

# 鷹匠 Takajō

[ English ] | [ 日本語 ]

Stable Version v2.3.0

GitHub Downloads 322

GitHub Stars 34

CODE BLUE Bluebox 2022

SECCON 2023

SANS DFIR Summit 2023

Maintenance Level **Actively Developed**

 Follow @SecurityYamato

## About Takajō

Takajō (鷹匠), created by [Yamato Security](#), is a fast forensics analyzer for [Hayabusa](#) results written in [Nim](#). Takajō means "[Falconer](#)" in Japanese and was chosen as Hayabusa's catches (results) can be put to even better use.

## Companion Projects

- [EnableWindowsLogSettings](#) - documentation and scripts to properly enable Windows event logs.
- [Hayabusa](#) - sigma-based threat hunting and fast forensics timeline generator for Windows event logs.
- [Hayabusa Rules](#) - detection rules for hayabusa.
- [Hayabusa Sample EVTxs](#) - sample evtX files to use for testing hayabusa/sigma detection rules.
- [WELA \(Windows Event Log Analyzer\)](#) - analyzer for Windows event logs written in PowerShell.

## Table of Contents

- [Companion Projects](#)
  - [Table of Contents](#)
  - [Features](#)
- [Downloads](#)
  - [Git cloning](#)
  - [Advanced: Compiling From Source \(Optional\)](#)
- [Command List](#)
  - [Extract Commands](#)

- [List Commands](#)
- [Split Commands](#)
- [Stack Commands](#)
- [Sysmon Commands](#)
- [Timeline Commands](#)
- [TTP Commands](#)
- [VirusTotal Commands](#)
- [Command Usage](#)
  - [Extract Commands](#)
    - [extract-scriptblocks](#) [command](#)
      - [extract-scriptblocks](#) [command example](#)
      - [extract-scriptblocks](#) [screenshot](#)
  - [List Commands](#)
    - [list-domains](#) [command](#)
      - [list-domains](#) [command examples](#)
    - [list-hashes](#) [command](#)
      - [list-hashes](#) [command examples](#)
    - [list-ip-addresses](#) [command](#)
      - [list-ip-addresses](#) [command examples](#)
    - [list-undetected-evt](#) [command](#)
      - [list-undetected-evt](#) [command examples](#)
    - [list-unused-rules](#) [command](#)
      - [list-unused-rules](#) [command examples](#)
  - [Split Commands](#)
    - [split-csv-timeline](#) [command](#)
      - [split-csv-timeline](#) [command examples](#)
    - [split-json-timeline](#) [command](#)
      - [split-json-timeline](#) [command examples](#)
  - [Stack Commands](#)
    - [stack-logons](#) [command](#)
      - [stack-logons](#) [command examples](#)
  - [Sysmon Commands](#)
    - [sysmon-process-tree](#) [command](#)
      - [sysmon-process-tree](#) [command examples](#)
      - [sysmon-process-tree](#) [screenshot](#)
  - [Timeline Commands](#)
    - [timeline-logon](#) [command](#)
      - [timeline-logon](#) [command examples](#)
      - [timeline-logon](#) [screenshot](#)
    - [timeline-partition-diagnostic](#) [command](#)
      - [timeline-partition-diagnostic](#) [command examples](#)
    - [timeline-suspicious-processes](#) [command](#)
      - [timeline-suspicious-processes](#) [command examples](#)

- [timeline-suspicious-processes\\_screenshot](#)
- [TTP Commands](#)
  - [ttp-summary\\_command](#)
    - [ttp-summary\\_command\\_examples](#)
    - [ttp-summary\\_screenshot](#)
  - [ttp-visualize\\_command](#)
    - [ttp-visualize\\_command\\_examples](#)
    - [ttp-visualize\\_screenshot](#)
- [VirusTotal Commands](#)
  - [vt-domain-lookup\\_command](#)
    - [vt-domain-lookup\\_command\\_examples](#)
  - [vt-hash-lookup\\_command](#)
    - [vt-hash-lookup\\_command\\_examples](#)
  - [vt-ip-lookup\\_command](#)
    - [vt-ip-lookup\\_command\\_examples](#)
- [Contribution](#)
- [Bug Submission](#)
- [License](#)
- [Twitter](#)

## Features

- Written in Nim so it is very easy to program, memory safe, as fast as native C code and works as a single standalone binary on any OS.
- Create a timeline of all of the various logon events, suspicious processes, etc...
- Print the process trees of a malicious processes.
- Various stacking analysis.
- Split up CSV and JSONL timelines.
- List up IP addresses, domains, hashes etc... to be used with VirusTotal lookups
- VirusTotal lookups of domains, hashes and IP addresses.
- List up `.evtx` files that cannot be detected yet.
- Visualize TTPs in MITRE ATT&CK Navigator.
- Many more!

## Downloads

Please download the latest stable version of Takajo with compiled binaries or compile the source code from the [Releases](#) page.

## Git cloning

You can git clone the repository with the following command and compile binary from source code:

*Warning: The main branch of the repository is for development purposes so you may be able to access new features not yet officially released, however, there may be bugs so consider it unstable.*

```
git clone https://github.com/Yamato-Security/takajo.git
```

## Advanced: Compiling From Source (Optional)

If you have Nim installed, you can compile from source with the following command:

```
> nimble update
> nimble build -d:release --threads:on
```

## Command List

### Extract Commands

- `extract-scriptblocks` : extract and reassemble PowerShell EID 4104 script block logs

### List Commands

- `list-domains` : create a list of unique domains to be used with `vt-domain-lookup`
- `list-hashes` : create a list of process hashes to be used with `vt-hash-lookup`
- `list-ip-addresses` : create a list of unique target and/or source IP addresses to be used with `vt-ip-lookup`
- `list-undetected-evtx` : create a list of undetected evtx files
- `list-unused-rules` : create a list of unused detection rules

