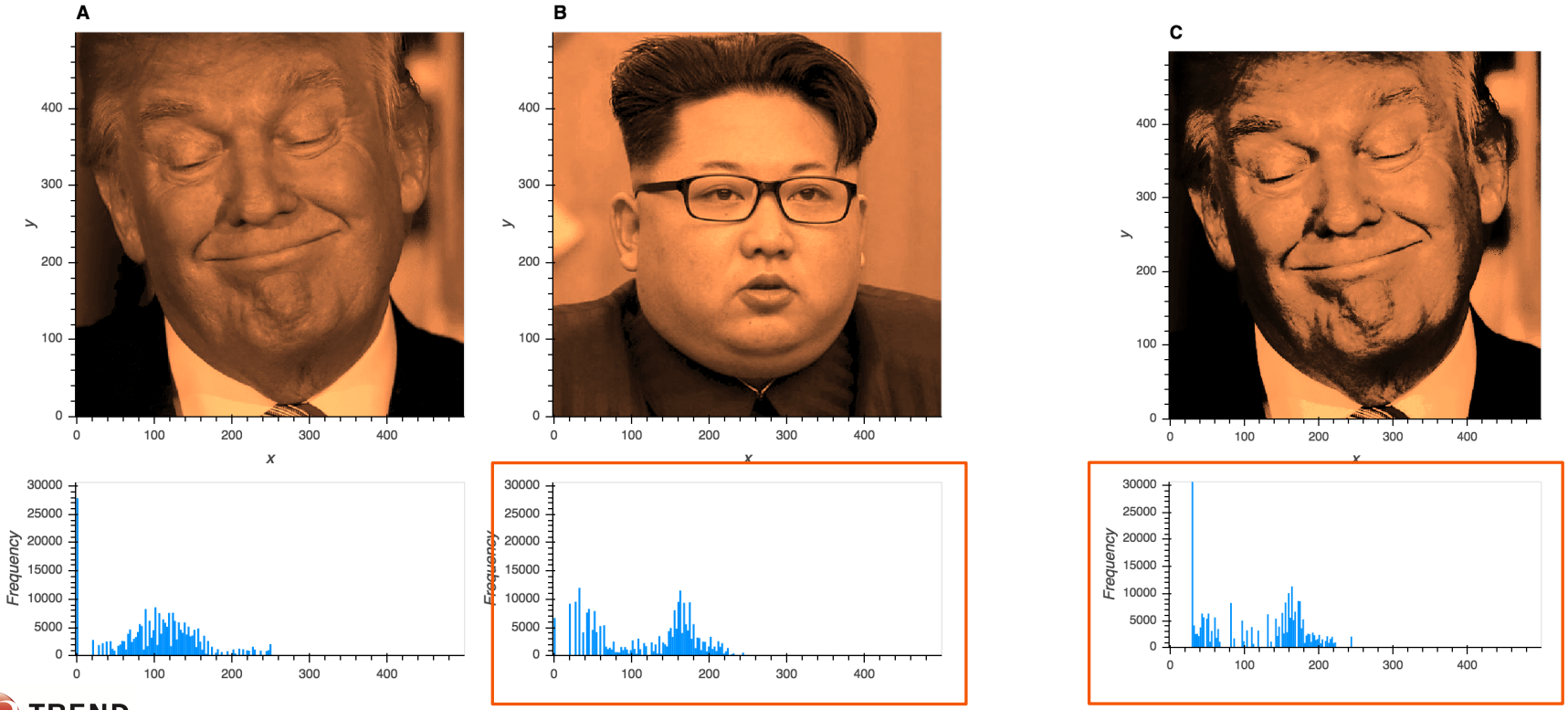
Sean Park

Senior Malware Scientist
Trend Micro


HUMAN
HUMAN
HUMAN
Bag
Pushcart
HUMAN
HUMAN
Bicycle
HUMAN
HUMAN
Bag

What we have tried…
and failed…

Features: Statistics



Features: Executable header


```
0000000000001000        struct __macho_header64 {
                            0xfeedfacf,                     // mach magic number identifier
                            0x1000007,                      // cpu specifier
                            0x80000003,                     // machine specifier
                            MH_EXECUTE,                     // type of file
                            5,                              // number of load commands
                            632,                            // the size of all the load commands
                            MH_NOUNDEFS|MH_DYLDLINK,        // flags
                            0x0                             // reserved
                        }
                            ;
                            ; Load Command 0
                            ;
0000000000001020        struct __macho_segment_command_64 {
                            LC_SEGMENT_64,                  // LC_SEGMENT_64
                            0x48,                           // includes sizeof section_64 structs
                            "__PAGEZERO", 0, 0, 0, 0, 0, 0, // segment name
                            0x0,                            // memory address of this segment
                            0x1000,                         // memory size of this segment
                            0x0,                            // file offset of this segment
                            0x0,                            // amount to map from the file
                            0x0,                            // maximum VM protection
                            0x0,                            // initial VM protection
                            0x0,                            // number of sections in segment
                            0                               // flags
                        }
```

Features: Executable header

Evading Machine Learning Malware Detection

Hyrum
Enc
hyrum@
by Filar
ame, Inc.
dgame.com

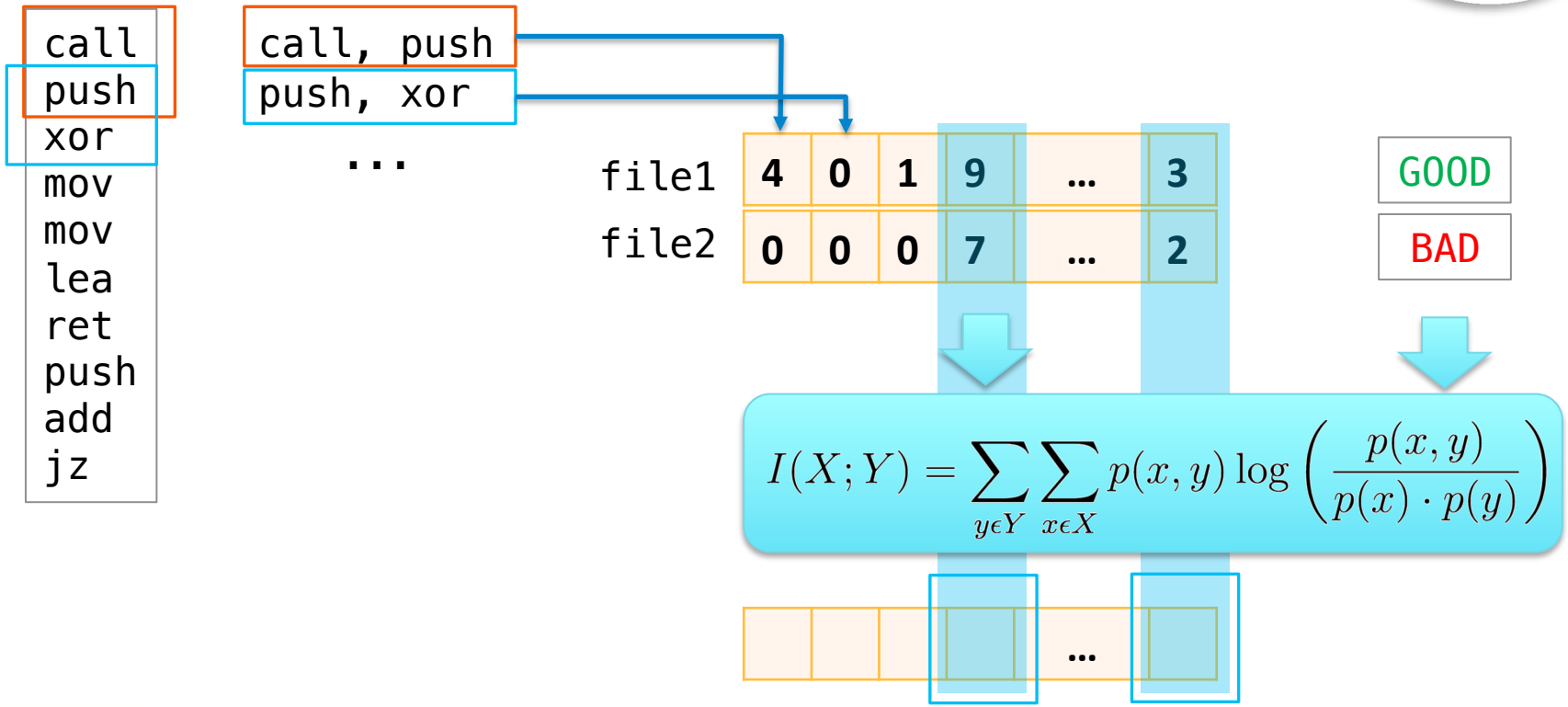
ABSTRACT

Machine learning is a
ware detection becau
seen malware familie
resulted in its practic
gines or supplementar
vendors. Recent worl
shown that models a
other attacks. In this
attacks that have bee
els in information secu
to have some degree o
tack. Importantly, ev
an attractive tool for anti-
mary detection engines or
ristics. Properly regularized
alize to new samples whose
ne distribution as the train-
rvised learning models auto-
elationships among features
discriminating between ma-
allows defenders to quickly
is manifest in the wild.
d sophisticated adversaries
le anti-malware engines, be

