black hat EUROPE 2018

DECEMBER 3-6, 2018

EXCEL LONDON / UNITED KINGDOM

Deep Impact : Recognizing Unknown Malicious Activities from Zero Knowledge

Hiroshi Suzuki / Hisao Nashiwa
Internet Initiative Japan Inc.

Who Are We

- Hiroshi Suzuki and Hisao Nashiwa are from “Internet Initiative Japan Inc.” that is called “IIJ” for short.
  - IIJ is a Japanese ISP (We are the first commercial ISP in Japan).

  - We belong to “IIJ-SECT”, which is the CSIRT team of our company.
  - We are malware analysts and forensic investigators.

- We are past Black Hat Briefing speakers/coauthors (USA, Europe and Asia) and Trainers (USA).

What Is This Talk About?

- This presentation is about detecting malicious activities using deep learning in the absence of matching patterns, blacklists, behavioral analysis and other indicators.

- We cover the following detection methods.
  - C2 servers detection
  - Exploit kits detection

What Does "Zero Knowledge" Mean in This Presentation?

- In this presentation, we use the word “Zero Knowledge” in the sense of not using any IoCs (Indicator of Compromises) since we would not be able to detect anything if we were completely zero knowledge.

- You could be relieved from analyzing malicious samples and collecting IoCs by using our methods!

Agenda

- Introduction
- C2 Servers Detection
- Exploit Kits Detection
- Conclusion

Introduction

The Problems and the Motivation (1)

- In order to detect malicious activities, there are several existent approaches such as:
  - Pattern matching
  - Blacklists
  - Behavioral analysis
  - Event correlation

The Problems and the Motivation (2)

- However, they have several problems. For instance:
  - Unknown threats and sophisticated attacks could circumvent the solutions.
  - Some of them require huge resources and are very expensive.
- We want to establish a new detection method that does not rely on the approaches mentioned earlier, and also without incurring any additional costs.
  - This is our main motivation.

The Problems and the Motivation (3)

- On the other hand, proxy logs and firewall logs are not used that much in our daily routine. We only use for:
  - SIEM in particular cases
  - Anomaly detection (quantity of logs, new hosts, …)
  - Pattern matching when you get IoCs

- We want to utilize such logs more effectively.
  - This is our second motivation.

A Concern to Utilize Logs

- Log files are too huge to analyze!
  - They could be tons of gigabytes per a day!
  - It could have a lot of columns.

- Therefore, we decided to use deep learning to achieve the purpose.

Why Deep Learning?

- Deep learning could solve the problem because it can handle:
  - Huge samples (lines)
    - Tons of million samples.
  - Many features (columns)
    - For example, an image could have 256 x 256 x 3 = 196,608 features!

- It can also recognize patterns automatically.

C2 Servers Detection

Why Is C2 Servers Detection Difficult?

In order to detect C2 servers:

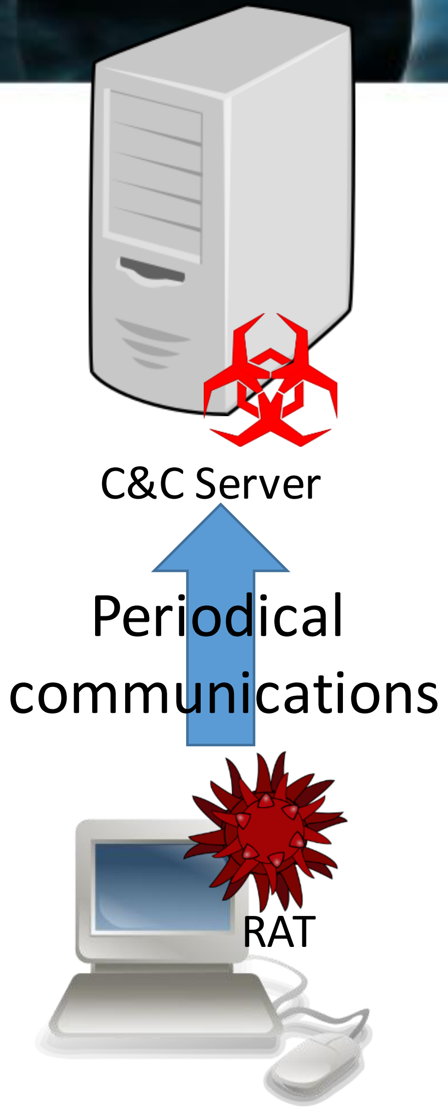
- We need to collect malware samples and analyze them.
  - Collecting malware is very hard especially in targeted attacks.
  - In order to extract IoCs, you need to analyze malware.
    - Sometimes, you need to deal with anti-analysis techniques.

- Attackers change IoCs frequently and easily.

How to Detect C2 Servers

How to Detect C2 Servers (1)

- A kind of malware such as Bots and RATs connects to C2 (C&C) servers frequently.
- The sort of malware checks commands from attackers or sends keep-alive message by accessing to C2 servers at some intervals.
  - Typical polling intervals are between 20 seconds and 7 minutes.



How to Detect C2 Servers (2)

- Intervals depend on malware’s developers or attackers.
  - If they choose a short interval:
    - Attackers will be able to move around freely, because they don’t need to wait for a long time.
    - However, malware can be detected easily because it communicates frequently.
  - If they choose a long interval:
    - It will be difficult for us to detect malware infection.
    - However, they will be limited to move because they have to wait for a long time.
  - This is a double-edged sword for them.

How to Detect C2 Servers (3)

- The following two examples express communications (1) between a benign client and a web server and (2) between an infected client and a C2 server in an hour.

(1) To a benign web server

	0	1	2	3	4	5	6	7	8	9
00	9	0	0	0	0	0	0	0	0	0
10	0	0	0	0	8	1	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0

(min)

(2) To a C2 server

	0	1	2	3	4	5	6	7	8	9
00	1	0	1	0	1	0	1	0	1	0
10	1	0	1	0	1	0	1	0	1	0
20	1	0	1	0	1	0	1	0	1	0
30	1	0	1	0	1	0	1	0	1	0
40	1	0	1	0	1	0	1	0	1	0
50	1	0	1	0	1	0	1	0	1	0

(min)

- It is rare for typical users to communicate periodically with a legitimate web server for a long time.

How to Detect C2 Servers (4)

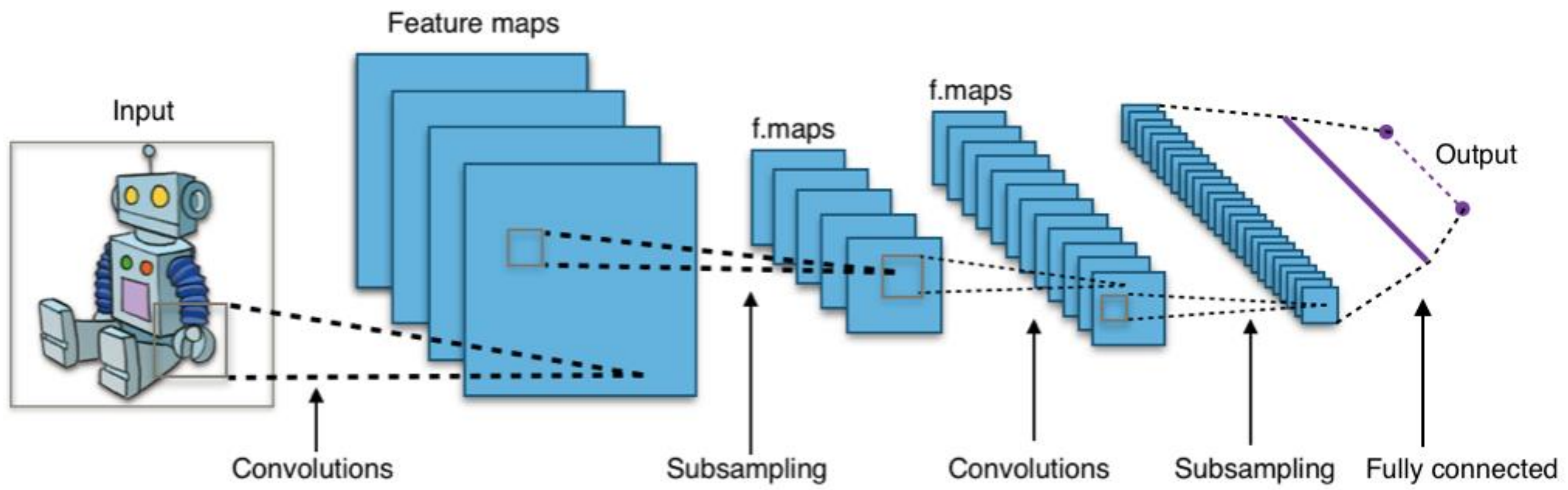
- Contrastingly, since Bots/RATs communicate at some intervals, you can see a "pattern" like vertical straight lines in the Example (2).

- Therefore we considered deep learning models are able to recognize the difference between benign and malicious communication patterns.

- We thought CNN (Convolutional Neural Network) can especially recognize the difference if we can image logs.

What are CNNs?

- CNN, which is an abbreviation for "Convolutional Neural Network", is a sort of deep neural networks.
- It is one of the best methods for image classification and several CNN models are already superior to human beings.