### Split Commands

- `split-csv-timeline` : split up a large CSV timeline into smaller ones based on the computer name
- `split-json-timeline` : split up a large JSONL timeline into smaller ones based on the computer name

### Stack Commands

- `stack-logons` : stack logons by target user, target computer, source IP address and source computer

### Sysmon Commands

- `sysmon-process-tree` : output the process tree of a certain process

### Timeline Commands

- `timeline-logon` : create a CSV timeline of logon events
- `timeline-partition-diagnostic` : create a CSV timeline of partition diagnostic events
- `timeline-suspicious-processes` : create a CSV timeline of suspicious processes

### TTP Commands

- `ttp-summary` : summarize tactics and techniques found in each computer
- `ttp-visualize` : extract TTPs and create a JSON file to visualize in MITRE ATT&CK Navigator

### VirusTotal Commands

- `vt-domain-lookup` : look up a list of domains on VirusTotal and report on malicious ones

- `vt-hash-lookup` : look up a list of hashes on VirusTotal and report on malicious ones
- `vt-ip-lookup` : look up a list of IP addresses on VirusTotal and report on malicious ones

## Command Usage

### Extract Commands

#### `extract-scriptblocks` command

Extracts and reassembles PowerShell EID 4104 script block logs.

*Note: The PowerShell scripts are best opened as `.ps1` files with code syntax highlighting but we use the `.txt` extension in order to prevent any accidental running of malicious code.*

- Input: JSONL
- Profile: Any
- Output: Terminal and directory of PowerShell Scripts

Required options:

- `-t, --timeline <JSONL-FILE>` : Hayabusa JSONL timeline

Options:

- `-l, --level` : specify the minimum alert level (default: `low` )
- `-o, --output` : output directory (default: `scriptblock-logs` )
- `-q, --quiet` : do not display the launch banner (default: `false` )

#### `extract-scriptblocks` command example

Prepare the JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Extract PowerShell EID 4104 script block logs to the `scriptblock-logs` directory:

```
takajo.exe extract-scriptblocks -t ../hayabusa/timeline.jsonl
```

#### `extract-scriptblocks` screenshot

Creation Time	Computer Name	Script ID	Script Name	Records	Level	Alerts
2019-09-09 22:35:08.655 +09:00	MSEdgeWIN10	37f6d110-cfdf-4118-8748-17638e258531	no-path	1/1	med	"Suspicious FromBase64String Usage On Gzip Archive - Ps Script", "Potentially Malicious PwSh"
2019-09-09 22:35:09.315 +09:00	MSEdgeWIN10	c7ca7056-b317-4fff-b796-05d8ef896dcd	no-path	1/1	high	"Suspicious PowerShell Get Current User", "PowerShell Credential Prompt", "Manipulation of User Computer or Group Security Principals Across AD", "Potentially Malicious PwSh"
2020-06-30 23:24:08.254 +09:00	MSEdgeWIN10	27f08bda-c330-419f-b83b-eb5c0f699930	C:\Users\Public\lsass_wer_ps.ps1	1/1	high	"PowerShell Get-Process LSASS in ScriptBlock", "Malicious PowerShell Keywords", "Suspicious Process Discovery With Get-Process", "WinAPI Function Calls Via PowerShell Scripts", "Use Remove-Item to Delete File", "Potentially Malicious PwSh"
2020-08-26 14:09:28.845 +09:00	DESKTOP-RIPLCLIP	fd51159-9602-40cb-839d-c31039ebbc3a	no-path	1/1	high	"Usage Of Web Request Commands And Cmdlets - ScriptBlock", "Powershell Token Obfuscation - Powershell"

The extracted PowerShell ScriptBlock is saved in the directory: `scriptblock-logs`  
 Saved summary file: `scriptblock-logs/summary.csv` (1.11 KB)

Elapsed time: 0 hours, 0 minutes, 0 seconds

## List Commands

### list-domains command

Creates a list of unique domains to be used with `vt-domain-lookup` . Currently it will only check queried domains in Sysmon EID 22 logs but will be updated to support built-in Windows DNS Client and Server logs.

- Input: JSONL
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: Text file

Required options:

- `-o, --output <TXT-FILE>` : save results to a text file.
- `-t, --timeline <JSONL-FILE>` : Hayabusa JSONL timeline.

Options:

- `-s, --includeSubdomains` : include subdomains. (default: `false` )
- `-w, --includeWorkstations` : include local workstation names. (default: `false` )
- `-q, --quiet` : do not display logo. (default: `false` )

### list-domains command examples

Prepare the JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Save the results to a text file:

```
takajo.exe list-domains -t ../hayabusa/timeline.jsonl -o domains.txt
```

Include subdomains:

```
takajo.exe list-domains -t ../hayabusa/timeline.jsonl -o domains.txt -s
```

### list-hashes command

Create a list of process hashes to be used with `vt-hash-lookup` (input: JSONL, profile: standard)

- Input: JSONL
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: Text file

Required options:

- `-t, --timeline <JSONL-FILE>` : JSONL timeline created by Hayabusa.
- `-o, --output <BASE-NAME>` : specify the base name to save the text results to.

Options:

- `-l, --level` : specify the minimum level. (default: `high` )
- `-q, --quiet` : do not display logo. (default: `false` )

### list-hashes command examples

Prepare JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Save the results to a different text file for each hash type:

```
takajo.exe list-hashes -t ../hayabusa/timeline.jsonl -o case-1
```

For example, if MD5 , SHA1 and IMPHASH hashes are stored in the sysmon logs, then the following files will be created: case-1-MD5-hashes.txt , case-1-SHA1-hashes.txt , case-1-ImportHashes.txt

### list-ip-addresses command

Creates a list of unique target and/or source IP addresses to be used with vt-ip-lookup . It will extract the TgtIP fields for target IP addresses and SrcIP fields for source IP addresses in all results and output just the unique IP addresses to a text file.