- add a … is never used … lored in [11]
- mani…
- creat…
- append bytes to extra space at the end of sections
- create a new entry point which immediately jumps to the original entry point
- manipulate (break) signature
- manipulate debug info
- pack or unpack the file
- modify (break) header checksum
- append bytes to the overlay (end of PE file)

Random mutations	13%
Black box attack	16%
Score-based attack	14%

Features: N-gram

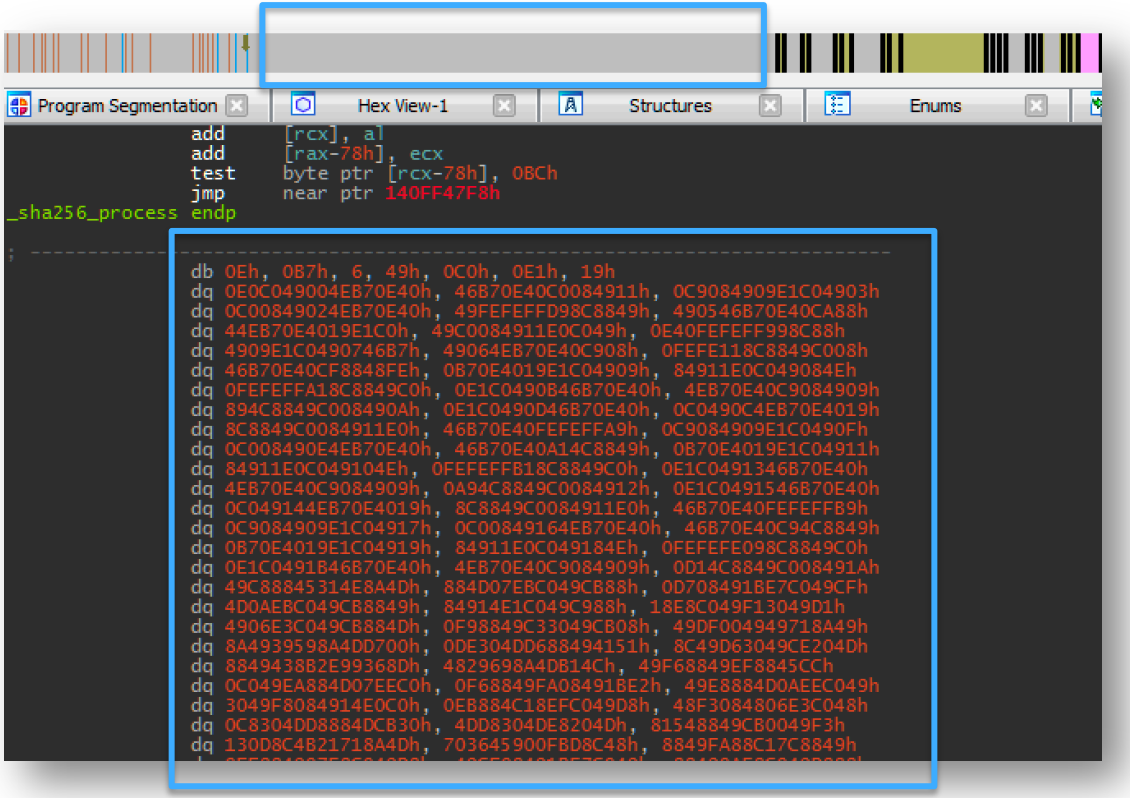


Features: N-gram

```
call
push
xor
mov
ret
push
call
push
```

```
call
push
mov
mov
call
push
mov
sub
pop
jz
add
mov
ret
```

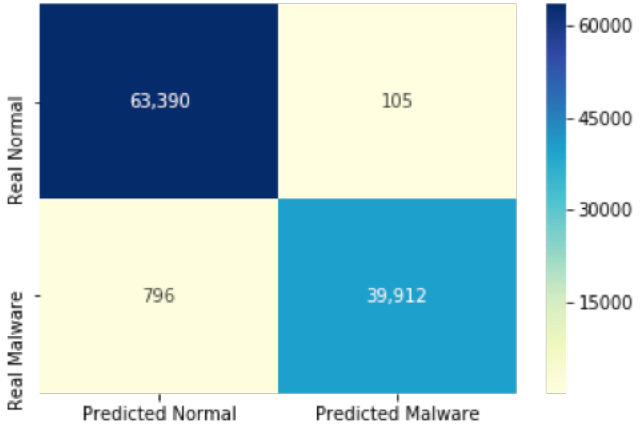
Features: Raw bytes



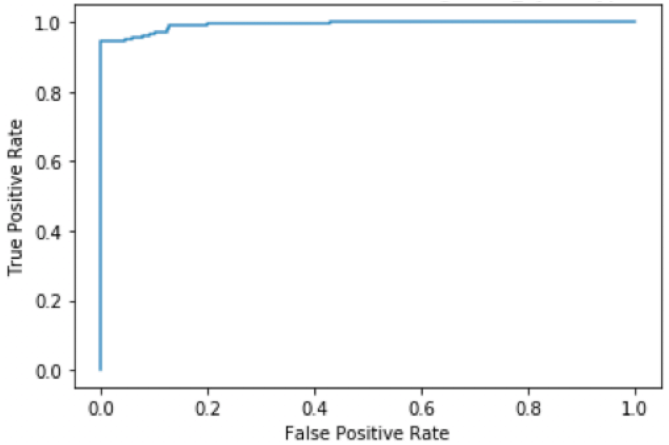
Outbreaks

What do we care about in malware detection?

Confusion Matrix

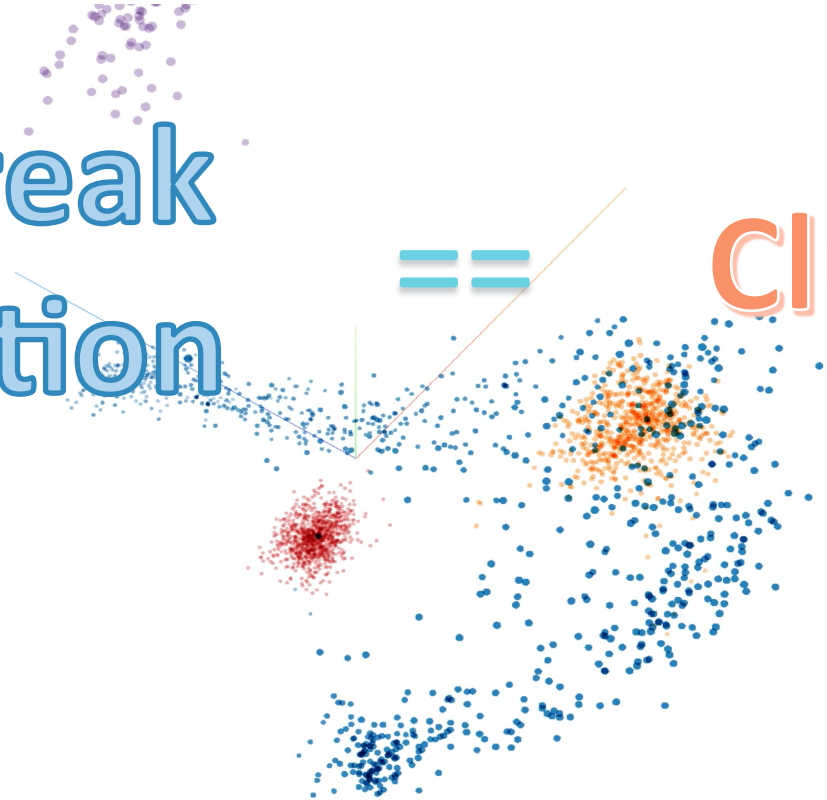


ROC Curve



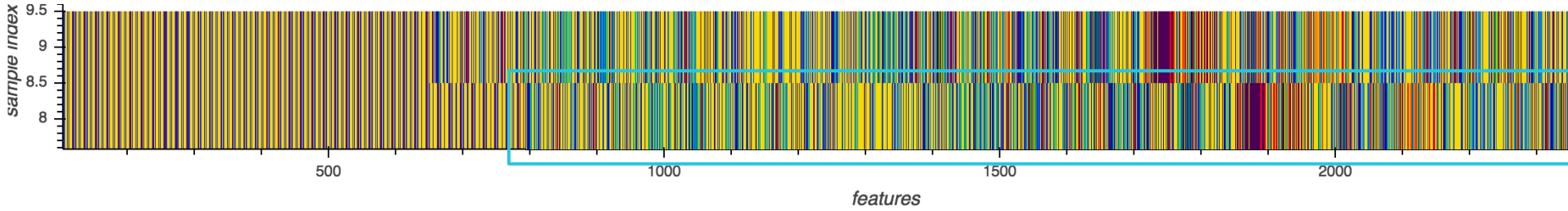
So, how do you detect outbreaks?

