February 2020

METHOD AND SYSTEM FOR DETECTING A PROCESS OR ACTIVITY USING RECURRENT AND CONVOLUTIONAL 1D NEURAL NETWORKS

Andrey Kvasyuk

Hazim Dahir

Omar Santos

METHOD AND SYSTEM FOR DETECTING A PROCESS OR ACTIVITY USING RECURRENT AND CONVOLUTIONAL 1D NEURAL NETWORKS

AUTHORS:

Andrey Kvasyuk
Hazim Dahir
Omar Santos

ABSTRACT

Presented herein are techniques that use multiple neural networks and segmentation of the traffic to detect the presence of applications or business processes within a noisy mixture of network traffic. In addition, the techniques presented herein provide a novel way to detect unusual, bad intentioned, and/or malicious activity, which is also a “process”, using recurrent and convolutional neural networks. The learning outcome can potentially identify compromised network infrastructure devices and/or telemetry collectors.

DETAILED DESCRIPTION

Data is being generated at high volumes, at high velocities, and with veracity by many different devices, applications, things, processes, etc. Conventional approaches to analysis of data typically involves collection of the data from the sources using different collectors and then storing the data into centralized repositories. In some cases, data may be tagged or labeled indicating the source (e.g., device or process of origin). However, more often data is not labeled (for various reasons); and subsequently, the relationship between the data and its source (process or thing) is not obvious. Similarly, sequences from multiple processes are lost in the mix of large repositories full of untagged data. In addition, cyber attackers are using techniques for attacking modern applications and "end-to-end processes."

As described further below, the techniques presented herein propose automated and smart methods for analyzing data within telemetry collectors to glean knowledge about the underlying "processes" and application interactions. By learning about the underlying process, it is possible to create a model the describes the "end-to-end process" behavior.

This information, in turn, allows the system to recognize bad intentioned and/or malicious activity from future data streams.

Applications, Devices, Things or Processes (collectively and generally referred to herein as “processes”) generate sequential data streams (e.g., network flows) in high volume. Data Collectors aggregate such sequences of data from multiple source processes into a single repository. In numerous cases, data streams are stored in a repository in a "blended" form or with a random mixture of multiple data sequences. In addition, in many cases, the elements of the data sequences are not tagged or labeled by the name of the source process generating the telemetry data stream. For example, processes can use the same symbols when generating telemetry data.

The proposed techniques automatically detect the presence of the known “processes” in a random mixture of "unlabeled" data sequences. For example, as shown below in Figure 1, three processes each generate specific sequence of symbols.

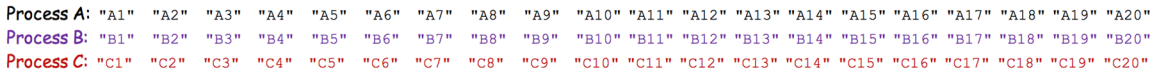

Figure 1

In addition, as shown below in Figure 2, the flow collector captures “network and/or data flows” and saves the flows into a repository as an interlaced mixture in much less clear form.

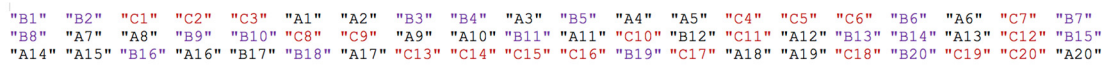

Figure 2

As shown, each symbol does not carry any information about the source process (i.e., the process from which the symbol was originated). Therefore, as shown below in Figure 3, the collector stores the data from Processes A, B, and C as an unlabeled sequential mixture of symbols.