- Input: JSONL
- Profile: Any besides all-field-info and all-field-info-verbose
- Output: Text file

Required options:

- -o, --output <TXT-FILE> : save results to a text file.
- -t, --timeline <JSONL-FILE> : Hayabusa JSONL timeline.

Options:

- -i, --inbound : include inbound traffic. (default: true )
- -O, --outbound : include outbound traffic. (default: true )
- -p, --privateIp : include private IP addresses (default: false )
- -q, --quiet : do not display logo. (default: false )

### list-ip-addresses command examples

Prepare the JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Save the results to a text file:

```
takajo.exe list-ip-addresses -t ../hayabusa/timeline.jsonl -o ipAddresses.txt
```

Exclude inbound traffic:

```
takajo.exe list-ip-addresses -t ../hayabusa/timeline.jsonl -o ipAddresses.txt -i=false
```

Include private IP addresses:

```
takajo.exe list-ip-addresses -t ../hayabusa/timeline.jsonl -o ipAddresses.txt -p
```

### list-undetected-evtx command

List up all of the .evtx files that Hayabusa didn't have a detection rule for. This is meant to be used on sample evtx files that all contain evidence of malicious activity such as the sample evtx files in the [hayabusa-sample-evtx](#) repository.

- Input: CSV
- Profile: `verbose` , `all-field-info-verbose` , `super-verbose` , `timesketch-verbose`

*You first need to run Hayabusa with a profile that saves the `%EvtxFile%` column information and save the results to a CSV timeline. You can see which columns Hayabusa saves according to the different profiles [here](#).*

- Output: Terminal or text file

Required options:

- `-e, --evtx-dir <EVTX-DIR>` : The directory of `.evtx` files you scanned with Hayabusa.
- `-t, --timeline <CSV-FILE>` : Hayabusa CSV timeline.

Options:

- `-c, --column-name <CUSTOM-EVTX-COLUMN>` : specify a custom column name for the evtx column. (default: Hayabusa's default of `EvtxFile` )
- `-o, --output <TXT-FILE>` : save the results to a text file. (default: output to screen)
- `-q, --quiet` : do not display logo. (default: `false` )

### **list-undetected-evtx** command examples

Prepare the CSV timeline with Hayabusa:

```
hayabusa.exe -d <EVTX-DIR> -p verbose -o timeline.csv -w
```

Output the results to screen:

```
takajo.exe list-undetected-evtx -t ../hayabusa/timeline.csv -e <EVTX-DIR>
```

Save the results to a text file:

```
takajo.exe list-undetected-evtx -t ../hayabusa/timeline.csv -e <EVTX-DIR> -o undetected-evtx.txt
```

### **list-unused-rules** command

List up all of the `.yaml` detection rules that did not detect anything. This is useful to help determine the reliability of rules. That is, which rules are known to find malicious activity and which are still untested and need sample `.evtx` files.

- Input: CSV
- Profile: `verbose` , `all-field-info-verbose` , `super-verbose` , `timesketch-verbose`

*You first need to run Hayabusa with a profile that saves the `%RuleFile%` column information and save the results to a CSV timeline. You can see which columns Hayabusa saves according to the different profiles [here](#).*

- Output: Terminal or text file

Required options:

- `-r, --rules-dir <DIR>` : the directory of `.yaml` rules files you used with Hayabusa.
- `-t, --timeline <CSV-FILE>` : CSV timeline created by Hayabusa.

Options:

- `-c, --column-name <CUSTOM-RULE-FILE-COLUMN>` : specify a custom column name for the rule file column. (default: Hayabusa's default of `RuleFile` )
- `-o, --output <TXT-FILE>` : save the results to a text file. (default: output to screen)
- `-q, --quiet` : do not display logo. (default: `false` )

### **list-unused-rules** command examples

Prepare the CSV timeline with Hayabusa:

```
hayabusa.exe csv-timeline -d <EVTX-DIR> -p verbose -o timeline.csv -w
```

Output the results to screen:

```
takajo.exe list-unused-rules -t ../hayabusa/timeline.csv -r ../hayabusa/rules
```

Save the results to a text file:

```
takajo.exe list-unused-rules -t ../hayabusa/timeline.csv -r ../hayabusa/rules -o unused-rules.txt
```

## Split Commands

### **split-csv-timeline** command

Split up a large CSV timeline into smaller ones based on the computer name.

- Input: Non-multiline CSV
- Profile: Any
- Output: Multiple CSV files

Required options:

- `-t, --timeline <CSV-FILE>` : CSV timeline created by Hayabusa.

Options:

- `-m, --makeMultiline` : output fields in multiple lines. (default: `false` )
- `-o, --output <DIR>` : directory to save the CSV files to. (default: `output` )
- `-q, --quiet` : do not display logo. (default: `false` )

### **split-csv-timeline** command examples

Prepare the CSV timeline with Hayabusa:

```
hayabusa.exe csv-timeline -d <EVTX-DIR> -o timeline.csv -w
```

Split the single CSV timeline into multiple CSV timelines in the default `output` directory:

```
takajo.exe split-csv-timeline -t ../hayabusa/timeline.csv
```

Separate field information with newline characters to make multi-line entries and save to the `case-1-csv` directory:

```
takajo.exe split-csv-timeline -t ../hayabusa/timeline.csv -m -o case-1-csv
```

## split-json-timeline command

Split up a large JSONL timeline into smaller ones based on the computer name.

- Input: JSONL
- Profile: Any
- Output: Multiple JSONL files

Required options:

- `-t, --timeline <JSONL-FILE>` : Hayabusa JSONL timeline.