Outbreak Detection == Clustering

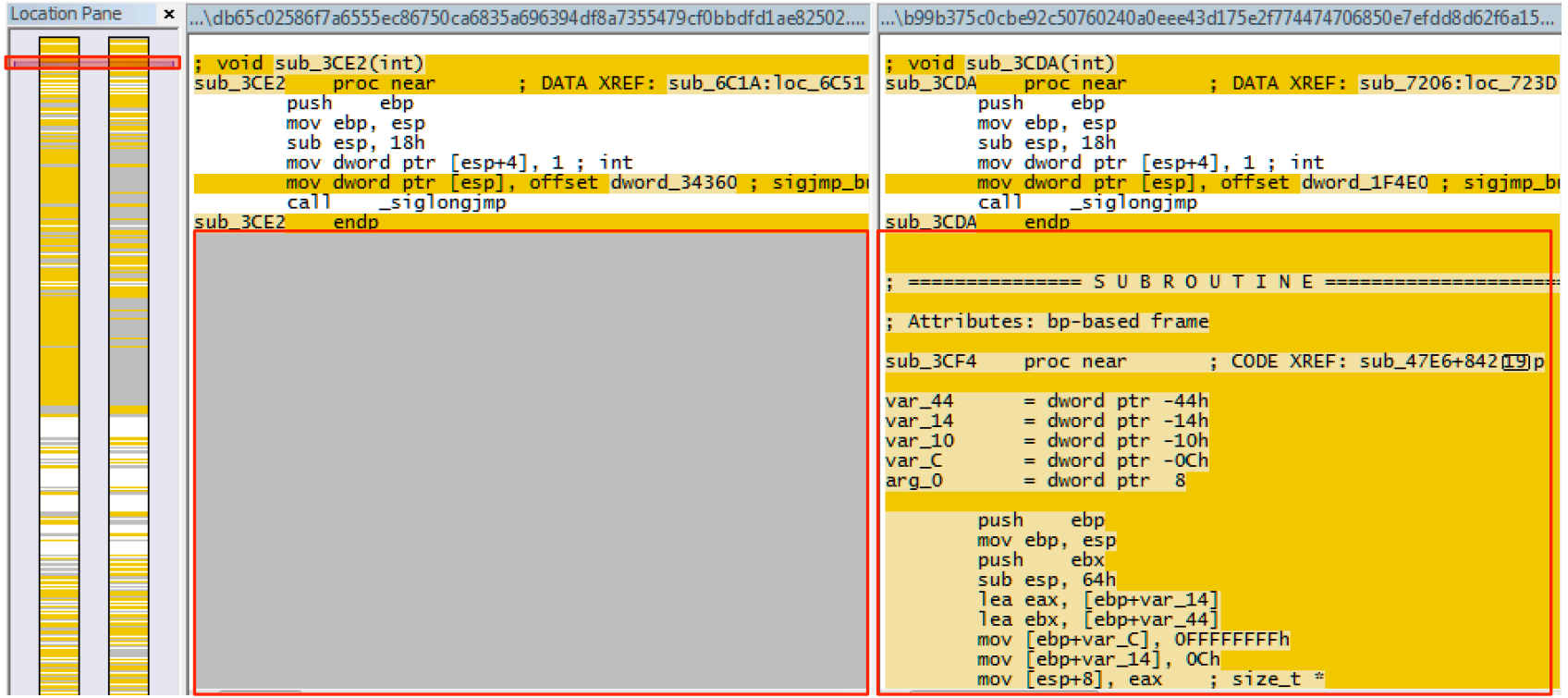


What does an outbreak look like? : Functional Upgrade

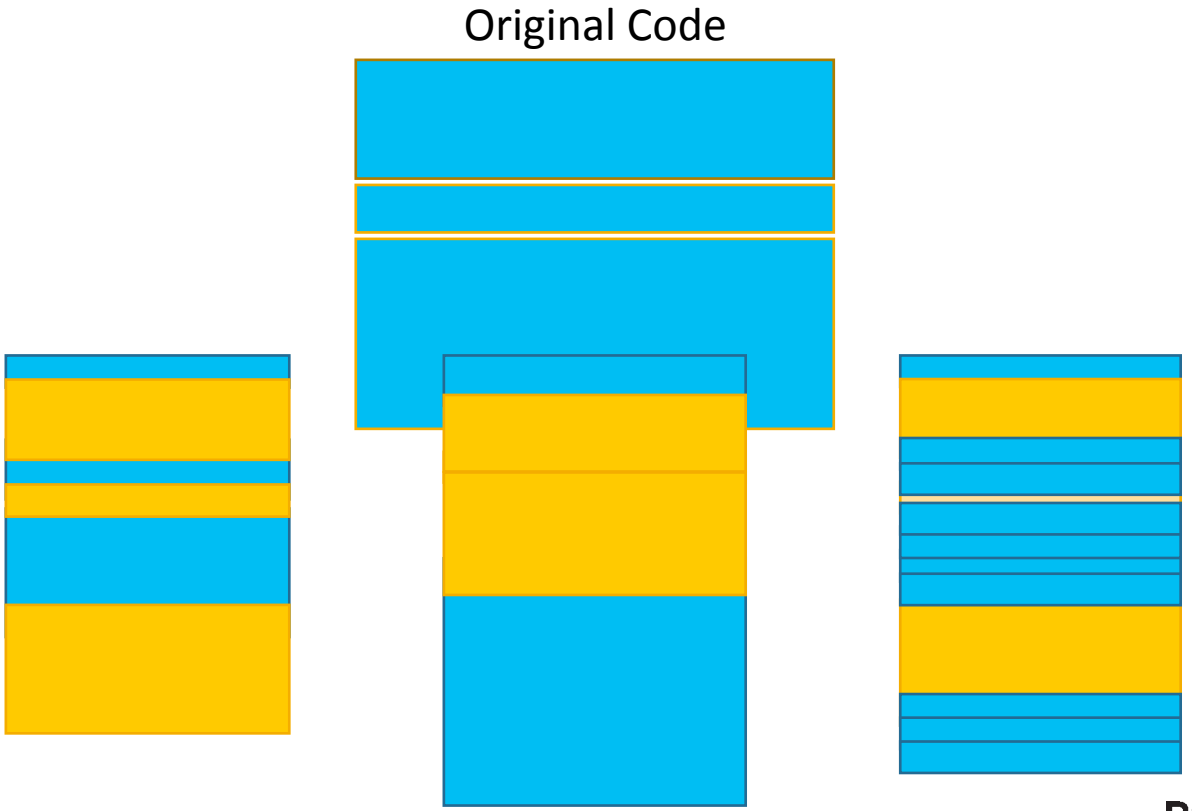
	name	id	file	filesize	offset	va	nfuncs	totalfuncsize
312	MAC.OSX.Trojan.FlashBack.AG	db65c02586f7a6555ec86750ca6835a696394df8a73554...	samples/2017-03_base/malware/db65c02586f7a6555...	220784	0	7152	366	41757
284	MAC.OSX.Trojan.FlashBack.F	b99b375c0cbe92c50760240a0eee43d175e2f774474706...	samples/2017-03_base/malware/b99b375c0cbe92c50...	132860	0	6448	280	16402