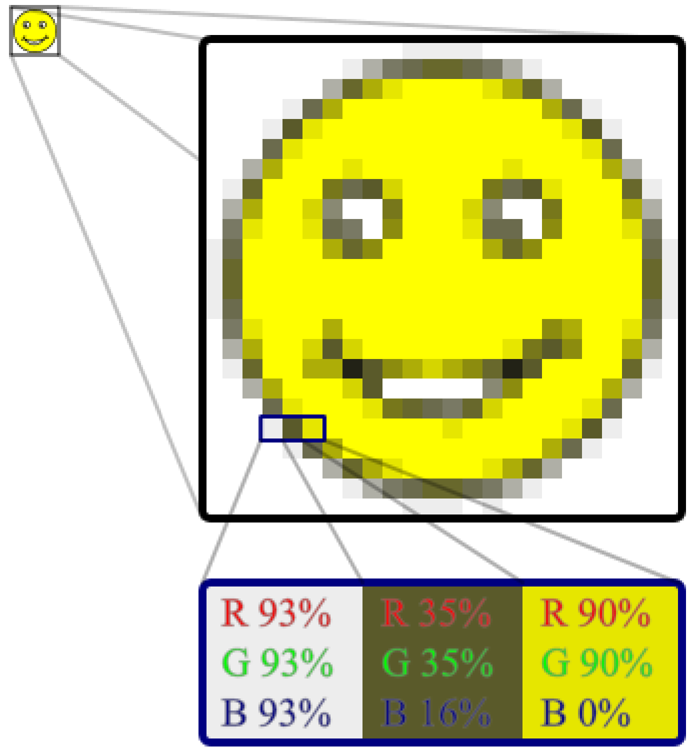


https://en.wikipedia.org/wiki/File:Typical_cnn.png

Converting Logs into "Images"

Converting Logs into an "Image" - A Graphical Image Structure

- In order to use CNNs, we need to convert logs into "images".
- A common graphical image (e.g. bitmap) structure consists of width, height and three color channels (RGB).
    - A mono-tone color image has width, height and a channel.



https://en.wikipedia.org/wiki/Raster_graphics

Converting Logs into an "Image" - Channels (1)

- However, since we generated images from logs we could not use degrees of RGB colors. Instead we needed to pick effective parameters for detection and use those as channels.
  - e.g. total numbers of communications during the time interval and sent/recv bytes

Total numbers of communications between 172.16.249.104 and example.com during the time interval

```
• 172.16.249.104 - [06/Jun/2016:05:01:32 +0900] "GET http://example.com/index.html HTTP/1.1"
  text/html 200 415 4666 - xxx.xxx.xxx.xxx
• 172.16.249.104 - [06/Jun/2016:05:01:32 +0900] "GET http://example.com/img1.jpg HTTP/1.1"
  image/jpeg 200 238 37349 - xxx.xxx.xxx.xxx
• 172.16.249.104 - [06/Jun/2016:05:01:33 +0900] "GET http://example.com/iimg2.jpg HTTP/1.1"
  image/jpeg 200 1521 57460 - xxx.xxx.xxx.xxx
```

Sent and received bytes.

Converting Logs into an "Image" - Channels (2)

- For example, let’s assume there are three logs below.
- If we pick only total numbers of communications during the time interval as a channel, the logs will be converted into the image on the right figure.

The total number of communications between 172.16.249.104 and example.com In 5:01 AM is three in this case.

```
• 172.16.249.104 - [06/Jun/2016:05:01:32 +0900]
  "GET http://example.com/index.html HTTP/1.1"
  text/html 200 415 4666 - xxx.xxx.xxx.xxx

• 172.16.249.104 - [06/Jun/2016:05:01:32 +0900]
  "GET http://example.com/img1.jpg HTTP/1.1"
  image/jpeg 200 238 37349 - xxx.xxx.xxx.xxx

• 172.16.249.104 - [06/Jun/2016:05:01:33 +0900]
  "GET http://example.com/iimg2.jpg HTTP/1.1"
  image/jpeg 200 1521 57460 - xxx.xxx.xxx.xxx
```

172.16.249.104 -> example.com

	0	1	2	3	4	5	6	7	8	9 (min)
00	0	3	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0

10 (W) * 6 (H) * 1 (C) Image
(Count per a minute with an hour window)

Converting Logs into an "Image" - Channels (3)

- We tested three patterns of parameters extraction.
  - Pattern A (3 channels)
    - total numbers of communications during the time interval
    - Averages of sent bytes during the time interval
    - Averages of received bytes during the time interval

(min)

	0			1			2			3			4			5			6			7			8			9		
00	0	0	0	3	136	522	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10 (W) * 6 (H) * 3 (C) Image (an hour window)

Converting Logs into an "Image" - Channels (4)

- Pattern B
  - total numbers of communications (20 if count > 20)

	0	1	2	3	4	5	6	7	8	9
00	0	3	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0

- Pattern C
  - Communication flags (1 if count >= 1, or 0)

	0	1	2	3	4	5	6	7	8	9
00	0	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0

Pattern B and C got nice results against in-the-wild malware.

Converting Logs into an "Image" - Width and Height

- We also tested these parameters below for dimensions (width and height of a “image”).
  - Aggregating data every:
    - 10 seconds (W * H = 360 pixels)
    - 30 seconds (W * H = 120 pixels)
    - minute (W * H = 60 pixels)
    - five minutes (W * H = 12 pixels)
    - ten minutes (W * H = 6 pixels)
- The last two were not able to distinguish malicious and benign samples.
- We have chosen the third option (aggregating data every minute) because it offers the best results against in-the-wild malware.

Datasets

Training dataset

- Benign data
  - We used over 1.5 million "images" that are converted from approximately 3.7 GB proxy logs.

- Malicious data
  - We generated over 1 million C2 like communications with a simple script.
    - We didn't use any actual malware traffic. That's why we call this method "Zero Knowledge".
    - We mention this later.

Testing datasets

- Benign data
  - We used approximately 4.5 million of benign images in total that are converted from about 11 GB proxy logs.
- Malicious data
  - We used in-the-wild malware communications (e.g. PlugX, xxmm, RedLeaves, KINS, Dreambot/ursnif and so on) from actual incidents.

Testing datasets – Malware families

- We prepared the eleven malware families below at this time.
    - PlugX
    - Asruex
    - xxmm
    - himawari/ReadLeaves
    - ChChes
    - Elirks
    - Logedrut
    - ursnif/gozi
    - Shiz/Shifu
    - Vawtrak
    - KINS

Generated C2ish Patterns for Training (1)

- The script outputs a variety of periodical patterns starting from once in three seconds (20 counts in every minute) and going up to once in every three minutes, building up in 100 milliseconds increments.

	0	1	2	3	4	5	6	7	8	9
0	20	20	20	20	20	20	20	20	20	20
10	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20
30	20	20	20	20	20	20	20	20	20	20
40	20	20	20	20	20	20	20	20	20	20
50	20	20	20	20	20	20	20	20	20	20

20 counts in every minute

…

	0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	0	0	1	0	0	1
10	0	0	1	0	0	1	0	0	1	0
20	0	1	0	0	1	0	0	1	0	0
30	1	0	0	1	0	0	1	0	0	1
40	0	0	1	0	0	1	0	0	1	0
50	0	1	0	0	1	0	0	1	0	0

Three-minute basis

Generated C2ish Patterns for Training (2)

- The script also outputs sparse patterns starting from every three minutes and going up to every twelve minutes, building up in 10 seconds increments.

	0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	0	0	1	0	0	1
10	0	0	1	0	0	1	0	0	1	0
20	0	1	0	0	1	0	0	1	0	0
30	1	0	0	1	0	0	1	0	0	1
40	0	0	1	0	0	1	0	0	1	0
50	0	1	0	0	1	0	0	1	0	0

Three-minute basis

…

	0	1	2	3	4	5	6	7	8	9
0	1	0	0	0	0	0	0	0	0	0
10	0	0	1	0	0	0	0	0	0	0
20	0	0	0	0	1	0	0	0	0	0
30	0	0	0	0	0	0	1	0	0	0
40	0	0	0	0	0	0	0	0	1	0
50	0	0	0	0	0	0	0	0	0	0

twelve-minute basis

Generated C2ish Patterns for Training (3)

- Since there are some samples that sleep for several minutes after connecting to C2 servers frequently, the script generates similar patterns in advance.

