Deep-Learning for Log Analysis to Detect Malicious Communications

2.1 Introduction

Deep learning can be used to discover malicious communications. Here, we describe two methods of detecting malware communications caused by malware and by exploit kits in the huge volumes of logs generated by commonly used devices such as firewalls and Web proxy servers.

This chapter is a retelling of a presentation titled "Deep Impact: Recognizing Unknown Malicious Activities from Zero Knowledge"[*1] given as part of the Briefings sessions at the Black Hat Europe 2018 international security conference.

2.2 Background

In most cases at present, the following methods are used to detect malicious activities, including malware infections.

- Pattern matching (including blacklists and whitelists)
- Behavioral analysis
- Event correlation

However, sophisticated and unknown attacks can circumvent these solutions. And even with attacks that are not particularly sophisticated or unknown, detection rules for pattern matching, for instance, need to be changed in response to even small changes in an attack's pattern. This is because the existing detection methods are based on information that an attacker can easily alter, such as the C2 server domain name, IP address, and the executable's binary pattern. Hence, if it is possible to use detection criteria that do not rely on these existing methods and that apply to essential aspects of an attack that are difficult for attackers to alter, we can combine such criteria with existing methods to achieve an ever greater level of security.

Some of the solutions described earlier are also very expensive and thus not necessarily something that all organizations can deploy. The aim of our work, therefore, was to develop a general-purpose solution that would enable many organizations to detect threats based on the logs created by common types of servers and network devices, such as Web proxy servers, routers, and firewalls, rather than specialized equipment and security devices. These logs have rarely been used effectively in the past, with their use being limited to cases such as the following.

- Anomaly detection based on communication volume, frequency, etc.
- SIEM event correlation
- Detection using IoCs (indicators of compromise) when they are available[*2]

If we can make use of these sorts of logs, which consume valuable disk space, many organizations may be able to achieve greater levels of security without making large changes to network structure or additional investments.

One possible reason why these sorts of logs have not been put to effective use is that, although somewhat dependent on system and organizational scale, the logs themselves are very large, preventing effective analysis that involves high computational complexity. Deep learning, however, is known to be suitable for big data analysis; for example, it is capable of processing hundreds of millions of images each on the order of tens to hundreds of kilobytes in size. So if such logs can be optimized for deep learning, it may be possible to solve this problem.

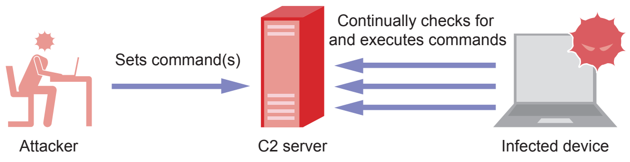


Figure 1: Bot or RAT Continually Communicating with a Command & Control (C2) Server

*1 Deep Impact: Recognizing Unknown Malicious Activities from Zero Knowledge (https://www.blackhat.com/eu-18/briefings/schedule/index.html#deep-impact-recognizing-unknown-malicious-activities-from-zero-knowledge-12276).

*2 For example, IoCs may be obtained from host names discovered via anomaly detection or analysis of suspicious devices reported by users, as well as from external reports, etc.

2.3 Detecting Communications with Malware C&C (C2) Servers

Some types of malware such as bots and RATs continually connect to their C2 servers to receive commands from attackers, which they then execute (Figure 1). Typically, they use polling intervals that range from several dozen seconds to several minutes or so. The longer the interval, the longer the program waits for any single command, which makes it difficult for the attacker to take action. Conversely, the shorter the interval, the easier it is for the attacker to act, but the easier defenders can detect the activity, since the activity will appear toward the top in a simple analysis of communication frequency broken down by destination hosts[*3]. What this boils down to is that adjusting the frequency of communication presents both an important task and a tough decision for the attacker. Meanwhile, when ordinary users within an organization communicate with external networks, such as when accessing the Web, it is rare for those communications to occur frequently or persist over a long time. Figure 2 illustrates what communication frequency looks like over the course of an hour when a user accesses harmless Web servers (left) and when malware is continually communicating with a C2 server (right). Different communication patterns almost always arise, so we thought that if our system could learn the differences, it would be able to detect malware communications. This method does not rely on DNS name, IP address, URL, and the like, so it should detect malware that existing detection methods miss.