Options:

- `-o, --output <DIR>` : directory to save the JSONL files to. (default: `output` )
- `-q, --quiet` : do not display logo. (default: `false` )

## split-json-timeline command examples

Prepare the JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Split the single JSONL timeline into multiple JSONL timelines in the default `output` directory:

```
takajo.exe split-json-timeline -t ../hayabusa/timeline.jsonl
```

Save to the `case-1-jsonl` directory:

```
takajo.exe split-json-timeline -t ../hayabusa/timeline.jsonl -o case-1-jsonl
```

## Stack Commands

### stack-logons command

Creates a list logons according to `Target User` , `Target Computer` , `Logon Type` , `Source IP Address` , `Source Computer` . Results are filtered out when the source IP address is a local IP address by default.

- Input: JSONL
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: Terminal or CSV file

Required options:

- `-t, --timeline <JSONL-FILE>` : JSONL timeline created by Hayabusa.

Options:

- `-l, --localSrcIpAddresses` : include results when the source IP address is local.
- `-o, --output <CSV-FILE>` : specify the base name to save the text results to.
- `-q, --quiet` : do not display logo. (default: `false` )

### stack-logons command examples

Run with default settings:

```
takajo.exe stack-logons -t ../hayabusa/timeline.jsonl
```

Include local logons:

```
takajo.exe stack-logons -t ../hayabusa/timeline.jsonl -l
```

## Sysmon Commands

### sysmon-process-tree command

Output the process tree of a certain process, such as a suspicious or malicious process.

- Input: JSONL
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: Terminal or text file

Required options:

- `-p, --processGuid <Process GUID>` : sysmon process GUID
- `-t, --timeline <JSONL-FILE>` : JSONL timeline created by Hayabusa.

Options:

- `-o, --output <TXT-FILE>` : a text file to save the results to.
- `-q, --quiet` : do not display logo. (default: `false`)

### sysmon-process-tree command examples

Prepare JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Save the results to a text file:

```
takajo.exe sysmon-process-tree -t ../hayabusa/timeline.jsonl -p "365ABB72-3D4A-5CEB-0000-0010FA93FD00" -o process-tree.txt
```

### sysmon-process-tree screenshot

```
Running the Process Tree module
System (N/A / 4 / 747F3D96-B75F-5EA4-0000-0010EB030000 / N/A)
├─ C:\Windows\System32\smss.exe (2020-04-26 07:19:20.134 +09:00 / 300 / 747F3D96-B75F-5EA4-0000-0010622C0000 / 747F3D96-B75F-5EA4-0000-0010EB030000)
├─ C:\Windows\System32\smss.exe (2020-04-26 07:19:22.596 +09:00 / 388 / 747F3D96-B763-5EA4-0000-00106A480000 / 747F3D96-B75F-5EA4-0000-0010622C0000)
├─ C:\Windows\System32\winit.exe (2020-04-26 07:19:23.222 +09:00 / 468 / 747F3D96-B764-5EA4-0000-0010904D0000 / 747F3D96-B763-5EA4-0000-00106A480000)
├─ C:\Windows\System32\services.exe (2020-04-26 07:19:24.049 +09:00 / 584 / 747F3D96-B764-5EA4-0000-00106F550000 / 747F3D96-B764-5EA4-0000-0010904D0000)
├─ C:\Windows\System32\svchost.exe (2020-04-26 07:19:40.692 +09:00 / 6964 / 747F3D96-B776-5EA4-0000-0010A74D0800 / 747F3D96-B764-5EA4-0000-00106F550000)
├─ C:\Windows\System32\consent.exe (2020-04-26 07:20:21.263 +09:00 / 7036 / 747F3D96-B7A5-5EA4-0000-0010EAB91300 / 747F3D96-B776-5EA4-0000-0010A74D0800)
Elapsed time: 0 hours, 0 minutes, 1 seconds
```

## Timeline Commands

### timeline-logon command

This command extracts information from the following logon events, normalizes the fields and saves the results to a CSV file:

- `4624` - Successful Logon

- 4625 - Failed Logon
- 4634 - Account Logoff
- 4647 - User Initiated Logoff
- 4648 - Explicit Logon
- 4672 - Admin Logon

This makes it easier to detect lateral movement, password guessing/spraying, privilege escalation, etc...

- Input: JSONL
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: CSV

Required options:

- `-o, --output <CSV-FILE>` : the CSV file to save the results to.
- `-t, --timeline <JSONL-FILE>` : JSONL timeline created by Hayabusa.

Options:

- `-c, --calculateElapsedTime` : calculate the elapsed time for successful logons. (default: true )
- `-l, --outputLogoffEvents` : output logoff events as separate entries. (default: false )
- `-a, --outputAdminLogonEvents` : output admin logon events as separate entries. (default: false )
- `-q, --quiet` : do not display logon. (default: false )

### timeline-logon command examples

Prepare JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Save logon timeline to a CSV file:

```
takajo.exe timeline-logon -t ../hayabusa/timeline.jsonl -o logon-timeline.csv
```