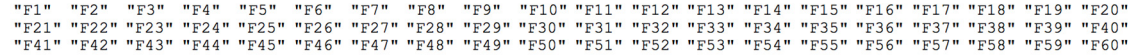

Figure 3

Figure 4, below, illustrates an application/business Process use case to detect te presence of the "Business Processes/Applications" in network flow data.

```
  [1] "SMBLTJUK" "WYFJAYWP" "AIFSWSDU" "QUNXXQTO" "OQLITHYG" "JLFBOQTW" "UFVOUOST" "EURHDQXH" "SXXPCARH" "CADDFVQH" "WEHEBMMH"
 [12] "EYXHFKRS" "HJIRGEDO" "DLCHNEWJ" "SOJFXXVS" "QZRUMZHG" "AGBPQTEW" "IHVLMLKZ" "FKJMBWJW" "UFYEEVMM" "MYANUNOA" "XVGVZBIX"
 [23] "LIWLPTEG" "GAJVWAMM" "MPAGOSDL" "MQIZKGRA" "OGDRWZEB" "IWTHIUKB" "QUPVNMVM" "QKOZPEIN" "VMANSMXG" "LJDJDYCN" "LKDRELTU"
 [34] "PQVNRMQW" "LRDJEQXS" "IFEMDNND" "SYZDUMJZ" "DJFNPZWA" "OMAXHJNV" "LFNBYICQ" "WEUXAPIH" "OOBUUGBY" "IBAUDXUU" "JWIBIHQW"
 [45] "HNWRUZDD" "FRIXTNLF" "ZEIDWOTL" "GAGJWVRX" "LNEOZWLH" "CNAYXTAA" "ABEDGOXU" "EKRWDIFF" "DVPDVVIG" "PJILFUZQ" "FZVLBCTT"
 [56] "SDRCTVLJ" "SMDMGMVG" "VCVZPSXR" "AYGGDWKY" "GSQWTOYG" "EGYPZTZU" "YPEGYEFU" "CSFKBMOP" "TLXYSSCO" "IZYMTRIK" "OJQCFVBO"
 [67] "HCFJSADM" "TRGKWKQV" "PCAMHHNK" "INOVJLAF" "JOTOAICZ" "QTFQTMBH" "NCOLHCNT" "NNLZUTIO" "XGXXFEET" "NTEOOIXG" "VSKKXWYE"
 [78] "NDOTOBPV" "YNFXEEQI" "OWKHSKIH" "BWJASEPS" "JURNZOJJ" "GYMWSFFD" "UHRWPXZC" "HKYNVXLP" "QGRPYLOU" "AQWKDPPO" "GLWVUPPM"
 [89] "AXKPJGMK" "AVUOOGQP" "WOOUIXWH" "QQLZSVAN" "AESJMTSZ" "QUBBNTJE" "LPVSLYLS" "UDOYGDYP" "SNREUKGK" "JRQGSTZI" "EAXQSBYU"
[100] "DQMUSTHU" "ABVNRQLD" "UIUAVSAO" "BTWWNIJA" "LWFTPLWY" "JHTYOUPN" "LXASMPCO" "ODOAZYAU" "RSNRGSBX" "HJMSWDXT" "HXHTHVMB"
...
```

Figure 4

In Figure 4, the symbol "SMBLTJUK" represents conversation "152.15.40.215|52.21.9.111|443" (src, dst, dst_port), and it can be part of the traffic from different applications or business process.

Once each of "Business Processes/Applications" are detected and categorized, unusual, bad intent, and malicious activity could be identified using recurrent and convolutional neural networks. In addition, a similar approach can be taken to identify any potential malicious activity against a network infrastructure and/or telemetry collection device (e.g., if the network infrastructure device or collector is compromised).

Figure 5, below, is an overview diagram illustrating aspects of the techniques presented herein.

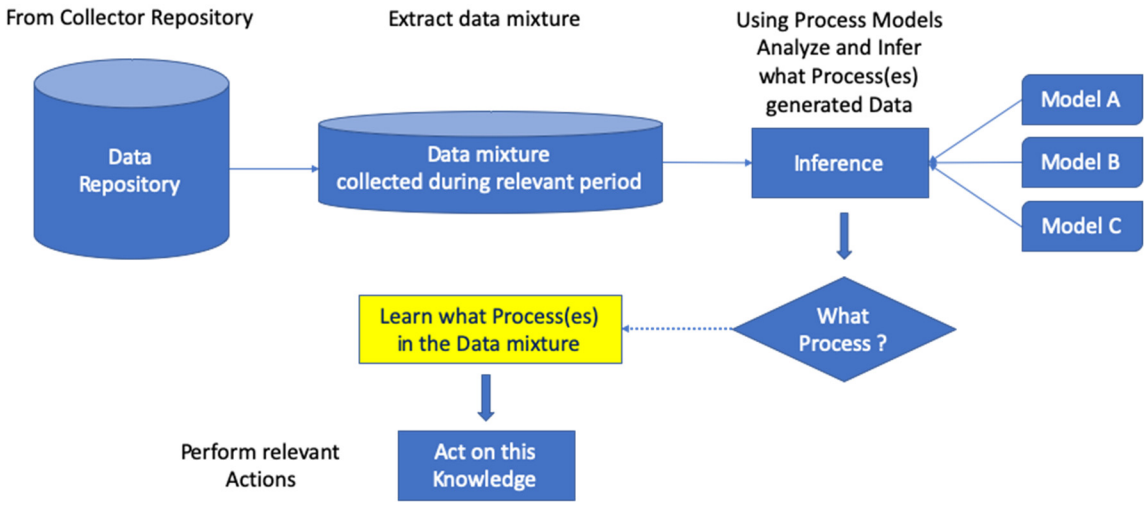

Figure 5

Aspects of the techniques presented herein will be described below with reference to four (4) general steps, namely: (1) Learning and Modeling Processes, (2) Generation of Training Data, (3) Building a Classification Model, and (4) Detection.