	0	1	2	3	4	5	6	7	8	9
0	1	1	0	0	1	1	0	0	1	1
10	0	0	1	1	0	0	1	1	0	0
20	1	1	0	0	1	1	0	0	1	1
30	0	0	1	1	0	0	1	1	0	0
40	1	1	0	0	1	1	0	0	1	1
50	0	0	1	1	0	0	1	1	0	0

2 min. sleep after 2 min. of activity

	0	1	2	3	4	5	6	7	8	9
0	1	1	1	0	0	0	1	1	1	0
10	0	0	1	1	1	0	0	0	1	1
20	1	0	0	0	1	1	1	0	0	0
30	1	1	1	0	0	0	1	1	1	0
40	0	0	1	1	1	0	0	0	1	1
50	1	0	0	0	1	1	1	0	0	0

3 min. sleep after 3 min. of activity

…

Generated C2ish Patterns for Training (4)

- Based on the patterns that have been generated so far, the following two methods were also used to better resist CNN attacks [1] and to better detect similar connection patterns.
  - Rotation
  - Random noise
    - [1] Simple Black-Box Adversarial Perturbations for Deep Networks
      - https://arxiv.org/abs/1612.06299

Generated C2ish Patter (5)

- Rotation

	0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	0	0	1	0	0	1
10	0	0	1	0	0	1	0	0	1	0
20	0	1	0	0	1	0	0	1	0	0
30	1	0	0	1	0	0	1	0	0	1
40	0	0	1	0	0	1	0	0	1	0
50	0	1	0	0	1	0	0	1	0	0

Original image
(every three minute)

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	1	0	0
10	1	0	0	1	0	0	1	0	0	1
20	0	0	1	0	0	1	0	0	1	0
30	0	1	0	0	1	0	0	1	0	0
40	1	0	0	1	0	0	1	0	0	1
50	0	0	1	0	0	1	0	0	1	0

One-minute rotation

	0	1	2	3	4	5	6	7	8	9
0	0	0	1	0	0	1	0	0	1	0
10	0	1	0	0	1	0	0	1	0	0
20	1	0	0	1	0	0	1	0	0	1
30	0	0	1	0	0	1	0	0	1	0
40	0	1	0	0	1	0	0	1	0	0
50	1	0	0	1	0	0	1	0	0	1

Two-minute rotation

Generated C2ish Patterns for Training (6)

- Random noise

	0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	0	0	1	0	0	1
10	0	0	1	0	0	1	0	0	1	0
20	0	1	0	0	1	0	0	1	0	0
30	1	0	0	1	0	0	1	0	0	1
40	0	0	1	0	0	1	0	0	1	0
50	0	1	0	0	1	0	0	1	0	0

Original image
(every three minute)

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	0	1	0	0
10	1	0	0	1	0	0	1	0	0	1
20	0	0	1	0	0	1	0	0	1	0
30	0	1	0	0	1	0	0	1	0	0
40	0	0	0	1	0	0	0	0	0	1
50	0	0	1	0	0	1	0	0	1	0

Random noised image

Our Model

One of the Best Models We Created (1)

Input shape = (Width = 60, Height = 1, Channel = 1)

```
def build_model_base(input_shape):
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(10, 1), activation='relu',
                input_shape=input_shape))
    model.add(Conv2D(64, (3, 1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 1)))
    model.add(Dropout(0.25))
    model.add(Conv2D(128, (3, 1), activation='relu'))
    model.add(Conv2D(256, (1, 1), activation='relu'))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))
    loss_func = keras.losses.binary_crossentropy
    return model, loss_func
```

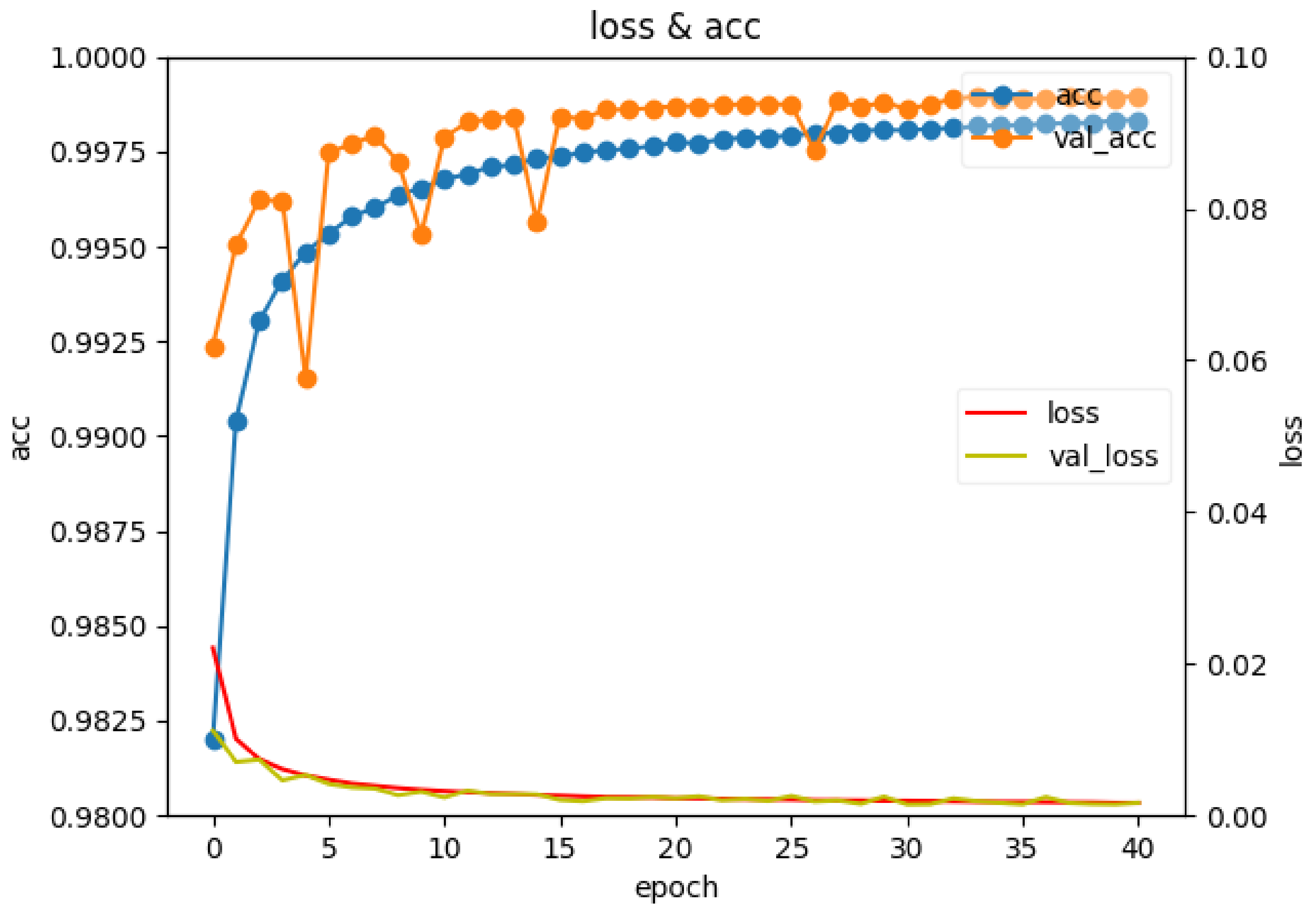
We use Keras with TensorFlow backend.

One of the Best Models We Created (3)

- We also do the following things to get better results.
  - Shuffling the training dataset before training
  - Feature scaling (Standardization)

- Other configuration values
  - Batch size = 1000
  - Epochs = 100
  - Class weight = 0.2
  - Optimizer = Adadelta
  - Early stop = 10

The Result for The Training Dataset



40th epoch	Accuracy	Loss
Training	0.998327	0.001677
Validation	0.998970	0.001697

The Results

The Results of Benign Web Sites

- The first testing dataset
  - Accuracy: 1,565,139/1,566,109 (99.94%)
  - False positive FQDNs: 64/246,190
- The second testing dataset
  - Accuracy: 1,540,419/1,541,050 (99.96%)
  - False positive FQDNs: 72/243,106
- The third testing dataset
  - Accuracy: 1,528,936/1,529,617 (99.96%)
  - False positive FQDNs: 65/243,185

There are small numbers of false positive FQDNs so that you can filter out them with a whitelist.

The Results of In-The-Wild Malware Families (from Actual Incidents) (1)

- Our model is able to detect all of the following eleven malware families prepared at this time.
  - PlugX
  - Asruex
  - xxmm
  - himawari/ReadLeaves
  - ChChes
  - Elirks
  - Logedrut
  - ursnif/gozi
  - Shiz/Shifu
  - Vawtrak
  - KINS

The Results of In-The-Wild Malware Families (from Actual Incidents) (2)

- PlugX

Pattern (1)

	0	1	2	3	4	5	6	7	8	9
0	6	6	0	3	6	3	0	6	6	0
10	3	6	3	0	6	6	0	3	7	2
20	0	6	6	0	3	7	2	0	6	6
30	0	3	8	1	0	6	6	0	3	8
40	1	0	6	6	0	3	8	1	0	6
50	6	0	4	8	0	0	7	5	0	5

Pattern (2)

	0	1	2	3	4	5	6	7	8	9
0	96	95	92	96	95	97	96	97	101	95
10	97	93	95	96	95	98	92	93	95	100
20	96	95	94	93	88	98	95	97	97	96
30	97	88	94	96	94	101	98	97	97	96
40	95	95	91	93	91	101	96	100	97	89
50	92	94	96	98	94	98	98	92	94	95

The Results of In-The-Wild Malware Families (from Actual Incidents) (3)

- Asruex

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	4
10	0	2	0	2	0	2	0	0	2	0
20	0	2	2	0	2	0	0	2	0	2
30	0	2	0	0	2	0	2	0	2	0
40	0	2	0	0	2	0	0	2	0	2
50	2	0	2	0	2	0	2	2	0	2