### timeline-logon screenshot

Timestamp	Ch	Event	Event	LogoffTime	ElapsedTime	FailureReason	TargetComputer	TargetUser	AdminLog	SourceComputer	SourceIP	Type
2016-08-18 23:47:04.676 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-18 23:47:04.676 +09:00	Sec	4624	Successful Logon	2016-08-18 23:47:20.053 +09:00	0d 0h 0m 15s 377ms		IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-18 23:47:36.671 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM	-	-	-	0 - SYSTEM
2016-08-18 23:47:36.671 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM	-	-	-	0 - SYSTEM
2016-08-18 23:47:38.430 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-18 23:47:38.430 +09:00	Sec	4624	Successful Logon	2016-08-18 23:48:31.289 +09:00	0d 0h 0m 52s 859ms		IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-18 23:47:38.430 +09:00	Sec	4624	Successful Logon	2016-08-18 23:48:31.289 +09:00	0d 0h 0m 52s 859ms		IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-18 23:49:38.281 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM	-	-	-	0 - SYSTEM
2016-08-18 23:49:38.281 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM	-	-	-	0 - SYSTEM
2016-08-18 23:49:40.000 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-18 23:49:40.000 +09:00	Sec	4624	Successful Logon	2016-08-19 00:28:28.043 +09:00	0d 0h 38m 48s 43ms		IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-18 23:49:40.000 +09:00	Sec	4624	Successful Logon	2016-08-19 00:28:28.043 +09:00	0d 0h 38m 48s 43ms		IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-19 00:29:27.609 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM	-	-	-	0 - SYSTEM
2016-08-19 00:29:27.609 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM	-	-	-	0 - SYSTEM
2016-08-19 00:29:29.859 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-19 00:29:29.859 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-19 00:29:29.859 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-19 00:29:29.859 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	127.0.0.1	2 - INTERACTIVE
2016-08-19 03:46:18.937 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM	-	-	-	0 - SYSTEM
2016-08-19 03:46:18.937 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM	-	-	-	0 - SYSTEM
2016-09-20 01:50:06.477 +09:00	Sec	4625	Failed Logon			Unknown - UNKNOWN USER	DESKTOP-MSSND4R	JcDkZtc		6hgtmVtrFuW065	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.477 +09:00	Sec	4625	Failed Logon			Unknown - UNKNOWN USER	DESKTOP-MSSND4R	JcDkZtc		6hgtmVtrFuW065	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.513 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		gC4ymsKbxVGSsMgY	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.513 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		gC4ymsKbxVGSsMgY	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.588 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		f2q1tdAUxH3GCH6	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.588 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		f2q1tdAUxH3GCH6	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.637 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		3EPNzcwy70AADWx	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.637 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		3EPNzcwy70AADWx	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.680 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		Abw5MP10R4h1W1	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.680 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		Abw5MP10R4h1W1	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.680 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		Abw5MP10R4h1W1	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.725 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		EEcdqppqsQ4RpPx	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.725 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		EEcdqppqsQ4RpPx	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.773 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		ngdRwzXX0AfrxGy	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.773 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		ngdRwzXX0AfrxGy	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.816 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		BbCFZw5qQgU7iQ9w	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.816 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		BbCFZw5qQgU7iQ9w	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.869 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		SxV7IA3MkV6xK36f	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.869 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		SxV7IA3MkV6xK36f	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.909 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		YVFS1R0AaOutml	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.909 +09:00	Sec	4625	Failed Logon			Unknown - WRONG PW	DESKTOP-MSSND4R	Administrator		YVFS1R0AaOutml	192.168.198.149	3 - NETWORK

## timeline-partition-diagnostic command

Creates a CSV timeline of partition diagnostic events by parsing Windows 10 `Microsoft-Windows-Partition%4Diagnostic.evtx` files and reporting information about all the connected devices and their Volume Serial Numbers, both currently present on the device and previously existed. This process is based on the tool [Partition-4DiagnosticParser](#).

- Input: JSONL
- Profile: Any
- Output: Terminal or CSV

Required options:

- `-t, --timeline <JSONL-FILE>` : JSONL timeline created by Hayabusa.

Options:

- `-o, --output <CSV-FILE>` : the CSV file to save the results to.
- `-q, --quiet` : do not display logo. (default: `false` )

## timeline-partition-diagnostic command examples

Prepare JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Create a CSV timeline of connected devices:

```
takajo.exe timeline-partition-diagnostic -t ../hayabusa/timeline.jsonl -o partition-diagnostic-timeline.csv
```

## timeline-suspicious-processes command

Create a CSV timeline of suspicious processes.

- Input: JSONL
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: Terminal or CSV

Required options:

- `-t, --timeline <JSONL-FILE>` : JSONL timeline created by Hayabusa.

Options:

- `-l, --level <LEVEL>` : specify the minimum alert level. (default: `high` )
- `-o, --output <CSV-FILE>` : the CSV file to save the results to.
- `-q, --quiet` : do not display logo. (default: `false` )

## timeline-suspicious-processes command examples

Prepare JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w
```

Search for processes that had an alert level of `high` or above and output results to screen:

```
takajo.exe timeline-suspicious-process -t ../hayabusa/timeline.jsonl
```

Search for processes that had an alert level of `low` or above and output results to screen:

```
takajo.exe timeline-suspicious-process -t ../hayabusa/timeline.jsonl -l low
```

Save the results to a CSV file:

```
takajo.exe timeline-suspicious-process -t ../hayabusa/timeline.jsonl -o suspicious-processes.csv
```

### timeline-suspicious-processes screenshot

Timestamp	Computer	Type	Level	Rule	Cmdline
2019-05-20 02:32:00.482 +09:00	DC1.inse...	Sysmon 1	high	Proc Exec (Sysmon Alert)	attrib +h nbtskan.exe
2019-05-22 00:32:57.286 +09:00	IEWIN7	Sysmon 1	high	Rundll32 Execution Without DLL File	rundll32.exe javascript:"..\mshtml,RunHTMLApplication "
2019-05-22 00:32:57.286 +09:00	IEWIN7	Sysmon 1	high	Rundll32 Execution Without DLL File	rundll32.exe javascript:"..\mshtml,RunHTMLApplication "
2019-05-22 00:32:57.286 +09:00	IEWIN7	Sysmon 1	high	Mshtml DLL RunHTMLApplication Abuse	cmd.exe /C rundll32.exe javascript:"..\mshtml,RunHTMLAp
2019-05-22 00:32:57.286 +09:00	IEWIN7	Sysmon 1	high	Mshtml DLL RunHTMLApplication Abuse	rundll32.exe javascript:"..\mshtml,RunHTMLApplication "
2019-05-22 00:32:57.286 +09:00	IEWIN7	Sysmon 1	high	Mshtml DLL RunHTMLApplication Abuse	cmd.exe /C rundll32.exe javascript:"..\mshtml,RunHTMLAp
2019-05-22 00:32:57.286 +09:00	IEWIN7	Sysmon 1	high	Mshtml DLL RunHTMLApplication Abuse	rundll32.exe javascript:"..\mshtml,RunHTMLApplication "
2019-05-22 00:32:57.867 +09:00	IEWIN7	Sysmon 1	high	Remotely Hosted HTA File Executed Via Mshta...	"C:\Windows\System32\mshta.exe" https://hotelesms.com/ta
2019-05-22 00:32:57.867 +09:00	IEWIN7	Sysmon 1	high	Remotely Hosted HTA File Executed Via Mshta...	"C:\Windows\System32\mshta.exe" https://hotelesms.com/ta
2019-05-22 00:32:57.867 +09:00	IEWIN7	Sysmon 1	high	Windows Shell/Scripting Processes Spawning ...	"C:\Windows\System32\mshta.exe" https://hotelesms.com/ta
2019-05-22 00:32:57.867 +09:00	IEWIN7	Sysmon 1	high	Windows Shell/Scripting Processes Spawning ...	"C:\Windows\System32\mshta.exe" https://hotelesms.com/ta
2019-05-22 00:32:59.769 +09:00	IEWIN7	Sysmon 1	high	Suspicious Command Patterns In Scheduled Ta...	"C:\Windows\System32\schtasks.exe" /Create /sc MINUTE /M
2019-05-22 00:32:59.769 +09:00	IEWIN7	Sysmon 1	high	Suspicious Command Patterns In Scheduled Ta...	"C:\Windows\System32\schtasks.exe" /Create /sc MINUTE /M
2019-05-22 00:32:59.769 +09:00	IEWIN7	Sysmon 1	high	Windows Shell/Scripting Processes Spawning ...	"C:\Windows\System32\schtasks.exe" /Create /sc MINUTE /M
2019-05-22 00:32:59.769 +09:00	IEWIN7	Sysmon 1	high	Windows Shell/Scripting Processes Spawning ...	"C:\Windows\System32\schtasks.exe" /Create /sc MINUTE /M
2019-05-24 02:46:04.671 +09:00	IEWIN7	Sysmon 1	high	RDP Port Forwarding Rule Added Via Netsh.EXE	netsh I p a v l=8001 listena=1.2.3.4 connectp=3389 c=1.2
2019-05-24 02:46:04.671 +09:00	IEWIN7	Sysmon 1	high	RDP Port Forwarding Rule Added Via Netsh.EXE	netsh I p a v l=8001 listena=1.2.3.4 connectp=3389 c=1.2
2019-05-24 10:33:53.112 +09:00	IEWIN7	Sysmon 1	high	Suspicious Process By Web Server Process	"c:\windows\system32\cmd.exe" /c net user"
2019-05-24 10:33:53.112 +09:00	IEWIN7	Sysmon 1	high	Suspicious Process By Web Server Process	"c:\windows\system32\cmd.exe" /c net user"
2019-05-26 13:01:43.567 +09:00	IEWIN7	Sysmon 1	high	Suspect Svchost Activity	C:\Windows\system32\svchost.exe
2019-05-26 13:01:43.567 +09:00	IEWIN7	Sysmon 1	high	Suspect Svchost Activity	C:\Windows\system32\svchost.exe
2019-05-26 13:01:43.567 +09:00	IEWIN7	Sysmon 1	high	Suspicious Svchost Process	C:\Windows\system32\svchost.exe
2019-05-26 13:01:43.567 +09:00	IEWIN7	Sysmon 1	high	Suspicious Svchost Process	C:\Windows\system32\svchost.exe
2019-05-27 10:28:42.711 +09:00	IEWIN7	Sysmon 1	high	Suspicious Encoded PowerShell Command Line	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.e

## TTP Commands

### ttp-summary command

This command summarize tactics and techniques found in each computer according to the MITRE ATT&CK TTPs defined in the `tags` field of the sigma rules.

- Input: JSONL
- Profile: A profile that outputs `%MitreTactics%` and `%MitreTags%` fields. (Ex: `verbose`, `all-field-info-verbose`, `super-verbose`)
- Output: Terminal or CSV

Required options:

- `-t, --timeline <JSONL-FILE>` : JSONL timeline created by Hayabusa.

Options:

- `-o, --output <CSV-FILE>` : the CSV file to save the results to.
- `-q, --quiet` : do not display logo. (default: `false`)

### ttp-summary command examples

Prepare JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w -p verbose
```

Print TTP summary to terminal:

```
takajo.exe ttp-summary -t ../hayabusa/timeline.jsonl
```

Save the results to a CSV file:

```
takajo.exe ttp-summary -t ../hayabusa/timeline.jsonl -o ttp-summary.csv
```

### ttp-summary screenshot

Computer	Tactic	Technique	Sub-Technique	Count
01566s-win16-ir.threebeesco.com	06. Privilege Escalation	Account Manipulation	-	1
01566s-win16-ir.threebeesco.com	06. Privilege Escalation	Scheduled Task/Job	Scheduled Task	1
01566s-win16-ir.threebeesco.com	07. Defense Evasion	Indicator Removal	Clear Windows Event Logs	1
01566s-win16-ir.threebeesco.com	07. Defense Evasion	Masquerading	-	8
01566s-win16-ir.threebeesco.com	07. Defense Evasion	Rogue Domain Controller	-	3
01566s-win16-ir.threebeesco.com	10. Lateral Movement	Remote Services	SMB/Windows Admin Shares	1
01566s-win16-ir.threebeesco.com	11. Collection	Data from Network Shared Drive	-	4
02694w-win10.threebeesco.com	06. Privilege Escalation	Boot or Logon Autostart Execution	LSASS Driver	6
02694w-win10.threebeesco.com	07. Defense Evasion	Hijack Execution Flow	-	2
02694w-win10.threebeesco.com	08. Credential Access	OS Credential Dumping	LSASS Memory	1
02694w-win10.threebeesco.com	11. Collection	Command and Scripting Interpreter	Python	2
04246w-win10.threebeesco.com	04. Execution	System Services	Service Execution	1
2012r2srv.maincorp.local	07. Defense Evasion	Access Token Manipulation	SID-History Injection	1
2012r2srv.maincorp.local	09. Discovery	Password Policy Discovery	-	1
2016dc.hqcorp.local	06. Privilege Escalation	Account Manipulation	-	1
DC1.insecurebank.local	06. Privilege Escalation	Account Manipulation	-	1

### ttp-visualize command

This command extracts TTPs and create a JSON file to visualize in [MITRE ATT&CK Navigator](#).