Here, we divide the logs up by client and server and count communications in 1-minute buckets for each 1-hour period. We thus convert the logs into pseudo 60-dot images on which we perform image recognition using CNNs (convolution neural networks), a class of neural network used in deep learning. It is known that, depending on the model used, CNNs can outperform human recognition accuracy, and we therefore transform the logs into images and use this class of network in the hopes that this will prove more effective than other deep learning models.

Our training dataset is constructed as follows. We use 1.5 million "images" created from Web proxy logs as our benign sample set. For our malicious samples, we use no actual (in-the-wild) malware communication patterns and instead emulate patterns of periodic communications. With this approach, simply by generating a range of conceivable malicious patterns and training our model on them, we should be able to detect malware even for which no real-world samples are available. This is what we mean by the term "zero

	0	1	2	3	4	5	6	7	8	9	(min.)
0	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	

(1) To a legitimate Web server

	0	1	2	3	4	5	6	7	8	9	(min.)
0	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	

(2) To a C2 server

Figure 2: Illustration of Communication with a Legitimate Web Server (left) and Continual Communication with a C2 Server (right)

[*3] Some recent malware samplesreceive sleep times from a C2 server. Only when the attacker is active, sleep times are short and communication is frequent. At other times, the programs sleep for long periods. These sorts of techniques can make it difficult to detect the anomalous communication based on, for example, daily average times.

knowledge" in the subtitle of our Black Hat presentation. We emulate patterns for a wide range of intervals, from 3 seconds up to 12 minutes (Figure 3). As a special case, we also generate patterns that include sleeps of several minutes following several minutes of continuous activity (Figure 4). Additionally, to account for patterns that differ only slightly from the ones we have come up with and to better resist CNN attacks[*4], we also (a) apply rotations that shift each dot in the generated patterns along a number of intervals and (b) randomly set existing values to zero. We thus generate a total of around one million patterns.

Moving on to our test dataset, we use around 4.5 million images constructed from Web proxy logs (in the same manner as for our training dataset) as our benign samples. For our malicious samples, we use images constructed from logs of malware communications taken from in-the-wild incidents to see if our system can detect these. Our investigation covers the following malware families[*5].

- PlugX
- Asruex
- xxmm
- himawari/ReadLeaves
- ChChes
- Elirks
- Logedrut
- ursnif/gozi
- Shiz/Shifu
- Vawtrak
- KINS

Using the model we built, we were able to detect all of these malware families. And as shown below, the rates of false

	0	1	2	3	4	5	6	7	8	9	(min.)
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	

Emulation of communication once every 3 seconds

...

	0	1	2	3	4	5	6	7	8	9	(min.)
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	

Emulation of communication every 12 minutes

Figure 3: Emulation of Malicious Communications

	0	1	2	3	4	5	6	7	8	9	(min.)
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. of continual communication followed by 2-min. sleep

	0	1	2	3	4	5	6	7	8	9	(min.)
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	0	0	0	0	1	1	1	0	0	0	

...

3 min. of continual communication followed by 3-min. sleep

Figure 4: Emulation of Malicious Communications (2)

*4 Simple Black-Box Adversarial Perturbations for Deep Networks (https://arxiv.org/abs/1612.06299).

*5 We do not obtain patterns simply by acquiring and running malware samples in a closed environment; instead, we use patterns obtained from actual incidents in which malware was connected to a C2 server. This is because the sleep times observed when malware is connected to an in-the-wild C2 server may differ from those observed when it is simply run in a closed environment (see footnote *3).

positives for our benign samples were also low. So we believe our system will be effective if we filter out these FQDNs using a whitelist.