- xxmm

	0	1	2	3	4	5	6	7	8	9
0	3	3	2	0	0	1	3	3	3	3
10	3	3	3	3	3	3	3	3	2	3
20	3	3	3	3	3	3	3	3	3	3
30	3	3	3	3	3	3	3	3	3	3
40	3	3	3	3	3	3	3	3	3	3
50	3	3	3	3	3	2	0	0	1	3

The Results of In-The-Wild Malware Families (from Actual Incidents) (4)

- himawari/ReadLeaves

	0	1	2	3	4	5	6	7	8	9
0	2	1	1	1	2	0	0	0	2	2
10	0	1	1	2	0	1	1	1	3	2
20	0	2	0	1	1	2	2	1	1	1
30	2	0	0	0	1	1	1	2	1	0
40	1	3	1	1	1	1	0	0	0	0
50	0	1	1	0	2	1	1	3	0	1

- ChChes

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	0	0	0	0	1	0
10	0	0	0	0	1	0	0	0	0	1
20	0	0	0	0	0	1	0	0	0	0
30	1	0	0	0	0	0	1	0	0	0
40	0	1	0	0	0	0	0	1	0	0
50	0	0	1	0	0	0	0	0	1	0

The Results of In-The-Wild Malware Families (from Actual Incidents) (5)

- Elirks

Pattern (1)

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0	0
10	1	0	0	0	0	0	1	0	0	0
20	0	0	1	0	0	0	0	1	0	0
30	0	0	0	1	0	0	0	0	1	0
40	0	0	0	1	0	0	0	0	1	0
50	0	0	0	1	0	0	0	0	1	0

Pattern (2)

	0	1	2	3	4	5	6	7	8	9
0	1	0	1	1	0	1	1	0	1	1
10	0	1	1	0	1	0	1	0	1	0
20	1	0	1	0	1	0	0	1	1	0
30	0	1	1	0	0	1	1	0	0	1
40	1	0	0	1	0	1	0	1	0	1
50	0	1	0	1	0	0	1	1	0	0

The Results of In-The-Wild Malware Families (from Actual Incidents) (6)

- Logedrut

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	1	0	0
20	0	0	0	0	0	0	0	0	0	1
30	0	0	0	0	0	0	0	0	0	0
40	1	0	0	0	0	0	0	0	0	0
50	0	1	0	0	0	0	0	0	0	0

The Results of In-The-Wild Malware Families (from Actual Incidents) (7)

- Ursnif/gozi

Pattern (1)

	0	1	2	3	4	5	6	7	8	9
0	2	0	0	0	0	0	0	0	0	1
10	1	0	1	0	2	1	0	0	0	1
20	2	0	0	0	0	1	1	0	0	1
30	1	0	0	0	1	2	0	0	1	1
40	1	0	0	0	2	1	0	0	0	1
50	3	0	0	0	1	1	1	0	0	0

Pattern (2)

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	1	1	0	1	0	1	0
10	1	1	0	1	0	2	0	0	0	1
20	0	1	0	1	1	1	0	0	1	1
30	1	0	0	1	1	1	0	0	0	1
40	1	0	0	0	1	1	0	0	0	1
50	0	1	0	0	0	1	1	0	0	0

The Results of In-The-Wild Malware Families (from Actual Incidents) (8)

- Shiz/Shifu

	0	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1	1	0
10	1	1	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1	1	1	1
50	1	1	1	1	1	1	1	0	1	1

- Vawtrak

	0	1	2	3	4	5	6	7	8	9
0	1	0	0	0	1	1	0	0	0	1
10	1	0	0	1	1	0	0	0	1	1
20	0	0	0	1	1	0	0	0	1	0
30	1	0	0	1	0	1	0	0	1	0
40	1	0	0	1	0	1	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0

The Results of In-The-Wild Malware Families (from Actual Incidents) (9)

- KINS

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	2	4	0	0	0	0	0
10	0	0	2	0	0	0	0	0	0	2
20	0	0	0	0	0	0	2	0	0	0
30	0	0	2	0	0	0	0	0	0	2
40	0	0	0	0	0	0	2	0	0	0
50	0	0	2	0	0	0	0	0	0	2

Summary for This Section

Summary for This Section

- Our simple CNN model can detect malicious communications with sufficiently high performance.
  - The result for our test datasets is high accuracy (99.95%) as well as low false positives (0.03%).
  - Against in-the-wild malware, our model can detect all eleven malware families that we tested.

- To accommodate various patterns, an image size should not be too big.
  - Even though it seems to work well at first glance, there is a possibility that it will not detect if a pattern changes slightly.

- To rotate a pattern and to mix noise could reduce false negatives.

- False positive measures are required somewhat.

To Apply Real World Environments

- Some external web sites such as the listed below, or internal servers (Proxy, mail), might cause false positives because of frequent reloading or accessing.
  - Cloud storages
  - Sports news sites
  - Stock markets
  - Web mails
  - SNS
  - Web analytics
- These web sites should be filtered with whitelists in advance. Or, filter them if the same alerts come up from many clients.

Exploit Kits Detection

Existing Approaches: Pattern matching (1)

- Pattern matching
  - URL pattern matching
  - Content pattern matching

- Each exploit kit has characteristics in its URLs and contents.

A URL path sample of Rig EK at Feb 2017

```
/?yus=Microsoft_Edge.113tl68.406b2j3b4&biw=Microsoft_Edge.95hj111.406s5j6s2&br_fl=3833&q=wXzQMvX
cJwDQAobGMvrESLtENknQA0KK2Iz2_dqyEoH9c2nihNzUSkry6B2aCm2&oq=E9_orfrdYOVHii02HKA1plIxZAQtAof
r9jknSzkDP0pGH-xaFUQ9G95CSF4F4nws&tuif=4980&ct=Microsoft_Edge
```

A regex pattern for Rig EK at Feb 2017

```
/\?(((oq|q)=[0-9a-zA-Z_\-]{50,}|(aqs|biw|yus)=[0-9a-zA-Z\._]+?|(sourceid|ct)=[a-zA-
Z_]+?|(es_sm|tuif|br_fl)=[0-9]+?|(ie|browser)=[0-9a-zA-Z\-]+?)&?){5,}$
```

Existing Approaches: Pattern matching (2)

- Pattern matching
  - URL pattern matching
  - Content pattern matching

- Problems
  - Characteristics of URLs and contents have been constantly changing.
  - Parts of their URLs are often randomized, and it's difficult to collect whole patterns of ongoing exploit kits.
  - Contents are heavily obfuscated and often contain meaningless sentences.

Existing Approaches: Behavioral analysis (1)

- Behavioral analysis
  - Sandboxes can detect exploits and malware infection by actually browsing the target website.

Existing Approaches: Behavioral analysis (2)

- Behavioral analysis
  - Sandboxes can detect exploits and malware infection by actually browsing the target websites.

- Problems
  - A sandbox requires a variety of web browsers and their plug-ins to detect malicious activities. It's impossible to cover all of combinations.
  - Behavioral analysis typically spends several minutes on each URL.
  - Exploit kits could detect web browsers and OS environments, and they could have several sandbox detection and evasion techniques.

Our Strategy

- Typically, exploit kits' servers send contents in the following order.
  1. Landing pages: detect a web browser and its plug-ins' version. They often contain anti-virus evasion, web browser exploits and so on.
  2. Exploit contents: for web browsers and its plug-ins such as SWF, PDF, Java and Silverlight files
  3. Payloads: are malware to be loaded by exploits.
- We built models to detect EKs with proxy logs by focusing on this content-type transition.
- Also, we tried to detect EKs with characteristics of URLs from proxy logs at first.
- From the next slide, we will explain the latter method first. Then, we will explain the former method.

Detecting Rig Exploit Kit with DNNs

Concept (1)

- In many cases, exploit kits' URLs may seem unfamiliar.

An exploit URL path samples:

1. Rig Exploit Kit (23rd Feb, 2017)

```
/?yus=Microsoft_Edge.113tl68.406b2j3b4&biw=Microsoft_Edge.95hj111.406s5j6s2&br_fl=3833&q
=wXzQMvXcJwDQAobGMvrESLtENknQA0KK2Iz2_dqyEoH9c2nihNzUSkry6B2aCm2&oq=E9_orfrdYOV
Hii02HKA1plIxZAQtAofr9jknSzkDP0pGH-xaFUQ9G95CSF4F4nws&tuif=4980&ct=Microsoft_Edge
```

2. Nebula Exploit Kit (23rd Feb, 2017)

```
/4325/5421.swf
```

3. Sundown Exploit Kit (7th Mar, 2017)

```
/0E2/?947545190441
```

Concept (2)

- Exploit kits' URLs have some of the following characteristics.
  - Paths and queries:
    - Have a lot of directories or parameters.
    - Consist of meaningless words or numbers.
  - Hostnames:
    - Have strange TLDs that you do not usually use in your country.
    - Have long and / or meaningless domain names.
  - Servers:
    - Are located in strange countries.

Concept (3)

- Deep neural networks (DNNs) could learn features of exploit kits' URLs.
  - In this sub section, we will use the term Deep Neural Network (DNN) as Multilayer perceptron (MLP).