Step 1: Learning and Modeling Processes

As the name suggests, during the learning phase, multiple streams of data sequences are collected and aggregated by one or multiple independent collectors into a central place, as shown below in Figure 6.

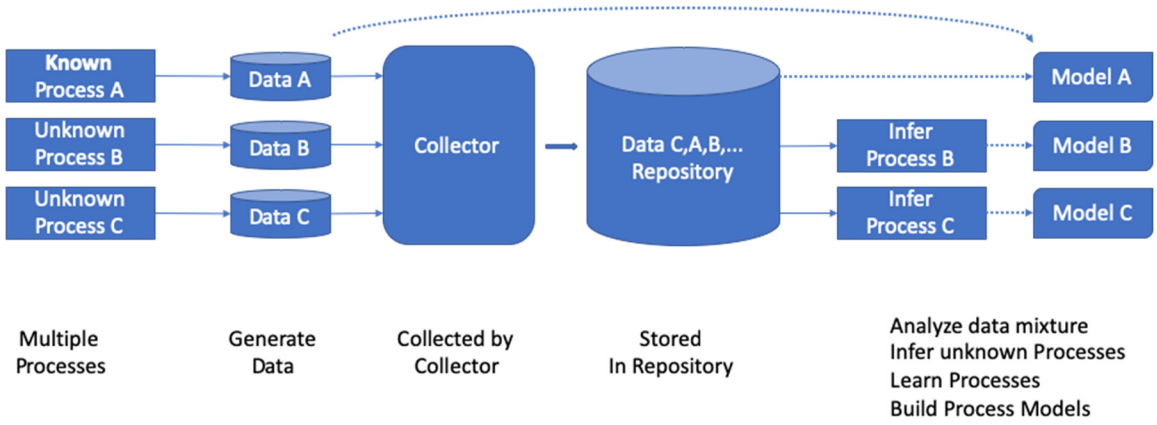

Figure 6

At this step, each process is discovered and modeled as a Stochastic Markov process. Each process is described by its flows transitioning matrix and flows distribution, as shown below in Figure 7.

Stochastic Flows Transitioning Matrix

		Followed By			
		FlowA	FlowB	FlowC	
This Flow	FlowA	1	0	0	1
	FlowB	0.5	0.3	0.2	1
	FlowC	0.05	0.95	0	1

Figure 7

Step 2: Generation of Training Data Using

In the second step, training data is generated using the process models from Step 1, above. For each process, training data may be generated using a 50/50 proportion of "Process"/"Non-Process." A transitional matrix may be used for each discovered process to generate 5,000 training data sequences (samples) of the "process" (Traffic). It may be possible to generate 5,000 total training samples from other processes, in order to keep 50/50 ratio of process/non-process.

These aspects may also include data augmentation to mix each sample with random noise (10-20%). In on example, a permutation could be repeated ten (10) times. In the end there will be 50,000 samples of the "process" and 50,000 samples of "Non-Process" for each process (e.g., repeat the above to generate a test data set for each process). For example, if there are ten (10) discovered processes, there will be 10 Training Data sets and 10 Test Data Sets. Each Training data set will have 100,000 samples in proportion 50/50 of "Process"/"Non-Process"

Step 3: Building a Classification Model

In the third step, a classification model is built using shallow Neural Networks (CONV1D or Simple RNN) and the data from step 2, above (i.e., train classification models for each process). Each classification model will classify only one process and provide a binary result of whether this is a process or not (One against-all). As a classification model, the techniques presented can use a simple recurrent neural network as well as Convolutional 1D models, although more complex recurrent neural networks such as GRU or LSTM may not produce better results than simple networks.