- Benign sample set 1
  Accuracy: 1,565,139/1,566,109 (99.94%)
  False positive FQDNs: 64/246,190
- Benign sample set 2
  Accuracy: 1,540,419/1,541,050 (99.96%)
  False positive FQDNs: 72/243,106
- Benign sample set 3
  Accuracy: 1,528,936/1,529,617 (99.96%)
  False positive FQDNs: 65/243,185

That said, it is conceivable that Web pages that frequently reload, such as Web mail interfaces and sports sites, could register as false positives. So to reduce false positives when the system is in operation, we can take steps like excluding such sites by whitelisting them or regarding communication with a Web server as legitimate when alerts from many users are raised for that same destination.

Figures 5–9 show examples of communication patterns successfully detected from actual malware communications. It is evident that communications from the actual samples are not perfectly periodic, but the system takes the differences in stride and detects the patterns using deep learning. Logedrut

	0	1	2	3	4	5	6	7	8	9	(min.)
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	

	0	1	2	3	4	5	6	7	8	9	(min.)
0	96	95	92	96	95	97	96	97	101	95	
10	97	93	95	96	95	98	92	93	95	101	
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	

Figure 5: PlugX Communication Pattern

	0	1	2	3	4	5	6	7	8	9	(min.)
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	

Figure 6: Asruex Communication Pattern

	0	1	2	3	4	5	6	7	8	9	(min.)
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	

Figure 7: Elirks Communication Pattern

	0	1	2	3	4	5	6	7	8	9	(min.)
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	

Figure 8: Logedrug Communication Pattern

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

Figure 9: Vawtrak Communication Pattern

(Figure 8) communicates infrequently, only once every 12 minutes, but it was still detected as being distinct from the benign sample. In the case of Vawtrak (Figure 9), no communications occur in the final 16 minutes, but the system detects the malware in cases like these as well.

For space reasons, we do not show all malware families and model details here. Further details can be found in our slides[*1] on the Black Hat Europe 2018 website. Since we are not building complicated models here, we believe the models can be trained on CPU-based systems.

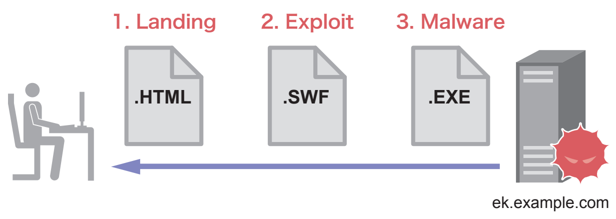
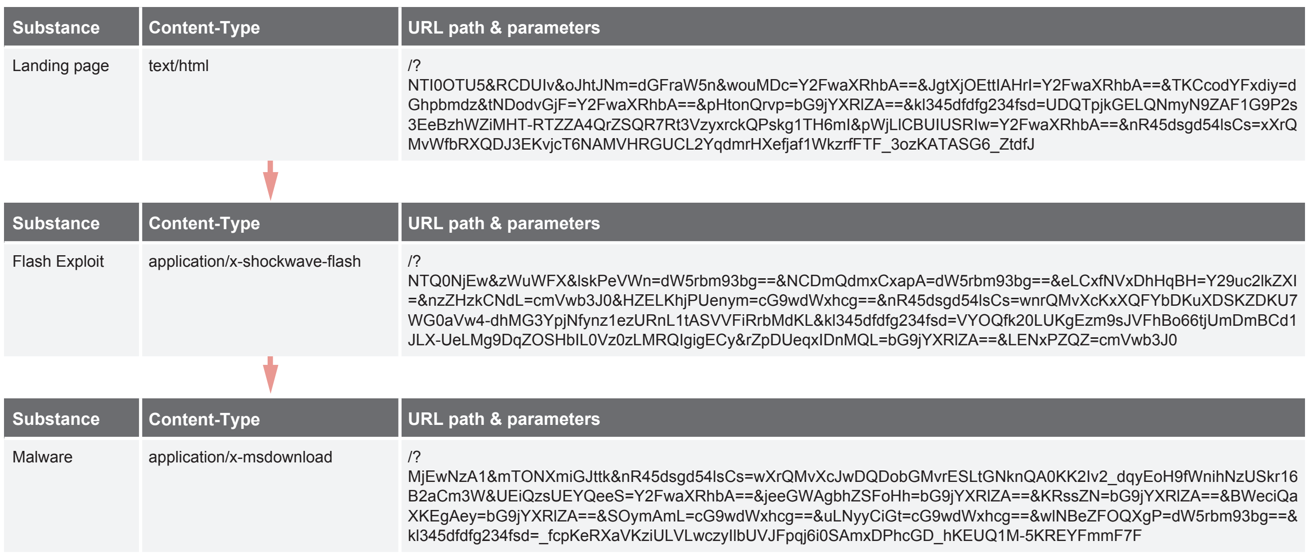