What does an outbreak look like? : Functional Upgrade



What does an outbreak look like? : Metamorphism



What does an outbreak look like? : Metamorphism

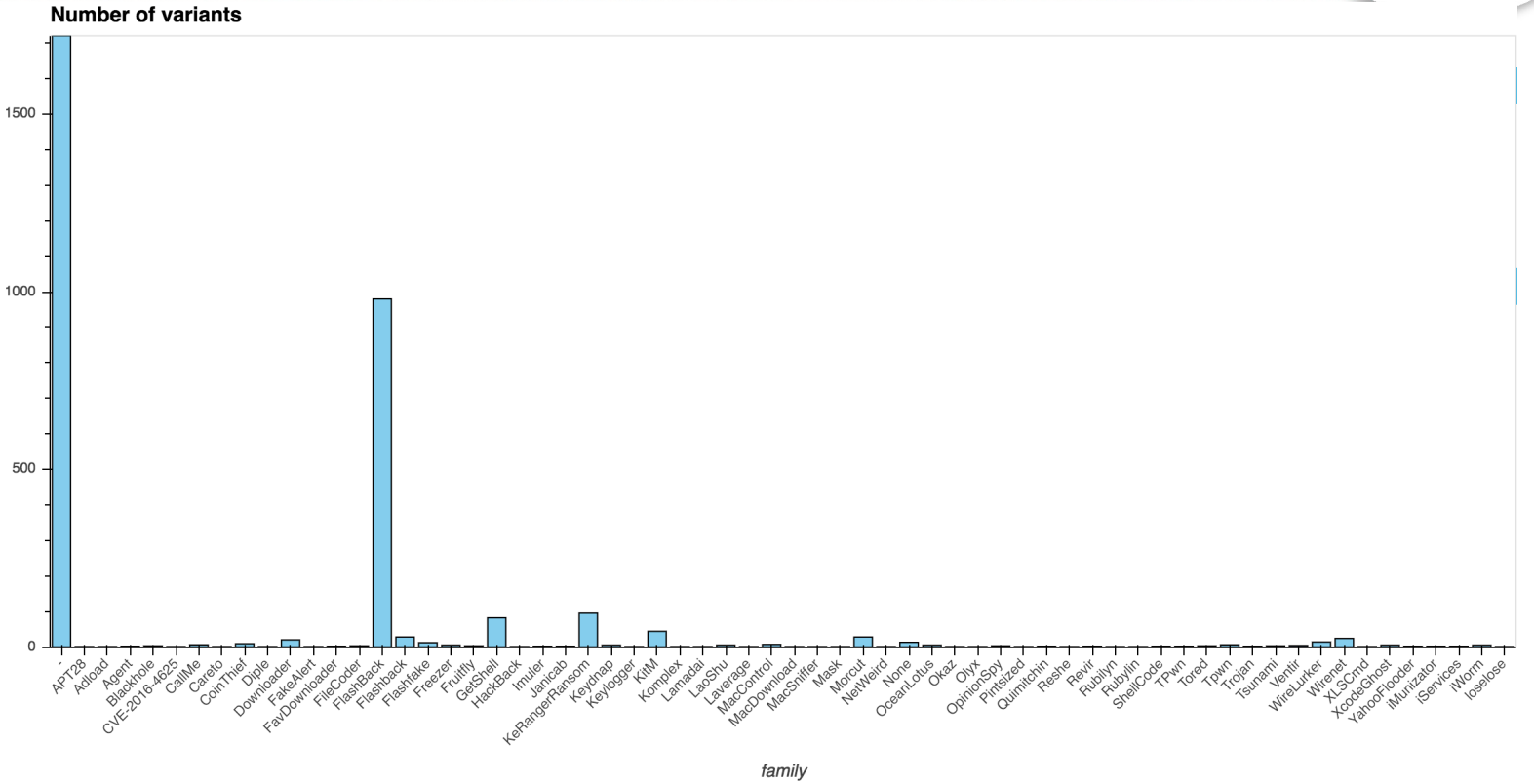

```
mov     ecx, [ebp-38h]
add     ecx, 1E6h
mov     edx, [ebp-2Ch]
sub     edx, ecx
mov     [ebp-2Ch], edx
mov     eax, [ebp-18h]
sub     eax, 2CEh
test    eax, eax
jz      short loc_41DFAB
mov     ecx, [ebp-38h]
add     ecx, [ebp-38h]
mov     edx, [ebp-18h]
sub     edx, ecx
mov     [ebp-18h], edx
mov     eax, [ebp-18h]
add     eax, 16Ah
mov     ecx, [ebp-18h]
sub     ecx, eax
mov     [ebp-18h], ecx
mov     edx, [ebp-38h]
mov     eax, [ebp-18h]
lea     ecx, [eax+edx+288h]
mov     [ebp-18h], ecx
mov     edx, [ebp-2Ch]
sub     edx, [ebp-18h]
test    edx, edx
jz      short loc_41DFE3
mov     eax, [ebp-2Ch]
sub     eax, [ebp-2Ch]
mov     ecx, [ebp-18h]
sub     ecx, eax
mov     [ebp-18h], ecx
mov     edx, [ebp-18h]
sub     edx, 0BEh
```

```
mov     ecx, [ebp-38h]
add     ecx, 1E6h
mov     edx, [ebp-2Ch]
sub     edx, ecx
mov     [ebp-2Ch], edx
mov     eax, [ebp-18h]
sub     eax, 2CEh
test    eax, eax
jz      short loc_41DFAB
mov     ecx, [ebp-38h]
add     ecx, [ebp-38h]
mov     edx, [ebp-18h]
sub     edx, ecx
mov     [ebp-18h], edx
mov     eax, [ebp-18h]
add     eax, 16Ah
mov     ecx, [ebp-18h]
sub     ecx, eax
mov     [ebp-18h], ecx
mov     edx, [ebp-38h]
mov     eax, [ebp-18h]
lea     ecx, [eax+edx+288h]
mov     [ebp-18h], ecx
mov     edx, [ebp-2Ch]
sub     edx, [ebp-18h]
test    edx, edx
jz      short loc_41DFE3
mov     eax, [ebp-2Ch]
sub     eax, [ebp-2Ch]
mov     ecx, [ebp-18h]
sub     ecx, eax
mov     [ebp-18h], ecx
mov     edx, [ebp-18h]
sub     edx, 0BEh
```

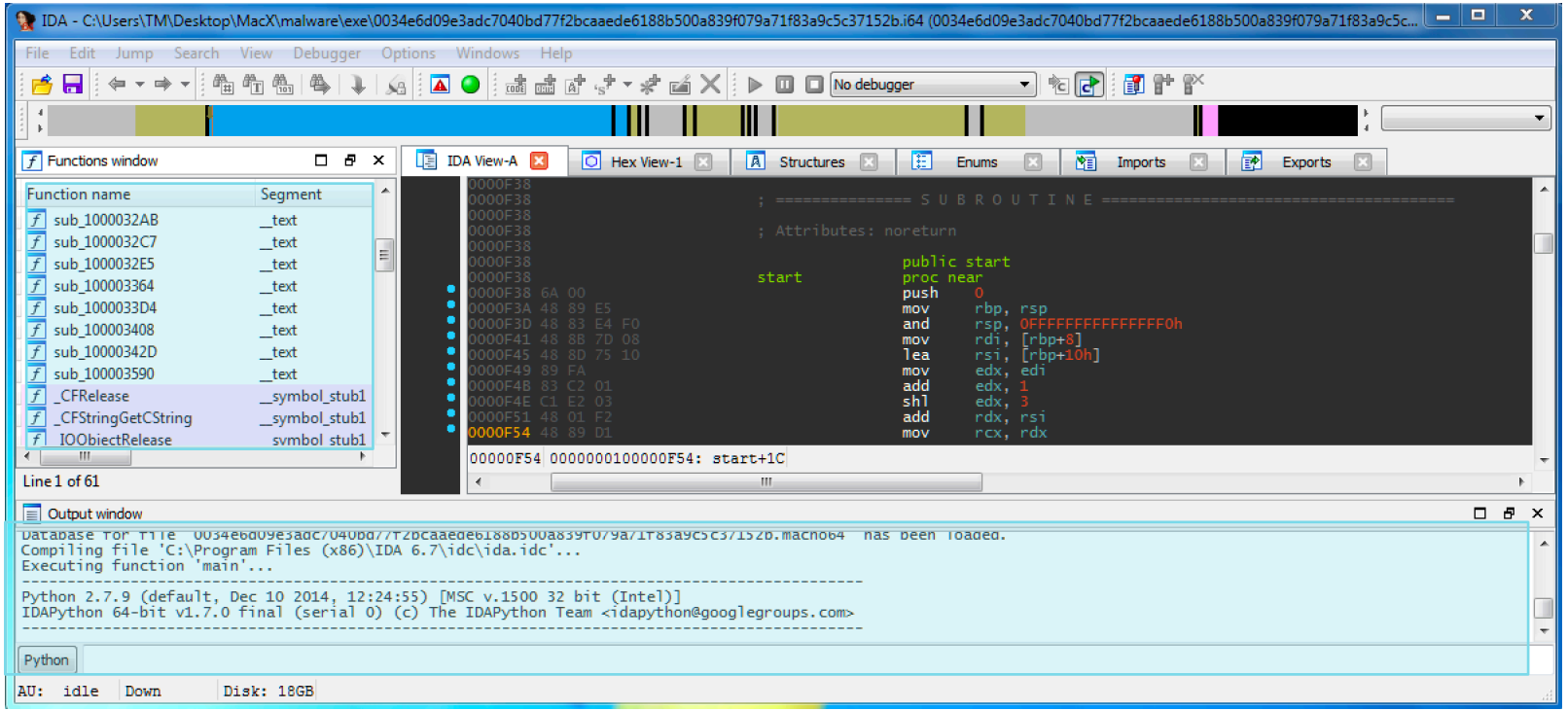
```
movzx   ecx, [ebp+var_18]
xor     ecx, 16h
mov     [ebp+var_2340], ecx
fld     ds:dbl_41CB70
fstp    [ebp+var_2318]
fld     ds:dbl_41CB68
fstp    [ebp+var_2310]
fld     ds:dbl_41CB60
fstp    [ebp+var_2308]
fld     ds:dbl_41CB58
fstp    [ebp+var_2300]
fld     ds:dbl_41CB50
fstp    [ebp+var_22F8]
fld     ds:dbl_41CB48
fstp    [ebp+var_22F0]
fld     ds:dbl_41CB40
fstp    [ebp+var_22E8]
fld     ds:dbl_41CB38
fstp    [ebp+var_22E0]
fld     ds:dbl_41CB30
fstp    [ebp+var_22D8]
fld     ds:dbl_41CB28
fstp    [ebp+var_22D0]
fld     ds:dbl_41CB20
fstp    [ebp+var_22C8]
fld     ds:dbl_41CB18
fstp    [ebp+var_22C0]
fld     ds:dbl_41CB10
fstp    [ebp+var_22B8]
```

```
push    ecx             ; cho
call    _strcat
add     esp, 8
mov     [ebp+var_36C4], eax
fld     ds:dbl_41FB30
fstp    [ebp+var_36B0]
fld     ds:dbl_41FB28
fstp    [ebp+var_36A8]
fld     ds:dbl_41FB20
fstp    [ebp+var_36A0]
fld     ds:dbl_41FB18
fstp    [ebp+var_3698]
fld     ds:dbl_41FB28
fstp    [ebp+var_3690]
fld     ds:dbl_41FB10
fstp    [ebp+var_3688]
fld     ds:dbl_41FB18
fstp    [ebp+var_3680]
fld     ds:dbl_41FB08
fstp    [ebp+var_3678]
fld     ds:dbl_41FB00
fstp    [ebp+var_3670]
fld     ds:dbl_41FAF8
fstp    [ebp+var_3668]
fld     ds:dbl_41FAF0
fstp    [ebp+var_3660]
fld     ds:dbl_41FB18
fstp    [ebp+var_3658]
fld     ds:dbl_41FB28
```