- Input: JSONL
- Profile: A profile that outputs %MitreTactics% and %MitreTags% fields. (Ex: verbose , all-field-info-verbose , super-verbose )
- Output: Terminal or CSV

Required options:

- -t, --timeline <JSONL-FILE> : JSONL timeline created by Hayabusa.

Options:

- -o, --output <JSON-FILE> : the JSON file to save the results to. (default: mitre-attack-navigator.json )
- -q, --quiet : do not display logo. (default: false )

### ttp-visualize command examples

Prepare JSONL timeline with Hayabusa:

```
hayabusa.exe json-timeline -d <EVTX-DIR> -L -o timeline.jsonl -w -p verbose
```

Extract out the TTPs and save to `mitre-attack-navigator.json` :

```
takajo.exe ttp-visualize -t ../hayabusa/timeline.jsonl
```

Open <https://mitre-attack.github.io/attack-navigator/>, click `Open Existing Layer` and upload the saved JSON file.

### ttp-visualize screenshot

Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
10 techniques	8 techniques	10 techniques	14 techniques	20 techniques	14 techniques	43 techniques	17 techniques	32 techniques	9 techniques	17 techniques	17 techniques	9 techniques	14 techniques
Active Scanning (029)	Acquire Access (029)	Content Injection (029)	Cloud Administration Command (029)	Account Manipulation (029)	Abuse Elevation Control Mechanism (103)	Abuse Elevation Control Mechanism (103)	Adversary-in-the-Middle (103)	Account Discovery (244)	Exploitation of Remote Services (029)	Adversary-in-the-Middle (103)	Application Layer Protocol (103)	Automated Exfiltration (029)	Account Access Removal (029)
Gather Victim Host Information (042)	Acquire Infrastructure (042)	Drive-by Compromise (042)	Command and Scripting Interpreter (042)	BITS Jobs (042)	Access Token Manipulation (042)	Access Token Manipulation (042)	Brute Force (042)	Application Window Discovery (042)	Internal Spearphishing (042)	Archive Collected Data (042)	Communication Through Removable Media (042)	Data Transfer Site Limits (042)	Data Destruction (042)
Gather Victim Identity Information (029)	Compromise Accounts (029)	Exploit Public-Facing Application (029)	Container Administration Command (029)	Boot or Logon Autostart Execution (474)	Access Token Manipulation (042)	Access Token Manipulation (042)	Credentials from Password Stores (106)	Browser Information Discovery (042)	Remote Service Hijacking (029)	Audio Capture (029)	Content Injection (029)	Data Encrypted for Impact (029)	Data Manipulation (029)
Gather Victim Network Information (042)	Compromise Infrastructure (042)	External Remote Services (042)	Container Administration Command (042)	Boot or Logon Initialization Scripts (042)	Account Manipulation (042)	Account Manipulation (042)	Exploitation for Credential Access (042)	Cloud Infrastructure Discovery (042)	Remote Service Hijacking (029)	Automated Collection (029)	Data Encoding (029)	Defacement (029)	Disk Wipe (029)
Gather Victim Org Information (042)	Develop Capabilities (042)	Hardware Additions (042)	Deploy Container (042)	Browser Extensions (042)	Boot or Logon Autostart Execution (474)	Boot or Logon Autostart Execution (474)	Exploitation for Credential Access (042)	Cloud Service Dashboard (042)	Remote Service Hijacking (029)	Automated Collection (029)	Data Encoding (029)	Endpoint Denial of Service (042)	Financial Theft (042)
Phishing for Information (042)	Establish Accounts (029)	Phishing (141)	Exploitation for Client Execution (042)	Browser Extensions (042)	Compromise Client Software Binary (042)	Compromise Client Software Binary (042)	Forge Web Credentials (042)	Cloud Storage Object Discovery (042)	Remote Service Hijacking (029)	Clipboard Data (042)	Dynamic Resolution (029)	Exfiltration Over Other Network Channel (042)	Inhibit System Recovery (042)
Search Closed Sources (029)	Obtain Capabilities (029)	Replication Through Removable Media (029)	Inter-Process Communication (042)	Compromise Client Software Binary (042)	Boot or Logon Initialization Scripts (105)	Boot or Logon Initialization Scripts (105)	Input Capture (042)	Container and Resource Discovery (042)	Remote Service Hijacking (029)	Data from Cloud Storage (042)	Encrypted Channel (029)	Firmware Corruption (042)	Network Denial of Service (029)
Search Open Technical Databases (042)	Stage Capabilities (042)	Supply Chain Compromise (042)	Scheduled Task/Job (209)	Create or Modify System Process (144)	Create or Modify System Process (144)	Create or Modify System Process (144)	Modify Authentication Process (042)	Debugger Evasion (042)	Software Deployment Tools (042)	Data from Configuration Repository (042)	Failback Channels (042)	Exfiltration Over Physical Medium (029)	Resource Hijacking (042)
Search Open Websites/Domains (029)	Valid Relationships (029)	Trusted Relationship (029)	Serverless Execution (042)	Event Triggered Execution (042)	Domain Policy Modification (042)	Domain Policy Modification (042)	Multi-Factor Authentication Interception (042)	File and Directory Discovery (042)	Task Shared Content (042)	Data from Information Repositories (042)	Multi-Stage Channels (042)	Scheduled Transfer (042)	System Shutdown/Reboot (042)
Search Victim-Owned Websites (029)	Valid Accounts (029)	Valid Accounts (029)	Software Deployment Tools (042)	External Remote Services (042)	Hide Artifacts (211)	Hide Artifacts (211)	Multi-Factor Authentication Interception (042)	Group Policy Discovery (042)	Use Alternate Authentication Material (144)	Data from Local System (042)	Non-Application Layer Protocol (042)	Transfer Data to Cloud Account (042)	
			System Services (102)	Event Triggered Execution (042)	Hijack Execution Flow (202)	Hijack Execution Flow (202)	Network Sniffing (042)	Log Enumeration (042)	Data from Removable Media (042)	Data from Network Shared Drive (042)	Non-Standard Port (042)	Protocol Tunneling (042)	
			User Execution (103)	Implant Internal Image (042)	Impersonation (311)	Impersonation (311)	OS Credential Dumping (068)	Network Share Discovery (042)	Data from Removable Media (042)	Data from Removable Media (042)	Proxy (244)	Remote Access Software (042)	
			Windows Management Instrumentation (042)	Modify Authentication Process (042)	Indicator Removal (209)	Indicator Removal (209)	Steal Application Access Token (042)	Password Policy Discovery (042)	Email Collection (029)	Input Capture (042)	Web Service (029)		
			Office Application Startup (272)	Process Injection (042)	Masquerading (209)	Masquerading (209)	Steal or Forge Authentication Certificates (042)	Peripheral Device Discovery (042)					
			Power Settings (042)	Scheduled Task/Job (209)	Modify Cloud Compute Infrastructure (042)	Modify Cloud Compute Infrastructure (042)	Steal or Forge Kerberos Tickets (106)	Permission Groups Discovery (209)					
			Pre-OS Boot (029)	Valid Accounts (029)	Modify Registry (042)	Modify Registry (042)	Process Discovery (042)	Process Discovery (042)					