An example of a CONV 1D Network may be represented as:

```
model = keras_model_sequential() %>%
layer_embedding(input_dim = max_codes, output_dim = 64, name =
"EMBEDDING") %>%
layer_conv_1d(filters = 4, name = "CONV1D-1", kernel_size = 7,
activation = "relu") %>%
layer_global_max_pooling_1d() %>% # this layer flattens 3D to 2D
layer_dropout(0.5) %>%
layer_dense(units = 1, activation = "sigmoid", name = "OUTPUT")
```

An example of a simple RNN network may be represented as:

```
model = keras_model_sequential(name = "MODEL_RNN") %>%
layer_embedding(input_dim = max_codes, output_dim = 64, name =
"EMBEDDING") %>%
layer_simple_rnn(units = 128, name = "RNN-1",
dropout = 0.1, recurrent_dropout = 0.1,
return_sequences = FALSE) %>%
layer_dense(units = 1, activation = "sigmoid", name = "OUTPUT")
```

An example of a compile and train phase may be represented as:

```
compile(
model,
optimizer = "rmsprop",
loss = "binary_crossentropy",
metrics = c("acc")
)
history = fit(
model,
train_set, train_labels,
epochs = 10,
batch_size = 128,
validation_split = 0.2,
callbacks = list(early_stop),
shuffle = TRUE
)
```

Step 4: Detection

A challenge with the use of neural network for the classification is that a neural network is trained using samples of the desired Class and samples of Non-Class. If the classification model is to recognize a Process "Cat," then the model was presented thousands of variations of this "Cat" (e.g., in one example, samples or "pictures" of traffic from the process "Cat" mixed with some noise). A second model designed to recognize only the process "Dog" is trained in a similar manner using thousands of variations of the process "Dog." If the "Cat" model is presented traffic or "pictures" from process "Cat," then they can be perfectly recognized as "Cat," with the same result with the "Dog" model and process.

Moreover, even if pictures with "dog" and "cat" are presented next to each other, each model will be able to recognize what it has been trained to recognize (e.g., Model "Dog" will recognize "Dog", and Model "Cat" will recognize "Cat"). However, a problem arises from the fact that the real-life "pictures" of traffic are random mixtures/blends or double/triple/quadruple/etc. exposures of "dogs" and "cats", because collectors can mix the traffic. Such a mixed "picture" is neither a "dog" nor a "cat". Therefore neither of the models can recognize such mixed images. Figure 8, below, illustrates examples of this image analogy (e.g., Cat + Dog = Neither Dog Nor Cat) which are on-recognizable by any model.

Figure 8

To address this issued, the techniques presented herein carved the "picture" of the traffic into multiple overlapping segments using a moving window of relatively small size and then attempt to classify each segment separately. These small classifications can then be combined as a vote. The main idea behind this is that, if an image or sequence as a whole is recognizable as "dog", then the combination of classifications of multiple segments

should also be recognizable as "dog". This idea has been verified and confirmed using CONV1D and Simple Recurrent Neural Networks. Therefore, when there is a mixture, the system will try to classify each segment using the classification models, and then aggregate. The combined result will give us a distribution of what Signals/Processes where detected or recognized. This is shown below in Figure 9.

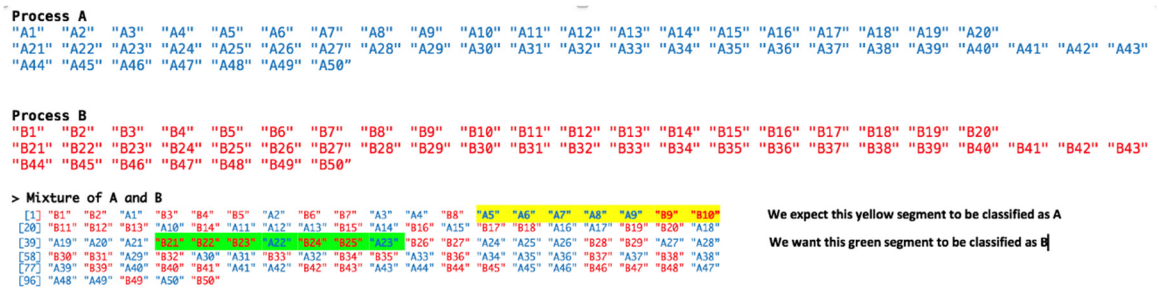

Figure 9

When a mixture of sequences of processes A and B are presented, Classifier A will detect Process A in 50% of segments, and Classifier B will detect Process B in 50% of segments. Each segment is classified as A or B with 70-80% guarantee. In the end the detection mechanism will provide an answer that there are two Processes in this traffic, namely A and B, as shown below in Figure 10.