OS X Malware Dataset



OS X Features



OS X Features


```
def walk(self):
  text = ''

  prev_end = MinEA() + 1 # Account for prev_end-1 below.
  for i, fea in enumerate(Functions(MinEA(), MaxEA())):
    start = GetFunctionAttr(fea, FUNCATTR_START)
    end = GetFunctionAttr(fea, FUNCATTR_END)
    name = GetFunctionName(fea)

    lib = self.islib(fea)

    if not lib:
      self.add_function(name, start, end)

    prev_end = end
```

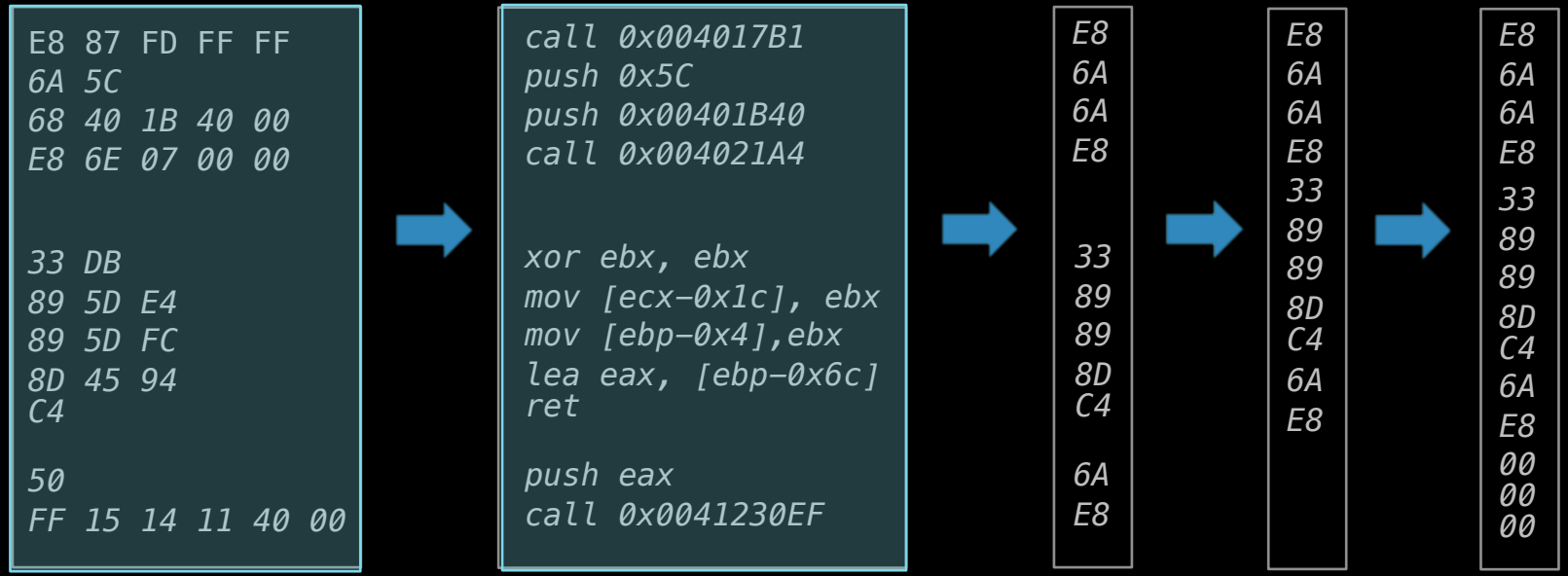
functions.json

```
{
    "va": 4294967295,
    "functions": [
        [
            4294971192,
            4294971251
        ],
        [ … ] // 2 items
    ],
    "filesize": 27245,
    "file": "0034e6d09e3adc7040bd77f2bcaaede6188b500a839f079a71f83a9c5c37152b.
}
```

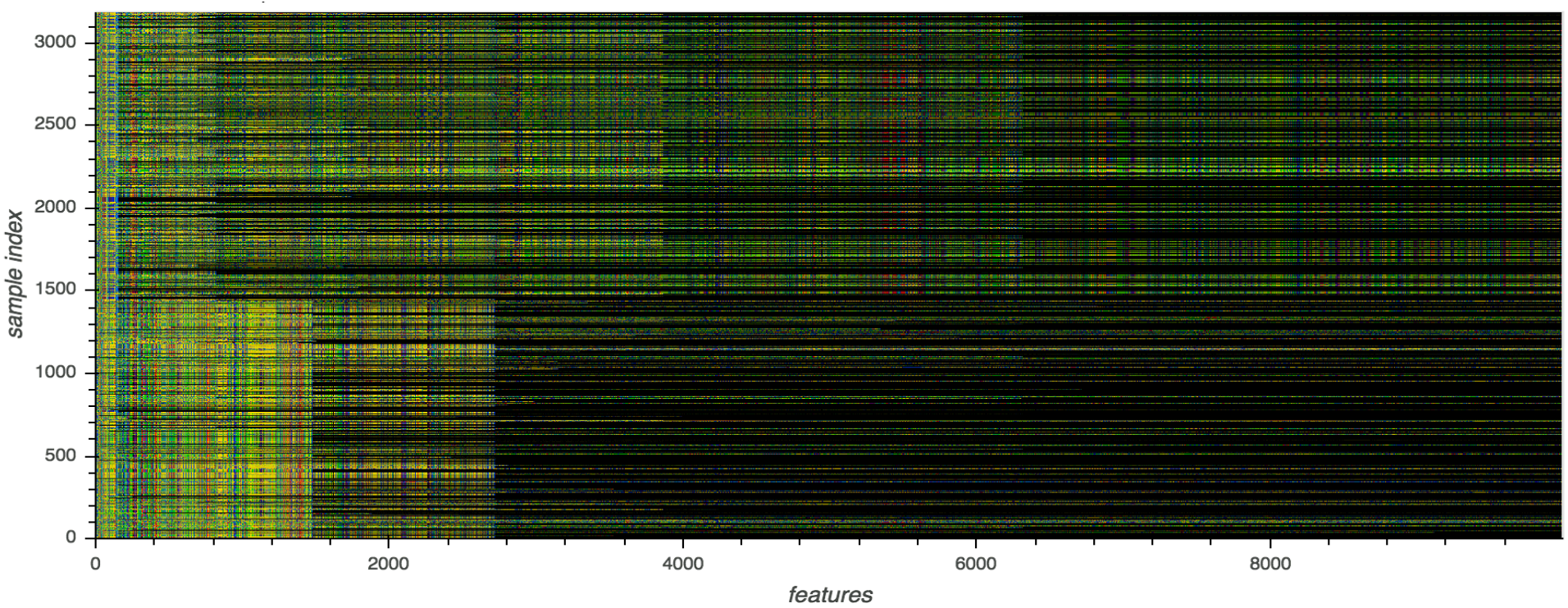
```
from capstone import *
from capstone.x86 import *

# Process IDA genereated JSON
func_features = []
filepath = os.path.join(dirpath, filename)
with open(filepath) as f:
    js = json.load(f)

    if js['arch'] == 'x86':
        md = Cs(CS_ARCH_X86, CS_MODE_32)
    elif js['arch'] == 'x64':
        md = Cs(CS_ARCH_X86, CS_MODE_64)

    for func in js['functions']:
        code = binascii.unhexlify(func['code']) # convert hex string to byte string.
        function = [ins.id for ins in md.disasm(code, len(code))]
        func_features.append(np.array(function, dtype=np.uint8))
```

program.npz

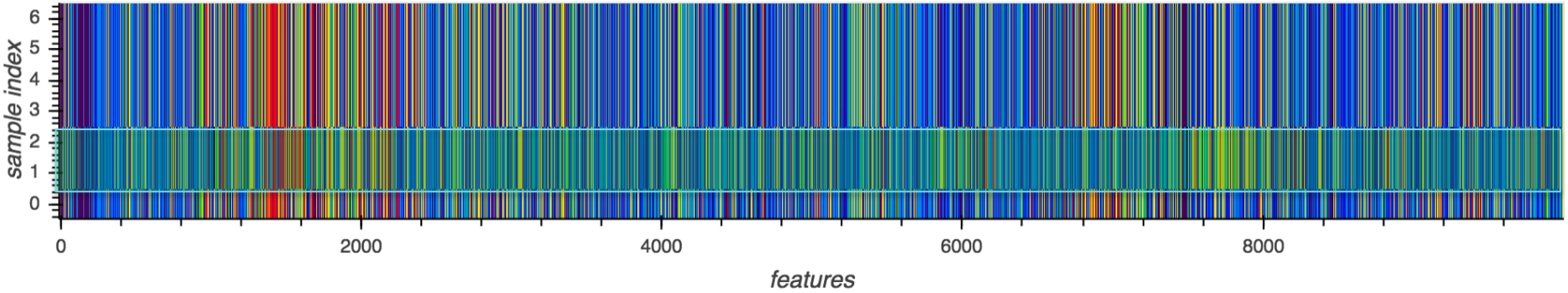
Feature Construction



Features: Instructions



Locality Sensitive Hash (LSH)