Concept (4)

- Our prime target was to detect Rig Exploit Kit with supervised learning.
  - That was because Rig Exploit Kit has been the most popular exploit kit over the last few years, and we have collected many samples to prove our theory.

Vectorizing Features of URLs

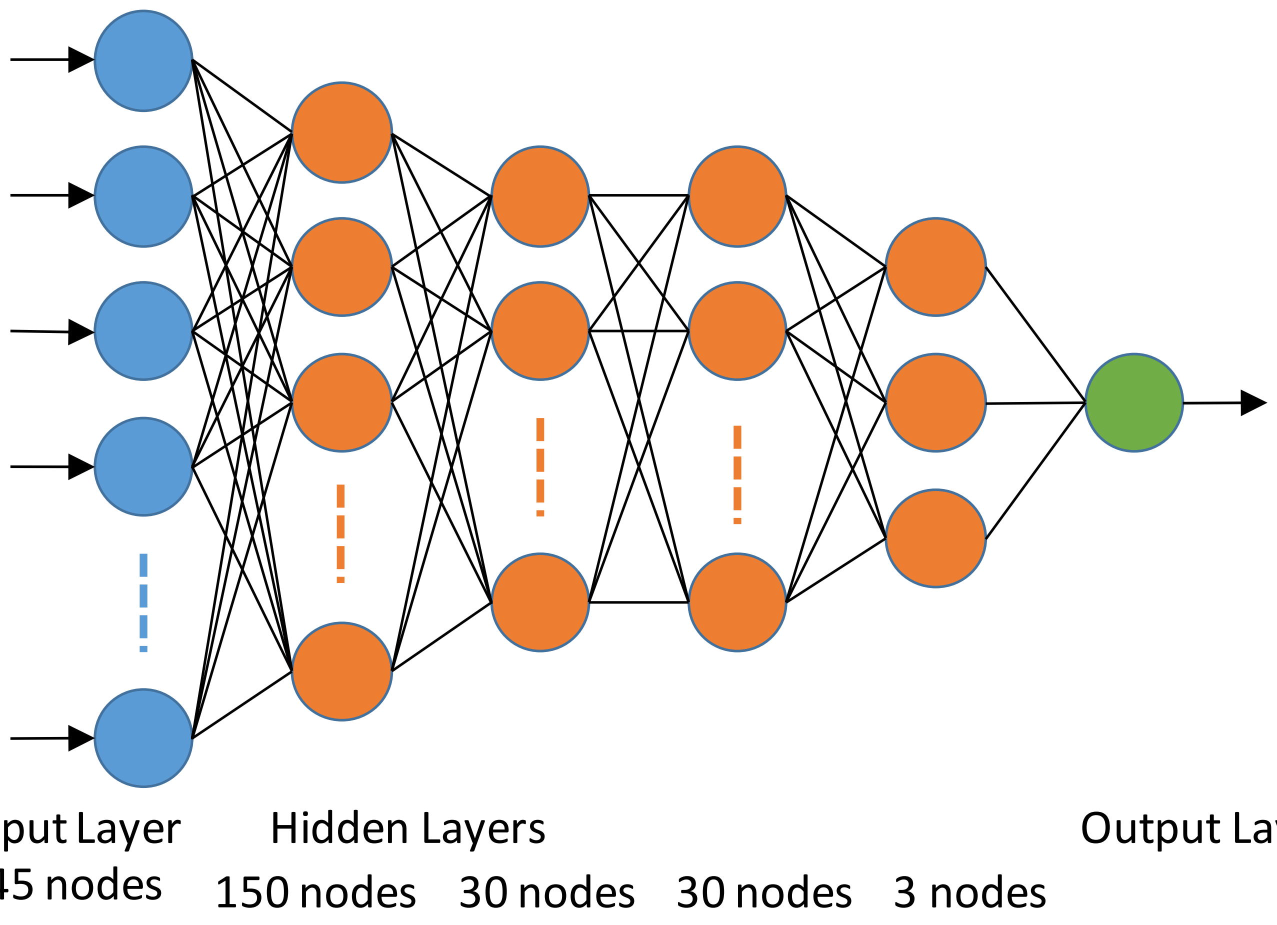
- We converted each line of proxy logs into a feature vector that has 345 dimensions for supervised learning.
- The underlined features could express characteristics of EKs' URLs.

- AS Country code[56]* (56 countries)
- HTTP Method[5]* (5 methods)
- content_type[83]* (83 types)
- extension[113]* (113 extensions)
- Does an User-Agent contain "mozilla"?
- Does a Referer exist?
- Do a Referer and URL contain the same domain?
- Number of slashes[11]** (11 classes)
- Number of query parameters[11]** (11 classes)
- Length of FQDN[8]** (8 classes)
- Length of sub-domain[8]** (8 classes)
- Length of path[7]** (7 classes)
- Length of query parameters[7]** (7 classes)
- Length of User-Agent[7]** (7 classes)
- Size of received data[13]** (13 classes)
- Size of sent data[13]** (13 classes)

* We converted a raw value into a feature vector with One-Hot-Encoding.
** We used a frequency distribution instead of a raw value.

Our DNN Model

- 5-layered fully connected DNN model
  - Activation: Relu
  - Dropout: 0.2



Datasets & Test Results

- Training dataset
  - 2,058,232 lines of proxy logs
  - collected from January to February 2017
  - 26,406 positive samples of Rig EK
- Test dataset
  - 2,011,816 lines of proxy logs
  - collected in March 2017
  - 4,098 positive samples of Rig EK
- Test result
  - 0.9999 accuracy and 1.000 precision

Problems

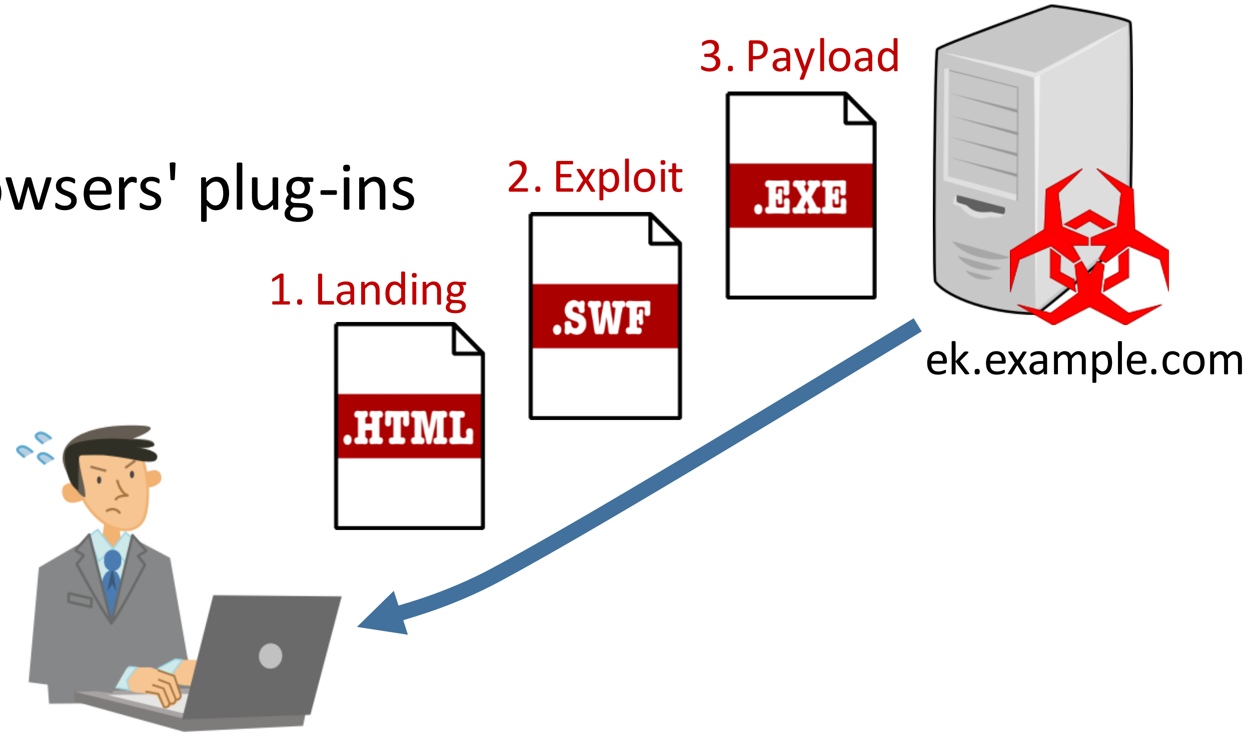
- The model could not detect other EKs such as Nebula and Sundown.
- That is because their characteristics in URLs are very different from each other.

- The model is still useful to detect variants and to trace small changes, but it could not detect unlearned EKs.
- URLs of each EK could be dramatically changed since they are not important for EK's functions and purposes.