Figure 10: Sequence of Content-Types Sent by an Exploit Kit Server

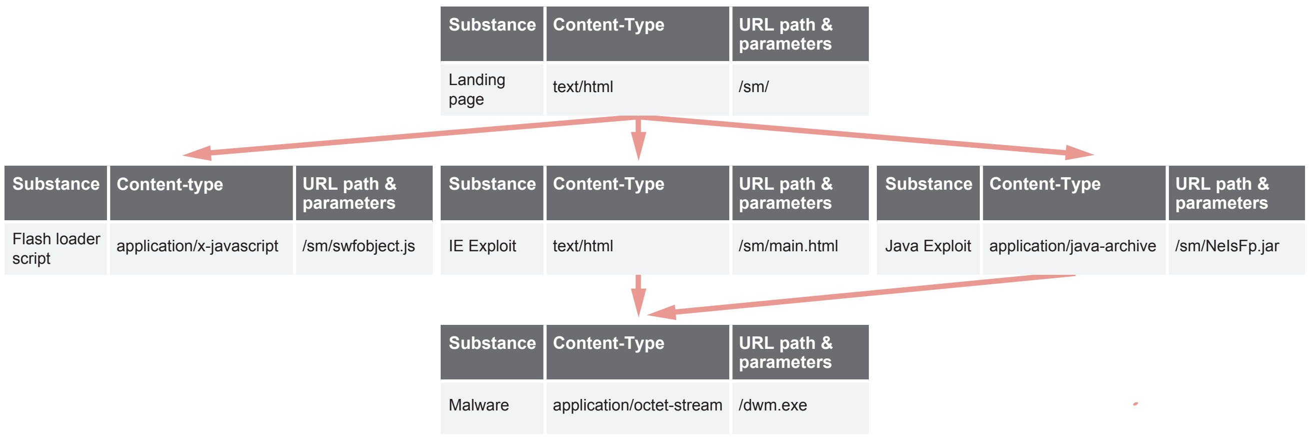


Substance	Content-Type	URL path & parameters
Landing page	text/html	/?NTI0OTU5&RCDUIv&oJhtJNm=dGFraW5n&wouMDc=Y2FwaXRhbA==&JgtXjOEttlAHrl=Y2FwaXRhbA==&TKCcodYFxdiy=dGhpbmdz&tNDodvGjF=Y2FwaXRhbA==&pHtonQrvp=bG9jYXRlZA==&kl345dfdfg234fsd=UDQTpjkGELQNmyN9ZAF1G9P2s3EeBzhWZiMHT-RTZZA4QrZSQR7Rt3VzyxrckQPskg1TH6ml&pWjLICBUIUSRIw=Y2FwaXRhbA==&nR45dsgd54lsCs=xXrQMvWfbRXQDJ3EKvjcT6NAMVHRGUCL2YqdmrHXefjaf1WkzrfFTF_3ozKATASG6_ZtdfJ

Substance	Content-Type	URL path & parameters
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	URL path & parameters
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

Figure 11: Example of Rig Exploit Kit's Content-Type Sequence



Substance	Content-Type	URL path & parameters
Landing page	text/html	/sm/

Substance	Content-type	URL path & parameters
Flash loader script	application/x-javascript	/sm/swfobject.js

Substance	Content-Type	URL path & parameters
IE Exploit	text/html	/sm/main.html

Substance	Content-Type	URL path & parameters
Java Exploit	application/java-archive	/sm/NelsFp.jar

Substance	Content-Type	URL path & parameters
Malware	application/octet-stream	/dwm.exe

Figure 12: Example of KaiXin Exploit Kit's Content-Type Sequence

2.4 Exploit Kit Detection

When a PC accessing the Web is redirected to an exploit kit, the exploit kit server sends content in the order shown in Figure 10.