7E227A6559E738025EDBF3E3964162076BA0DB124803F761798C5F6747A88B8F83C2DE

7E227A6559E738025EDBF3E3964162076BA0DB124803F761798C5F6747A88B8F83C2DE

Distance: 0

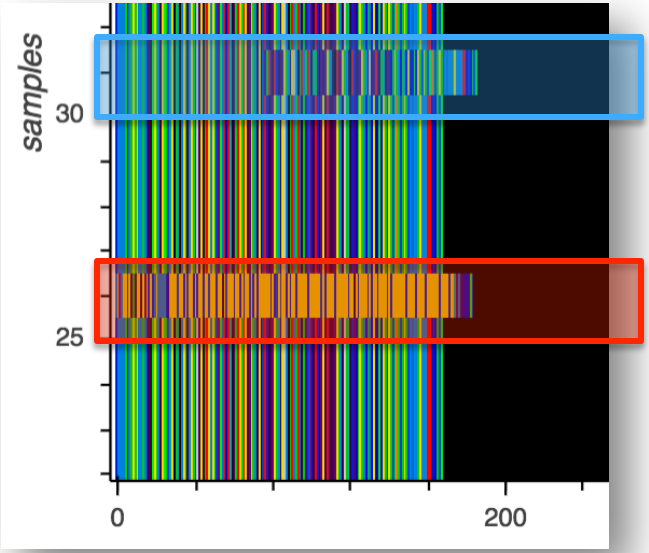
7E227A6559E738025EDBF3E3964162076BA0DB124803F761798C5F6747A88B8F83C2DE

D322567548E778866E97F3E35A82124BAB60EB120447F7417E8C5F524BAD874E02C2ED

Distance: **92**

HDBSCAN

Class 38 (nsamples=669)
Mac.OSX.iWorm.F (1 samples) [471]
Trojan.MAC.KeRangerRansom.A (**53** samples)
Trojan.MAC.KeRangerRansom.D (**28** samples)
Trojan.MAC.Tsunami.A (1 samples)
Unknown (586 samples)

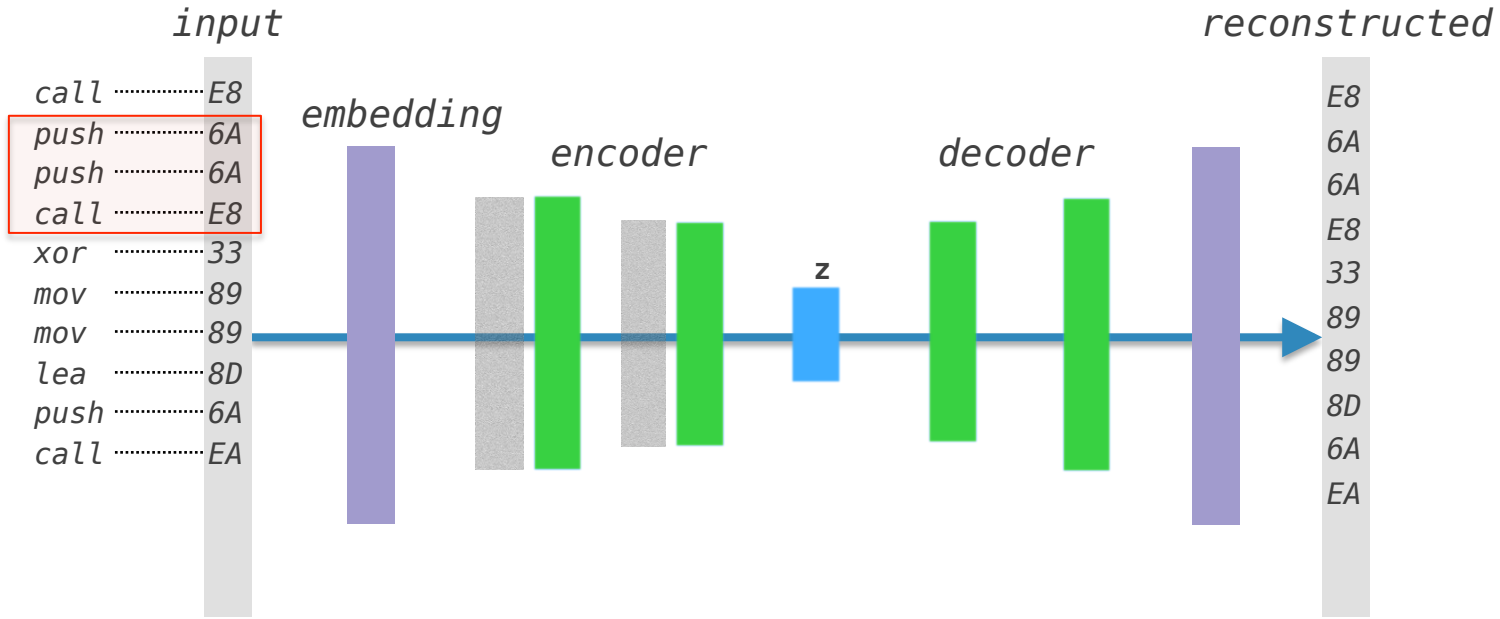


Sample 31 #598
train Trojan.MAC.Tsunami.A(label=180)
id=a3620953462331 0b50d9460314f147e21a92c0e462f74faf046718ba8fb18cbb
filesize=24576
va=0xffffffff
nfuncs=4
totalfuncsize=187
scan_date=2017-06-07-08:11:09

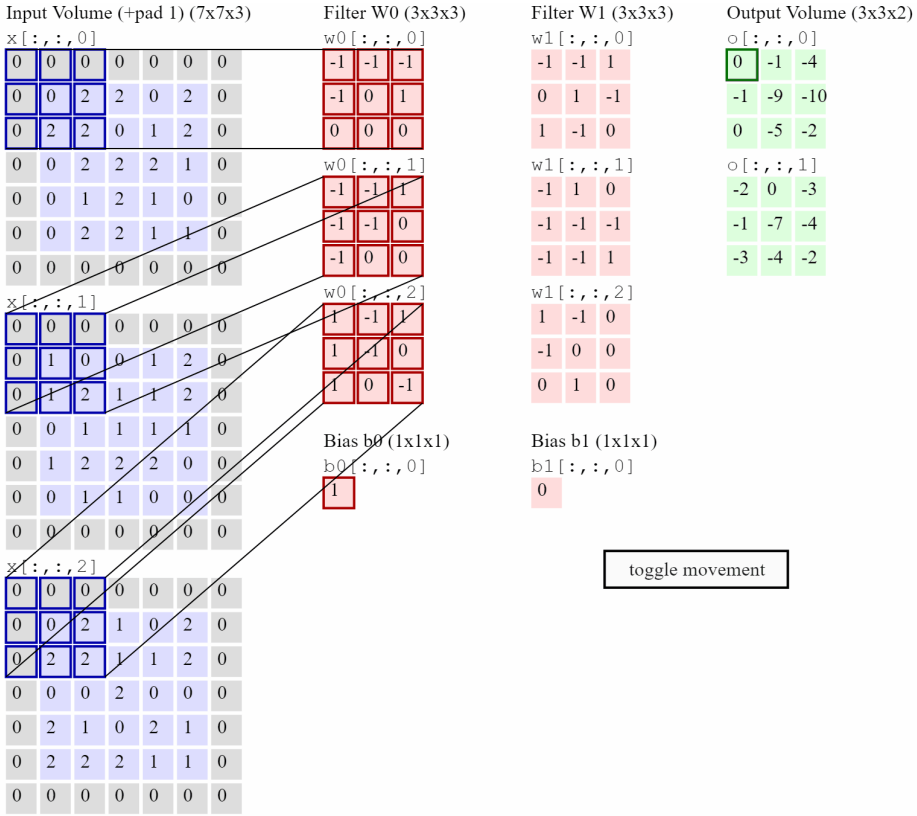
Sample 26 #471
train Mac.OSX.iWorm.F(label=142)
id=6ed282e6e54ccbec80b16bb39fbdf465576215a94ff1b7875fc8b28d29392fb6
filesize=177044
va=0x18f0
nfuncs=2
totalfuncsize=185
scan_date=2017-07-26-11:35:32

Neural Outbreak Detection

Stacked De-noising Autoencoder (SDA)