## VirusTotal Commands

### vt-domain-lookup command

Look up a list of domains on VirusTotal

- Input: Text file
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: CSV

Required options:

- `-a, --apiKey <API-KEY>` : your VirusTotal API key.
- `-d, --domainList <TXT-FILE>` : a text file list of domains.
- `-o, --output <CSV-FILE>` : save the results to a CSV file.

Options:

- `-j, --jsonOutput <JSON-FILE>` : output all of the JSON responses from VirusTotal to a JSON file.
- `-r, --rateLimit <NUMBER>` : the rate per minute to send requests. (default: 4 )
- `-q, --quiet` : do not display logo. (default: false )

### vt-domain-lookup command examples

First create a list of domains with the `list-domains` command. Then lookup those domains with the following:

```
takajo.exe vt-domain-lookup -a <API-KEY> -d domains.txt -o vt-domain-lookup.csv -r 1000 -j vt-domain-lookup.json
```

### vt-hash-lookup command

Look up a list of hashes on VirusTotal.

- Input: Text file
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: CSV

Required options:

- `-a, --apiKey <API-KEY>` : your VirusTotal API key.
- `-H, --hashList <HASH-LIST>` : a text file of hashes.
- `-o, --output <CSV-FILE>` : save the results to a CSV file.

Options:

- `-j, --jsonOutput <JSON-FILE>` : output all of the JSON responses from VirusTotal to a JSON file.
- `-r, --rateLimit <NUMBER>` : the rate per minute to send requests. (default: 4 )
- `-q, --quiet` : do not display logo. (default: false )

### vt-hash-lookup command examples

```
takajo.exe vt-hash-lookup -a <API-KEY> -H MD5-hashes.txt -o vt-hash-lookup.csv -r 1000 -j vt-hash-lookup.json
```

### vt-ip-lookup command

Look up a list of IP addresses on VirusTotal.

- Input: Text file
- Profile: Any besides `all-field-info` and `all-field-info-verbose`
- Output: CSV

Required options:

- `-a, --apiKey <API-KEY>` : your VirusTotal API key.
- `-i, --ipList <IP-ADDRESS-LIST>` : a text file of IP addresses.
- `-o, --output <CSV-FILE>` : save the results to a CSV file.

Options:

- `-j, --jsonOutput <JSON-FILE>` : output all of the JSON responses from VirusTotal to a JSON file.
- `-r, --rateLimit <NUMBER>` : the rate per minute to send requests. (default: 4 )
- `-q, --quiet` : do not display logo. (default: false )

### vt-ip-lookup command examples

```
takajo.exe vt-ip-lookup -a <API-KEY> -i ipAddresses.txt -o vt-ip-lookup.csv -r 1000 -j vt-ip-lookup.json
```

## Contribution

We would love any form of contribution. Pull requests, rule creation and sample evt logs are the best but feature requests, notifying us of bugs, etc... are also very welcome.

At the least, if you like our tool then please give us a star on Github and show your support!

## Bug Submission

Please submit any bugs you find [here](#). This project is currently actively maintained and we are happy to fix any bugs reported.

If you find any issues (false positives, bugs, etc...) with Hayabusa, please report them to the hayabusa github issues page [here](#).

If you find any issues (false positives, bugs, etc...) with Hayabusa rules, please report them to the hayabusa-rules github issues page [here](#).

If you find any issues (false positives, bugs, etc...) with Sigma rules, please report them to the upstream SigmaHQ github issues page [here](#).

## License

Takajō is released under the [GPLv3](#) license.

## Twitter

You can receive the latest news about Takajō, Hayabusa, rule updates, other Yamato Security tools, etc... by following us on Twitter at [@SecurityYamato](#).