1. Landing page: Identifies the PC's Web browser environment and loads the next stage of exploit content. May also include an exploit(s) for the browser itself. The Content-Type is text/html.
2. Exploit content: Content file containing exploit(s) for the browser and its plug-ins. Content-Types include application/x-shockwave-flash, application/x-java-archive, application/x-silverlight-app, application/pdf.
3. Malware: If the previous stage's exploit(s) succeeds, malware that infects the PC itself is loaded. In most cases, the Content-Type is application/octet-stream or application/x-msdownload.

Figures 11 and 12 show examples of the Content-Type sequences when, respectively, Rig Exploit Kit and KaiXin Exploit Kit are observed. The figures show that their Content-Type transitions are indeed as described above.

On the other hand, we can think of almost no cases in which normal Web browsing would produce these sorts of sequences in Content-Type sent by a server. In large-scale Web services, for example, dedicated servers are typically set up to handle each Content-Type, so content that is subject to exploits, such as Flash and Java, and HTML content like landing pages tend to come from different servers, as shown in Figure 13. And in cases where a single server hosts all of a service's content, data like images and CSS, which recent exploit kits do not use all that much, come from the same server, as shown in Figure 14.

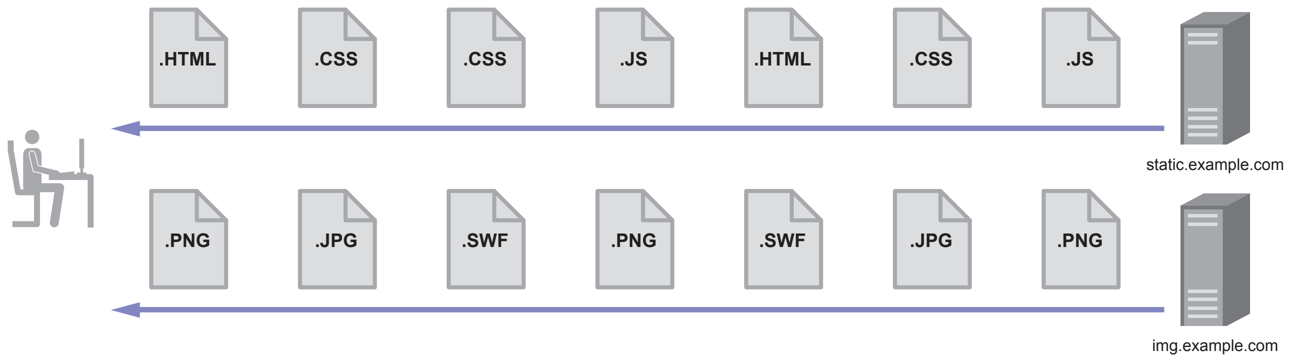


Figure 13: Example Content-Type Sequences for a Web Service with Separate Servers for Content-Types

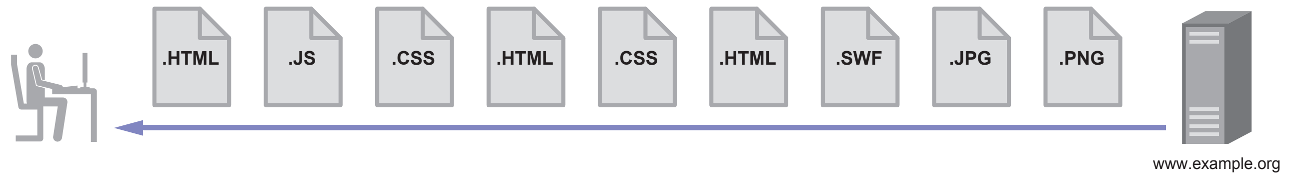


Figure 14: Example Content-Type Sequence for a Service Hosted on a Single Web Server