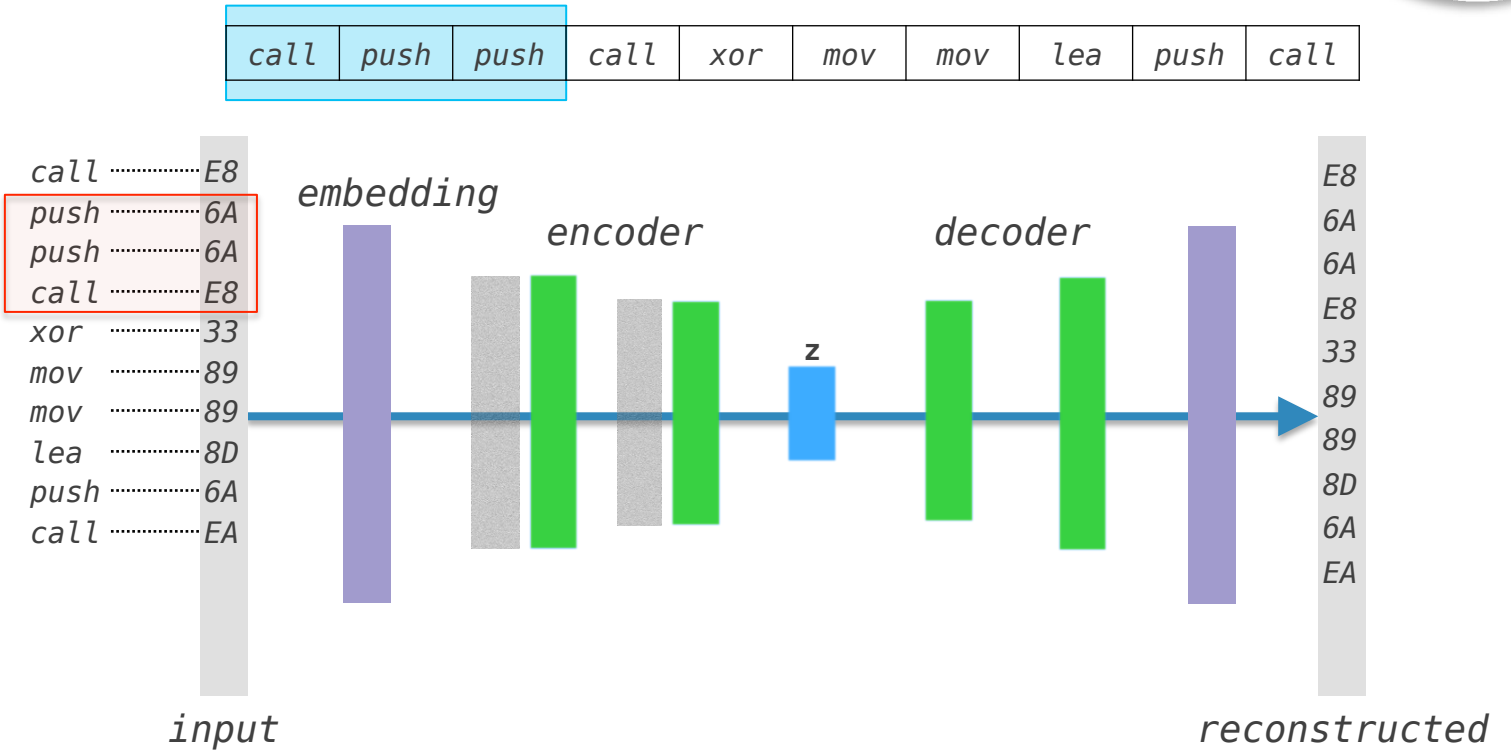


Convolutional Neural Network (CNN)

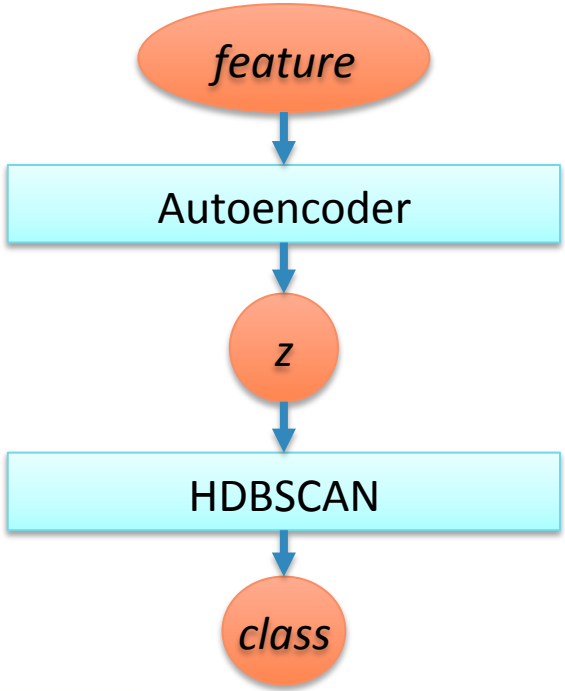


http://cs231n.github.io/convolutional-networks/

Sparse Convolutional Autoencoder



Putting them together




```
call, push, push, xor, mov, mov, lea, ret, …
```

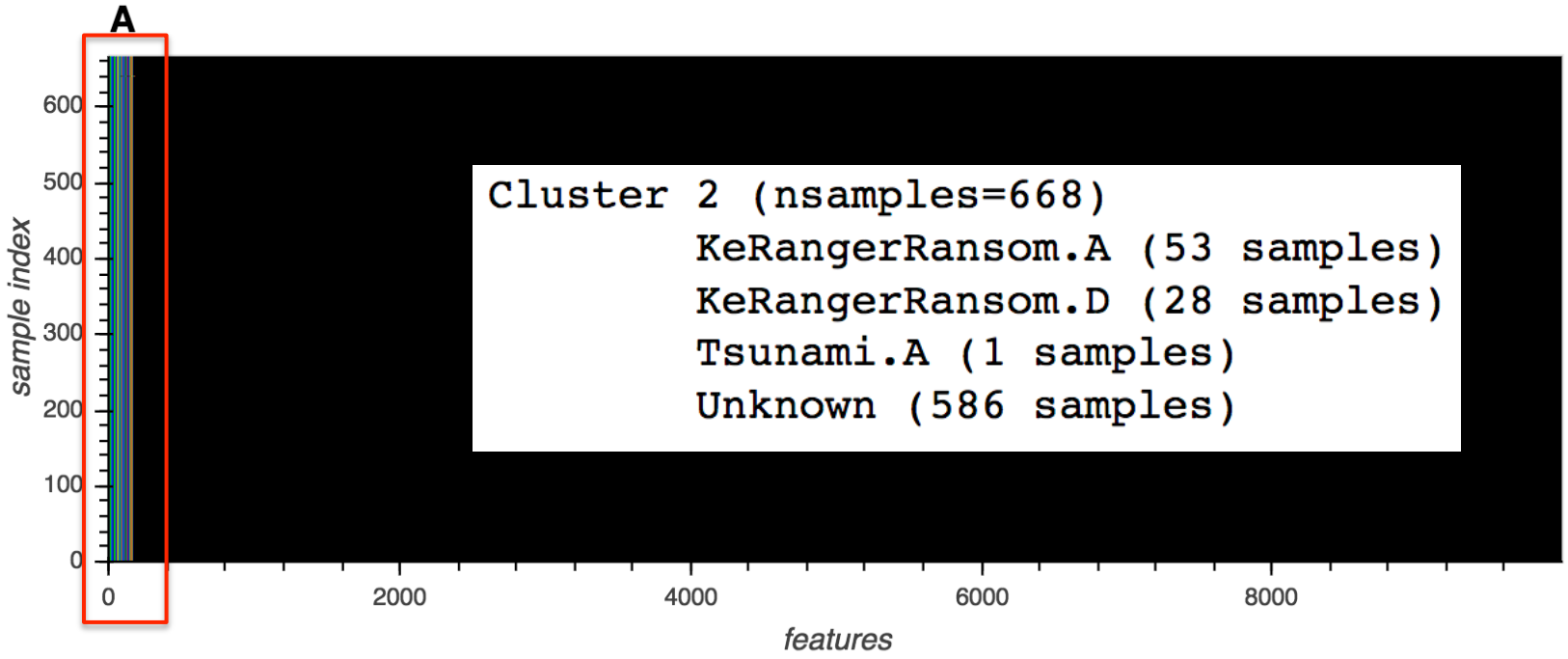
```
38, 99, 99, 186, 8, 8, 45, 127, …
```

```
1.3, 0.5, -0.91, 2.12, -0.01
```