Detecting Unlearned Exploit Kits with RNNs

Contents-types in Exploit Kits' Traffic (1)

- Each exploit kit's server sends contents in the following order.
  1. Landing page
     - **text/html**
  2. Exploits for web browsers or browsers' plug-ins
     - **application/x-shockwave-flash**
     - **application/x-java-archive**
     - **application/x-silverlight-app**
     - **application/pdf**
     - etc.
  3. Payload (Malware)
     - **application/octet-stream**
     - **application/x-msdownload**



Contents-types in Exploit Kits' Traffic (2)

- A URL path and content-type transition sample of Rig EK

Substance	Content-type	Path & Parameters of URL
Landing Page	**text/html**	/?NTI0OTU5&RCDUIv&oJhtJNm=dGFraW5n&wouMDc=Y2FwaXRhbA==&JgtXjOEttIAHrI=Y2FwaXRhbA==&TKCcodYFxdiy=dGhpbmdz&tNDodvGjF=Y2FwaXRhbA==&pHtonQrvp=bG9jYXRlZA==&kl345dfdfg234fsd=UDQTpjkGELQNmyN9ZAF1G9P2s3EeBzhWZiMHT-RTZZA4QrZSQR7Rt3VzyxrckQPskg1TH6mI&pWjLlCBUIUSRIw=Y2FwaXRhbA==&nR45dsgd54lsCs=xXrQMvWfbRXQDJ3EKvjcT6NAMVHRGUCL2YqdmrHXefjaf1WkzrfFTF_3ozKATASG6_ZtdfJ

Substance	Content-type	Path & Parameters of URL
Flash Exploit	**application/x-shockwave-flash**	/?NTQ0NjEw&zWuWFX&lskPeVWn=dW5rbm93bg==&NCDmQdmxCxapA=dW5rbm93bg==&eLCxfNVxDhHqBH=Y29uc2lkZXI=&nzZHzkCNdL=cmVwb3J0&HZELKhjPUenym=cG9wdWxhcg==&nR45dsgd54lsCs=wnrQMvXcKxXQFYbDKuXDSKZDKU7WG0aVw4-dhMG3YpjNfynz1ezURnL1tASVVFiRrbMdKL&kl345dfdfg234fsd=VYOQfk20LUKgEzm9sJVFhBo66tjUmDmBCd1JLX-UeLMg9DqZOSHbIL0Vz0zLMRQIgigECy&rZpDUeqxIDnMQL=bG9jYXRlZA==&LENxPZQZ=cmVwb3J0

Substance	Content-type	Path & Parameters of URL
Payload / Malware	**application/x-msdownload**	/?MjEwNzA1&mTONXmiGJttk&nR45dsgd54lsCs=wXrQMvXcJwDQDobGMvrESLtGNknQA0KK2Iv2_dqyEoH9fWnihNzUSkr16B2aCm3W&UEiQzsUEYQeeS=Y2FwaXRhbA==&jeeGWAgbhZSFoHh=bG9jYXRlZA==&KRssZN=bG9jYXRlZA==&BWeciQaXKEgAey=bG9jYXRlZA==&SOymAmL=cG9wdWxhcg==&uLNyyCiGt=cG9wdWxhcg==&wlNBeZFOQXgP=dW5rbm93bg==&kl345dfdfg234fsd=_fcpKeRXaVKziULVLwczyIlbUVJFpqj6i0SAmxDPhcGD_hKEUQ1M-5KREYFmmF7F

Contents-types in Exploit Kits' Traffic (3)

- A URL path and content-type transition sample of Neutrino EK

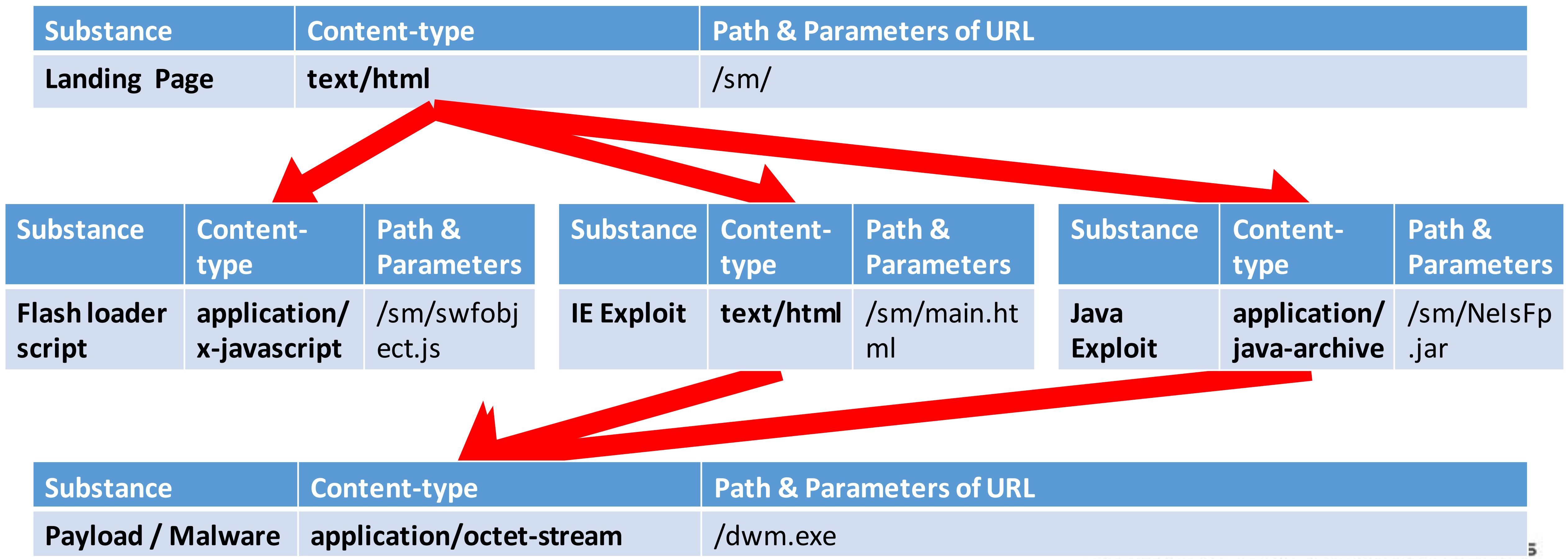
Substance	Content-type	Path & Parameters of URL
Landing Page	**text/html**	/bathroom/Zmhzbm9ncGc

Substance	Content-type	Path & Parameters of URL
Flash Exploit	**application/x-shockwave-flash**	/husband/1055103/grey-powder-lock.swf

Substance	Content-type	Path & Parameters of URL
Payload / Malware	**application/octet-stream**	/assemble/true-steady-23092006

Contents-types in Exploit Kits' Traffic (4)

- A URL path and content-type transition sample of KaiXin EK



Substance	Content-type	Path & Parameters of URL
Landing Page	**text/html**	/sm/

Substance	Content-type	Path & Parameters
Flash loader script	**application/x-javascript**	/sm/swfobject.js

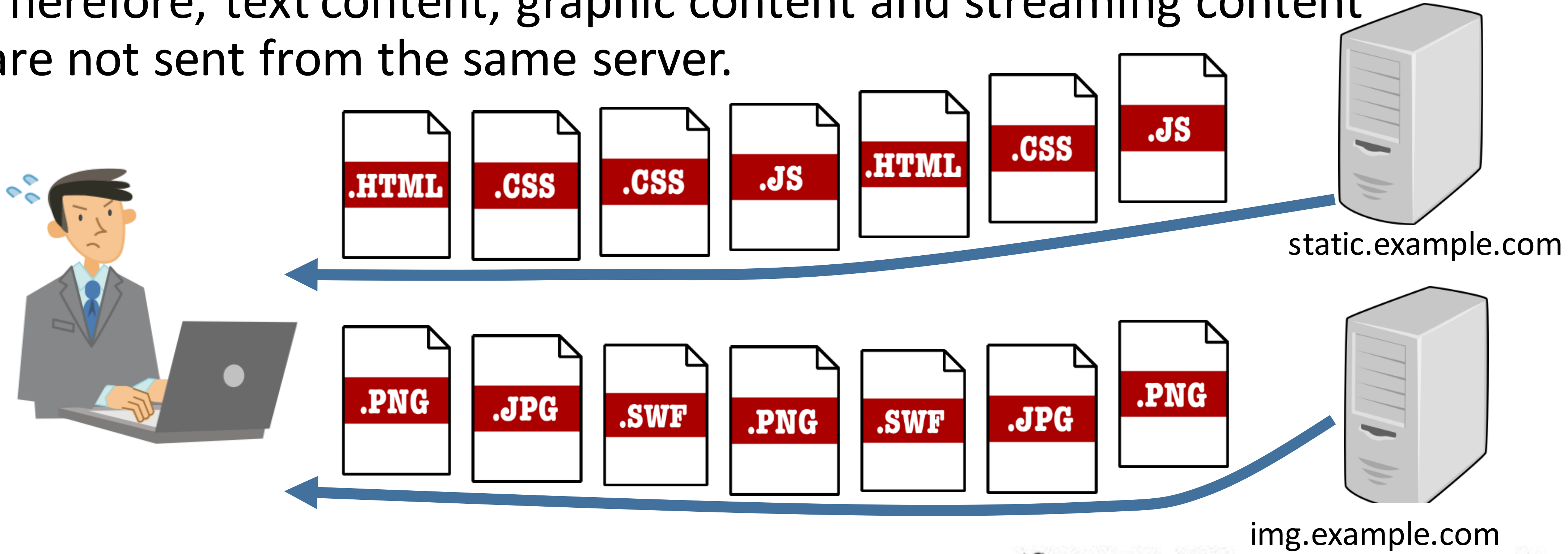
Substance	Content-type	Path & Parameters
IE Exploit	**text/html**	/sm/main.html