Given the above, we thought that if analysis of Web proxy logs could differentiate between the sequences for exploit kit servers and the sequences for normal Web server connections, it might be possible to detect exploit kits without relying on techniques like pattern matching. The Content-Type sequences peculiar to the exploit kits mentioned above are strongly related to the fundamental functions by which exploit kits force a Web browser to run exploits and infect a PC with malware. We should, therefore, be able to detect unknown exploit kits that operate in a similar manner. For the same reason, it should not be easy for an exploit kit author to evade detection by altering the sequence.

To differentiate between sequences, we use a class of neural networks called RNNs (recurrent neural network), which are used in natural language processing and the processing of time-series data such as video and audio streams. So first, we need to convert the Web proxy logs into a form that an RNN model can process. We split the logs into individual client PC – destination server pairs and convert series of requests and responses into sequences[*6]. To improve noise tolerance, we eliminate duplicate Content-Types from within each sequence. We also limit sequences to a length of five[*7], deleting any lines beyond that. Finally, we convert each line in the sequence to an 84-dimension vector. The first 83 slots in the vector represent the Content-Type in one-hot encoding, and the final slot is a flag indicating whether the referer and request URL contain the same domain name.

Our training dataset comprises a benign sample of around 580,000 sequences constructed from some 3.9 million lines from Web proxy logs, and a malicious sample of around 300,000 sequences designed to emulate conceivable exploit kit patterns. Instead of using patterns observed in the wild for our malicious sample, we generate a comprehensive range of patterns representing content sequences that could conceivably be produced by exploit kits. Figure 15 shows examples of such pseudo-sequences. Our sample includes