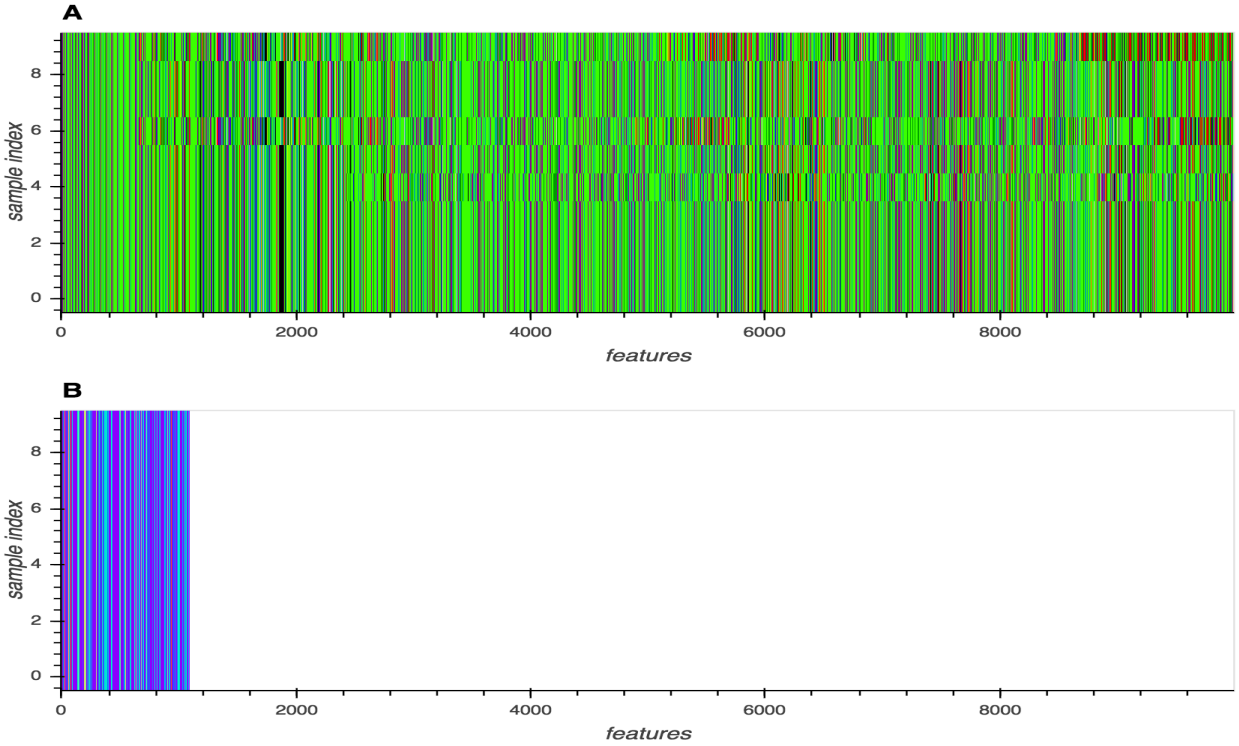
```
Class 7
```

Model Performance

Power of Instruction Features

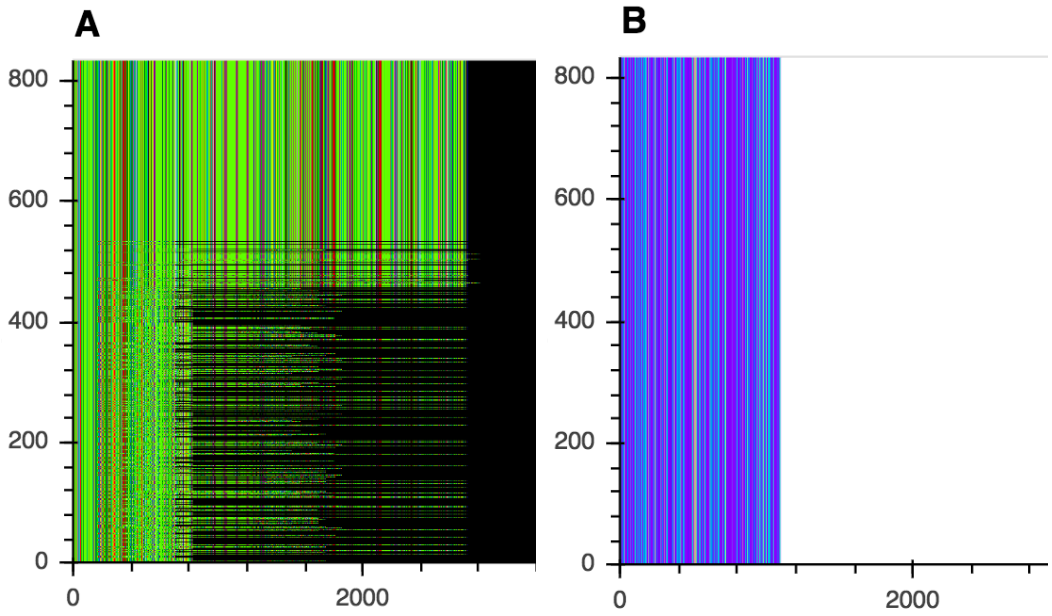


Power of Convolution




```
Cluster 11 (nsamples=10)
        FlashBack.AE (1 samples)
        FlashBack.AG (1 samples)
        FlashBack.AP (1 samples)
        FlashBack.E (1 samples)
        FlashBack.F (3 samples)
        FlashBack.P (2 samples)
        FlashBack.Q (1 samples)
```

Unknown Sample Detection

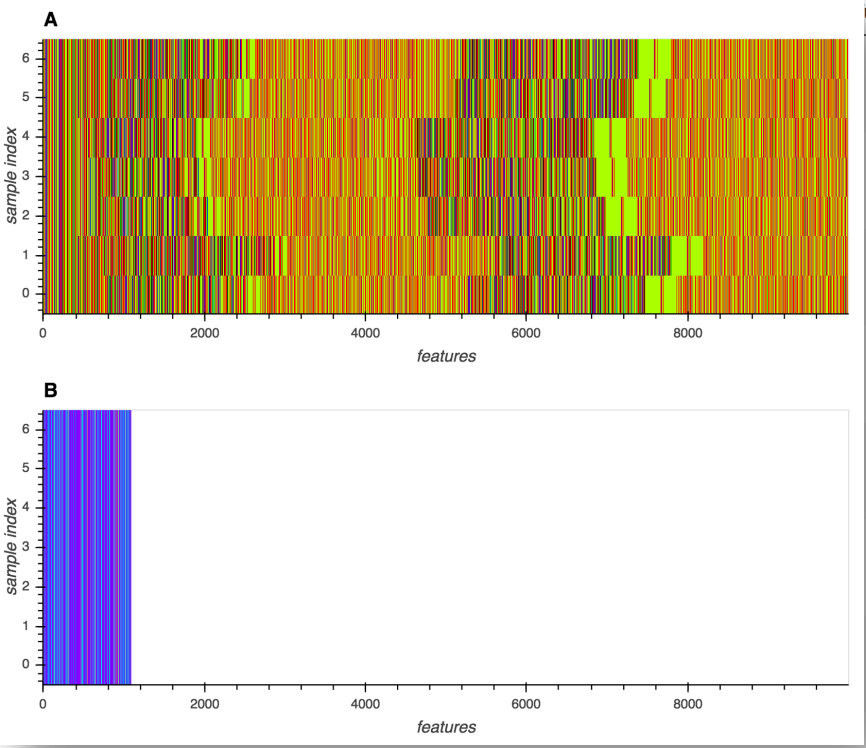



```
Cluster 49 (nsamples=836)
        FlashBack.AF (2 samples)
        FlashBack.L (344 samples)
        FlashBack.M (5 samples)
        FlashBack.Q (2 samples)
        Flashback.E (1 samples)
        Flashback.J (1 samples)
        Flashback.K (1 samples)
        Flashback.L (1 samples)
        Flashback.M (10 samples)
        Flashback.N (7 samples)
        Flashback.O (1 samples)
        Flashback.P (1 samples)
        Flashback.Q (1 samples)
        Trojan-Downloader.Flashfake.a
        Unknown (447 samples)
```

Early Detection

#RSAC

	name		uncsize	positives	scan_date	label	da
1344	Blackhole.D	b4088b79c61e484577c260a659	506703	31	2017-06-21-10:25:41	40	
30	Freezer.A	da2f382db0a46ed94a0b5eb9d540	496406	26	2017-06-07-08:11:37	109	
690	Freezer.A	87d12bdd1088c73165e03e12804	468589	29	2017-06-07-08:11:02	109	
866	Freezer.A	53cda694790f4fcb4cc1a6dfca9c	487178	28	2017-06-07-08:10:56	109	
553	Freezer.A	554f24058df192107c3057b4fb4	460489	29	2017-06-07-08:10:52	109	
1046	Blackhole.B	1c12d378c3f64567318121bb99	475478	28	2017-06-07-08:10:37	38	
355	Blackhole.C	12dea206341ca0563b8a17f6ef73	493339	31	2017-04-27-03:22:01	39	



Apply

- Use convolutional autoencoder and HDBSCAN to identify dynamically morphing malware variants across outbreaks.
- Grab malware samples in your repository, extract functions dataset using IDA Pro python script, train it with tensorflow or your favorite deep learning API, cluster it with HDBSCAN, visualize the clusters using holomap, and **name** all those ‘unknown’ samples.

Credits

Jon Oliver

Lili Diao

Will Zhuo

Steven Du

Gavin Gan

MacX team

MacTRT team

References

[1] Silver, David, et al. "Mastering the game of go without human knowledge." Nature 550.7676 (2017): 354.
[2] Anderson, Hyrum S., et al. "Evading machine learning malware detection." Black Hat (2017).
[3] Santos, Igor, et al. "Opcode sequences as representation of executables for data-mining-based unknown malware detection." Information Sciences 231 (2013): 64-82.
[4] Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." Proceedings of the 26th annual international conference on machine learning. ACM, 2009.
[5] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.
[6] Zhang, Yizhe, et al. "Deconvolutional paragraph representation learning." Advances in Neural Information Processing Systems. 2017.
[7] L. McInnes, J. Healy, S. Astels, hdbscan: Hierarchical density based clustering In: Journal of Open Source Software, The Open Journal, volume 2, number 11. 2017
[8] https://en.wikipedia.org/wiki/Locality-sensitive_hashing
[9] https://www.youtube.com/watch?v=Ilg3gGewQ5U
[10] http://cs231n.github.io/convolutional-networks/
[11] https://deeplearning4j.org/convolutionalnetwork
[12] https://docs.gimp.org/en/plug-in-edge.html

QUESTIONS?

Sean Park, Senior Malware Scientist
spark@trendmicro.com