Substance	Content-type	Path & Parameters
Java Exploit	**application/java-archive**	/sm/NeIsFp.jar

Substance	Content-type	Path & Parameters of URL
Payload / Malware	**application/octet-stream**	/dwm.exe

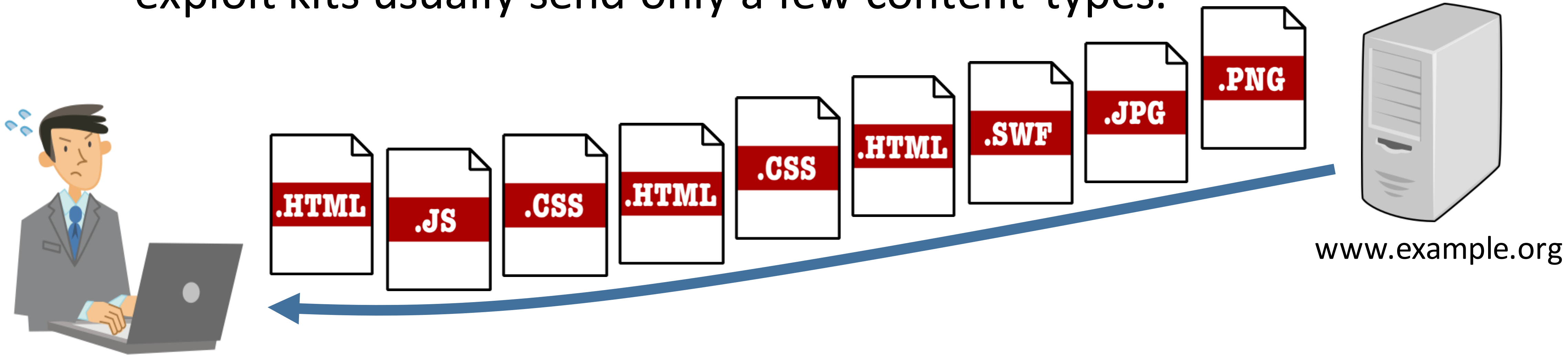
Contents-types in Benign Traffic (1)

- Large services
  - They usually prepare dedicated servers for each content-type. Therefore, text content, graphic content and streaming content are not sent from the same server.



Contents-types in Benign Traffic (2)

- Private / small services
  - Typically, they use a single server for all content-types. Thus, many static text and image contents are sent from the same server. Static image contents are not usually used in exploit kits, and exploit kits usually send only a few content-types.



Concept

- Focusing on content-type sequences

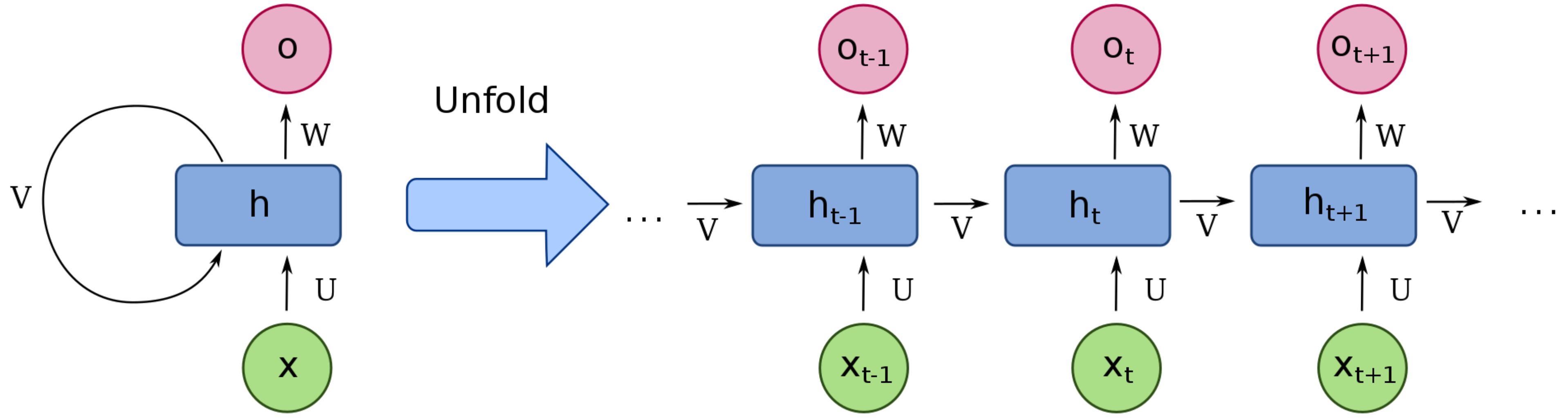
A typical sequence of content-type that is send from exploit kits' infection server.

1. **Landing page:** text/html
2. **Exploit content:** application/x-shockwave-flash, application/x-java-archive, etc...
3. **Payload:** application/octet-stream, application/x-msdownload

- Content-type sequences are strongly related to exploit kits' fundamental functions, and it is difficult to change the sequence pattern.
- It is possible to detect exploit kits by checking content-type sequences from each web server with recurrent neural networks (RNNs).

What are RNNs?

- RNN is a class of artificial neural network and it is widely used to process natural languages and time series data such as audio waveform and video stream.
- Each output of a hidden layer node is used with the next data in a sequence again. It enables RNN to remember its status in relationship to the previous data. Therefore, RNNs can recognize the order of contents in each sequence.



https://en.wikipedia.org/wiki/Recurrent_neural_network#/media/File:Recurrent_neural_network_unfold.svg

Sequencing Proxy Logs (1)

- We converted proxy logs into feature vector sequences in the following ways. We also set the length of each sequence as five.
  1. We split proxy logs by destination hosts. Then, we converted each of them into one sequence.
  2. We reduced each sequence to enable the learned model to be tolerant of noise.
  3. We converted each line into a feature vector that has 84 dimensions. They are the sub-set of that we used in our DNN model.

Sequencing Proxy Logs (2)

1. We split proxy logs by destination hosts. Then, we converted each of them into one sequence.
   - In this case, we show only URLs and content-types to focus the process.

```
http://blogs.example.com/ text/html
http://blogs.example.com/themes/style.css text/css
http://blogs.example.com/themes/fonts.css text/css
http://www.example.com/js/TEMP.js application/javascript
http://www.example.com/js/home.js application/javascript
http://blogs.example.com/images/nav-bg.png image/png
http://blogs.example.com/js/wp.js application/javascript
https://ad.example.net/ads/ga-audiences? text/html
http://www.example.org/analytics.js application/javascript
http://blogs.example.com/images/nav-act.png image/png
http://blogs.example.com/images/rss-icon.jpg image/jpeg
http://blogs.example.com/js/min.js application/javascript
```

```
http://blogs.example.com/ text/html
http://blogs.example.com/themes/style.css text/css
http://blogs.example.com/themes/fonts.css text/css
http://blogs.example.com/images/nav-bg.png image/png
http://blogs.example.com/js/wp.js application/javascript
http://blogs.example.com/images/nav-act.png image/png
http://blogs.example.com/images/rss-icon.jpg image/jpeg
http://blogs.example.com/js/min.js application/javascript
```

```
http://www.example.com/js/TEMP.js application/javascript
http://www.example.com/js/home.js application/javascript
```

```
https://www.example.org/ads/ga-audiences? text/html
```

```
http://ad.example.net/analytics.js application/javascript
```

Sequencing Proxy Logs (3)

2. We reduced each sequence to avoid noises (1).
   - When lines containing the same content-type are continuous, we omitted the second line and all of the following lines containing the same content-type.

```
http://blogs.example.com/ text/html
http://blogs.example.com/themes/style.css text/css
http://blogs.example.com/themes/fonts.css text/css
http://blogs.example.com/images/nav-bg.png image/png
http://blogs.example.com/js/wp.js application/javascript
http://blogs.example.com/images/nav-act.png image/png
http://blogs.example.com/images/rss-icon.jpg image/jpeg
http://blogs.example.com/js/min.js application/javascript
```

```
http://blogs.example.com/ text/html
http://blogs.example.com/themes/style.css text/css
~~http://blogs.example.com/themes/fonts.css text/css~~
http://blogs.example.com/images/nav-bg.png image/png
http://blogs.example.com/js/wp.js application/javascript
http://blogs.example.com/images/nav-act.png image/png
http://blogs.example.com/images/rss-icon.jpg image/jpeg
http://blogs.example.com/js/min.js application/javascript
```

Sequencing Proxy Logs (4)

2. We reduced each sequence to avoid noises (2).
   - When the length of each sequence is longer than five lines, we omitted the sixth and any future lines. It's because the length of sequence was set at five.

```
http://blogs.example.com/ text/html
http://blogs.example.com/themes/style.css text/css
http://blogs.example.com/images/nav-bg.png image/png
http://blogs.example.com/js/wp.js application/javascript
http://blogs.example.com/images/nav-act.png image/png
http://blogs.example.com/images/rss-icon.jpg image/jpeg
http://blogs.example.com/js/min.js application/javascript
```

http://blogs.example.com/ text/html
http://blogs.example.com/themes/style.css text/css
http://blogs.example.com/images/nav-bg.png image/png
http://blogs.example.com/js/wp.js application/javascript
http://blogs.example.com/images/nav-act.png image/png
~~http://blogs.example.com/images/rss-icon.jpg image/jpeg~~
~~http://blogs.example.com/js/min.js application/javascript~~

Sequencing Proxy Logs (5)

3. We converted each line into a feature vector that has 84 dimensions. They are the sub-set of that we used in our DNN model.
   - The first 83 dimensions express a content-type that was converted with one-hot-encoding.
   - The remaining dimension is a flag whether a URL and a referer contain the same domain or not.