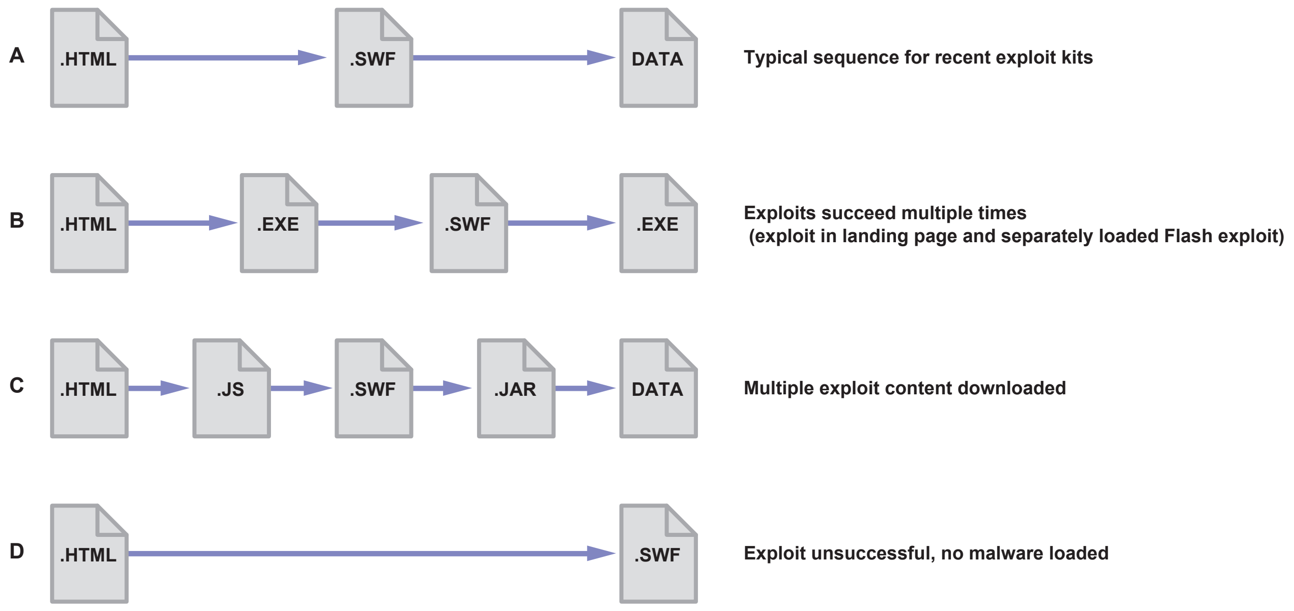

A
.HTML
.SWF
DATA
Typical sequence for recent exploit kits
B
.HTML
.EXE
.SWF
.EXE
Exploits succeed multiple times
(exploit in landing page and separately loaded Flash exploit)
C
.HTML
.JS
.SWF
.JAR
DATA
Multiple exploit content downloaded
D
.HTML
.SWF
Exploit unsuccessful, no malware loaded

Figure 15: Examples of Generated Pseudo-sequences for Exploit Kits

*6 To be precise, we split log lines up according to client PC – destination server pairs, and then further split them into Web browser sessions (being the series of requests and responses caused by a Web browser in order to display the Web page at a given URL). The Web proxy logs we used have each Web browser session recorded separately. In more common environments, it is possible to split up Web proxy logs according to fields such as the timestamp.

*7 With the exploit kits we have observed recently, in almost no cases is content sent by the server more than five times. However, if such cases were to rise in future, we think the upper limit would need to be raised.

sequences that represent cases where several types of exploit content are loaded, cases where an exploit succeeds multiple times in a row, and cases where the exploit is unsuccessful and no malware download takes place.

To test our model, we used a malicious sample constructed from actual communication data for the following 14 exploit kits, and a benign sample of around 1.7 million sequences constructed from Web proxy logs for a time period that differs from that of the training set.

- Rig
- Nebula
- Terror
- Sundown
- KaiXin
- Neutrino
- Angler
- Nuclear
- Magnitude
- Fiesta
- Sweet Orange
- Goon
- Infinity
- Astrum

The model we built was able to detect all of the exploit kits listed above. And as shown below, false positive rates for our benign samples were also relatively low.

- Benign sample set 1
  Sequences: 562,390
  False positives: 642
  Accuracy: 0.9988
- Benign sample set 2
  Sequences: 574,452
  False positives: 681
  Accuracy: 0.9988
- Benign sample set 3
  Sequences: 576,294
  False positives: 639
  Accuracy: 0.9988

We have confirmed that using a whitelist of around 15 lines can halve the number of false positives listed above. When applying a system like ours to production environments, we would recommend combining it with other methods to narrow down the alerts, such as whitelists, host reputation, anomaly detection, and automated sandbox analysis.

Our slides[*1] available on the Black Hat Europe 2018 website also present a method of identifying Rig Exploit Kit from Web proxy logs using an MLP (multilayer perceptron) model. This method focuses on features of individual exploit kits' URLs, so while it is not suited to detecting unknown exploit kits, it is useful for identifying known exploit kits and tracking their variants. Using it with the RNN-based exploit kit detection method described here can improve the accuracy of detection of known exploit kits.

Hiroshi Suzuki
Malware & Forensic Analyst, Office of Emergency Response and Clearinghouse for Security Information, Advanced Security Division, IIJ
As a member of IIJ-SECT, Mr. Suzuki is a malware analyst and a forensic investigator. He has dedicated over 13 years to the areas. As a frequent speaker and trainer for international conferences, he has given presentations and trainings at Black Hat (USA, Europe and Asia) and FIRST TC multiple times.

Hisao Nashiwa
Threat Analyst, Office of Emergency Response and Clearinghouse for Security Information, Advanced Security Division, IIJ
Mr. Nashiwa is a member of IIJ-SECT, which is IIJ's private CSIRT.
His work includes incident response, malware analysis and network traffic analysis, and he has thus been investigating malicious activities over nine years. He has been researching cyber crimes such as those involving exploit kits and malware for many years and has expertise in malware analysis. He is a frequent conference speaker and has given talks and hands-on training sessions multiple times at international conferences such as Black Hat and FIRST TC.