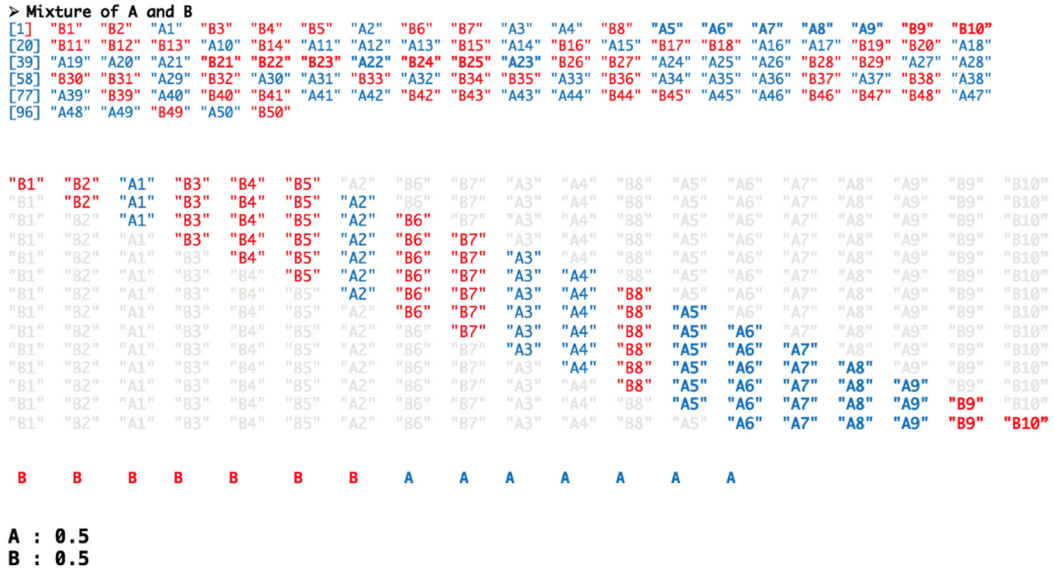

Figure 10

The following provides a series of detailed steps for the proposed solution:

#1 : Segment traffic sequence with overlapping segments, using moving window of W_len size and stride(shift) == 1.

#2 : Classify each segment by each RNN_Model or CONV1D_Model

p[3,2] : probability of Classifying Segment2 as Process3 by Neural Network RNN_Model3

	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Seg9
Process1	p[1,1]	p[1,2]		0.45			0.745		p[1,9]
Process2	p[2,1]	p[2,2]		0.8			0.75		p[2,9]
Process3	p[3,1]	p[3,2]		0.81			0.3		p[3,9]
Process4	p[4,1]	p[4,2]		0.47			0.56		p[4,9]

#3 : Apply credibility level threshold 0.75 (Configurable)

C_level = 0.75

If p[i,j] <= C_level then p[i,j] = 0 , then want the Classification Model to have high level of confidence.

	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Seg9
Process1	p[1,1]	p[1,2]		0			**0**		p[1,9]
Process2	p[2,1]	p[2,2]		0.8			**0**		p[2,9]
Process3	p[3,1]	p[3,2]		0.81			**0**		p[3,9]
Process4	p[4,1]	p[4,2]		**0**			**0**		p[4,9]

#4 : Calculate ARGMAX, which process gives the highest probability to each segment

If ALL probabilities are Zero, then assign Zero to argmax.

	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Seg9
Process1	p[1,1]	p[1,2]		0			0		p[1,9]
Process2	p[2,1]	p[2,2]		0.8			0		p[2,9]
Process3	p[3,1]	p[3,2]		0.81			0		p[3,9]
Process4	p[4,1]	p[4,2]		0			0		p[4,9]
argmax(seg)	p[4,1]	p[4,2]		**3**			**0**		

#5 : Mark <UNKNOWN> or undetectable segments

If argmax(i) == 0 then Seg[i] == <UNKN>

Hypothetical argmax(i):

	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Seg9
Process1	p[1,1]	p[1,2]		0			0		p[1,9]
Process2	p[2,1]	p[2,2]		0.8			0		p[2,9]
Process3	p[3,1]	p[3,2]		0.81			0		p[3,9]
Process4	p[4,1]	p[4,2]		0			0		p[4,9]
argmax(seg)	2	2	3	3	2	3	**<UNKN>**	3	1

#6 : Calculate distribution of detectible processes

Using argmax(seg):

2	2	3	3	2	3	<UNKN>	3	1

Calculate distribution of classification outcomes:

1	2	3	<UNKN>
0.1111111	0.3333333	0.4444444	0.1111111

#7 : Make conclusion(s)

Excluding <UNKN> segments, conclude :

1. Top One Process : Process 3
2. Top Two Processes : Process 3 and 2
3. There is trace of presence of Process 1, but it is too small to be considered.
4. Process 4 is not detected at all.

It is highly probable that data is generated by the Processes 2 and 3 out of four possible processes. By performing these classification outcomes, unusual processes, bad intentions, and malicious activity could also be detected. If these learning methodologies are applied, they could also be used to verify and perform attestation of abnormal behavior in network infrastructure devices and collectors (in the case that a network infrastructure device or telemetry collector is potentially compromised).