- content_type[83] (83 types)
- Do Referer and URL contain the same domain?

Our RNN Model

- We built LSTM (Long Short-Term Memory) models with Keras.
  - We also tested simple RNN models and GRU (Gated Recurrent Unit) models. We found that LSTM models worked better than others.
  - We used a grid search method to determine parameters, and the following values were the best in our environment.

```
model = Sequential()
model.add(LSTM(100, input_shape=(5, 84),return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(30))
model.add(Dropout(0.2))
model.add(Dense(1))
model.add(Activation("sigmoid"))
model.compile('nadam', 'binary_crossentropy', metrics=["accuracy"])
model.fit(trainX, trainY, epochs=20, batch_size=1000, shuffle=True, class_weight = {0:1., 1:100.})
```

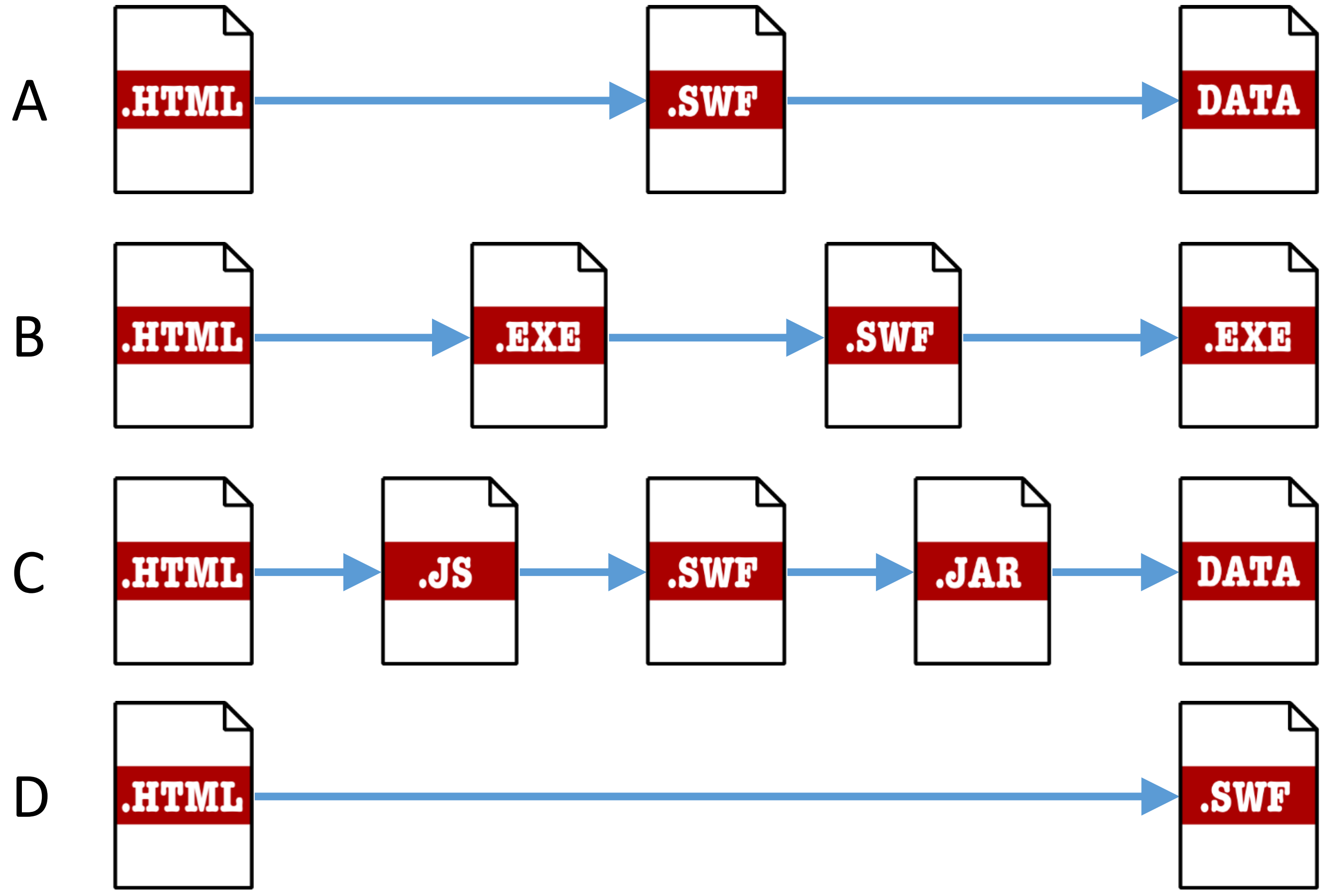
Training Datasets (1)

1. We gathered 578,035 benign sequences from 3,944,019 lines of our proxy logs.
2. We generated 303,156 exploit kit like sequences that have the following characteristics. We didn't collect sequences from actual EK traffic.
   - The length of each sequence is greater than two.
   - The content-type of the first line is "text/html".
   - The content-type of the second line is one of the following other than "text/html".
   - The content-type of the third line and later are one of the following.

- text/html
- text/plain
- text/xml
- application/javascript
- application/pdf
- application/x-shockwave-flash
- application/x-silverlight-app
- application/java-archive
- application/x-java-jnlp-file
- application/octet-stream
- application/x-msdownload

Training Datasets (2)

- Examples of generated content-type sequences



A typical sequence

A case that exploits succeeded twice

- One is included in a landing page and another is a typical flash exploit.

A case that multiple exploits are loaded

A case that an exploit failed

Test Datasets and Results (1)

- We tested the model with malicious sequences that consisted of actual exploit kits' traffic, the model could detect all of the following 14 exploit kits.
  - **Rig, Nebula, Terror, Sundown, KaiXin, Neutrino, Angler, Nuclear, Magnitude, Fiesta, SweetOrange, Goon, Infinity, Astrum**
    - We collected some of these exploit kits' traffic data from malware-traffic-analysis.net.

Test Datasets and Results (2)

- Example sequences that the model successfully detected.

A. Rig EK sequence

#	Content-type
1	text/html
2	application/x-shockwave-flash
3	application/x-msdownload

B. Nebula EK sequence that an exploit failed

#	Content-type
1	text/html
2	application/x-shockwave-flash

C. Neutrino EK sequence

#	Content-type
1	text/html
2	application/x-shockwave-flash
3	text/html
4	application/octet-stream

D. KaiXin EK sequence that contains multiple exploit contents, and a landing page was loaded twice

#	Content-type
1	text/html
2	application/javascript
3	application/x-shockwave-flash
4	text/html

Test Datasets and Results (3)

- We also tested the model with benign sequences that consist of our proxy logs. Each of them were gathered from about 4 million lines of proxy logs that we collected in May 2017.

Dataset	Num. of Sequences	False positives	Result (Accuracy)
Benign Dataset A	562,390	642	0.9988
Benign Dataset B	574,452	681	0.9988
Benign Dataset C	576,294	639	0.9988

To Reduce False Positives

- By applying a simple whitelist that contains only 15 domains, we reduced the number of false positives into half.
- We can also reduce false positives with applying the following methods.
  - Host reputation, anomaly analysis, automated sandboxes and manual analysis.

Conclusion of This Section

- We can detect unlearned EKs in proxy logs with our LSTM model. It is likely to detect unknown brand-new EKs.
  - Though we might need more length and features to detect complex EKs' attacks in the near future, LSTM models will be able to cover them.

- We can detect learned EKs in proxy logs with our DNN model. It is effective to trace variants of known EKs.

Closing Remarks

Summary

- Our simple CNN model can detect malicious C2 communications with sufficiently high performance.
  - The result of benign samples is high accuracy (99.95%) as well as low false positives (0.03%).
  - Against in-the-wild malware, our model can detect all eleven malware families that we tested.
- Our LSTM model can detect unlearned exploit kits' traffic.
  - The model could detect actual traffic of 14 in-the-wild exploit kits.
  - For benign traffics, the model resulted in acceptable accuracy (99.88%).

Black Hat Sound Bytes

- First, we provided practical ways to detect malicious activities. Each method we described works well since there are only small numbers of false positives and high accuracy.

- Second, we gave you several new techniques for utilizing logs of ordinary devices. The techniques can apply to common devices such as proxies, firewalls, routers so that everyone can detect malicious activities.

- Third, we disclosed all parameters to detect malicious activities and attendees are able to reproduce them.