

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

Takajōについて

Takajō (鷹匠)は日本の[Yamato Security](#)グループによって作られた [Hayabusa](#)から得られた結果を解析するツールです。Takajōは[Nim](#)で作られました。Takajōは、日本語で"[鷹狩りのスキルに優れた人](#)"を意味し、ハヤブサが得た結果をさらに活かすことから選ばれました。

関連プロジェクト

- [EnableWindowsLogSettings](#) - Sigmaベースの脅威ハンティングと、Windowsイベントログのファストフォレンジックタイムライン生成ツール。
- [Hayabusa](#) - Windowsイベントログを正しく設定するためのドキュメンテーションとスクリプト。
- [Hayabusa Rules](#) - Hayabusaのための検知ルール。
- [Hayabusa Sample EVTxs](#) - Hayabusa/Sigma検出ルールをテストするためのサンプルevtxファイル。
- [WELA \(Windows Event Log Analyzer\)](#) - PowerShellで書かれたWindowsイベントログの解析ツール。

目次

- [関連プロジェクト](#)
 - [目次](#)
 - [機能](#)
- [ダウンロード](#)
 - [Gitクローン](#)
 - [アドバンス: ソースコードからのコンパイル \(任意\)](#)
- [コマンド一覧](#)
 - [Extractコマンド](#)

- [Listコマンド](#)
- [Splitコマンド](#)
- [Stackコマンド](#)
- [Sysmonコマンド](#)
- [Timelineコマンド](#)
- [TTP Commands](#)
- [VirusTotalコマンド](#)
- [コマンド使用方法](#)
 - [Extractコマンド](#)
 - [extract-scriptblocks コマンド](#)
 - [extract-scriptblocks コマンドの使用例](#)
 - [extract-scriptblocks スクリーンショット](#)
 - [Listコマンド](#)
 - [list-domains コマンド](#)
 - [list-domains コマンドの使用例](#)
 - [list-hashes コマンド](#)
 - [list-hashes コマンドの使用例](#)
 - [list-ip-addresses コマンド](#)
 - [list-ip-addresses コマンドの使用例](#)
 - [list-undetected-evtx コマンド](#)
 - [list-undetected-evtx コマンドの使用例](#)
 - [list-unused-rules コマンド](#)
 - [list-unused-rules コマンドの使用例](#)
 - [Splitコマンド](#)
 - [split-csv-timeline コマンド](#)
 - [split-csv-timeline コマンドの使用例](#)
 - [split-json-timeline コマンド](#)
 - [split-json-timeline コマンドの使用例](#)
 - [Stackコマンド](#)
 - [stack-logons コマンド](#)
 - [stack-logons コマンドの使用例](#)
 - [Sysmonコマンド](#)
 - [sysmon-process-tree コマンド](#)
 - [sysmon-process-tree コマンドの使用例](#)
 - [sysmon-process-tree スクリーンショット](#)
 - [Timelineコマンド](#)
 - [timeline-logon コマンド](#)
 - [timeline-logon コマンドの使用例](#)
 - [timeline-logon スクリーンショット](#)
 - [timeline-partition-diagnostic コマンド](#)
 - [timeline-partition-diagnostic コマンドの使用例](#)
 - [timeline-suspicious-processes コマンド](#)
 - [timeline-suspicious-processes コマンドの使用例](#)

- [timeline-suspicious-processes](#) スクリーンショット
- [TTPコマンド](#)
 - [ttp-summary](#) コマンド
 - [ttp-summary](#) コマンドの使用例
 - [ttp-summary](#) スクリーンショット
 - [ttp-visualize](#) コマンド
 - [ttp-visualize](#) コマンドの使用例
 - [ttp-visualize](#) スクリーンショット
- [VirusTotalコマンド](#)
 - [vt-domain-lookup](#) コマンド
 - [vt-domain-lookup](#) コマンドの使用例
 - [vt-hash-lookup](#) コマンド
 - [vt-hash-lookup](#) コマンドの使用例
 - [vt-ip-lookup](#) コマンド
 - [vt-ip-lookup](#) コマンドの使用例
- [貢献](#)
- [バグの報告](#)
- [ライセンス](#)
- [Twitter](#)

機能

- Nimで開発され、プログラミングが簡単、メモリ安全、ネイティブCコードと同じくらい高速で、単一のスタンドアロンバイナリとして動作します。
- ログオンイベント、疑わしいプロセスなどさまざまなタイムラインを作成します。
- 不審なプロセスのプロセスツリーを出力します。
- さまざまなスタッキング分析ができます。
- CSVとJSONLのタイムラインを分割します。
- VirusTotalの検索で使用するIPアドレス、ドメイン、ハッシュなどをリストアップします。
- ドメイン、ハッシュ、IPアドレスをVirusTotalで検索します。
- 検知されていない .evtx ファイルをリストアップします。
- MITRE ATT&CK NavigatorでTTPを可視化します。
- その他、たくさん！

ダウンロード

[Releases](#) ページからTakajōの安定したバージョンでコンパイルされたバイナリが含まれている最新版もしくはソースコードをダウンロードできます。

Gitクローン

以下の `git clone` コマンドでレポジトリをダウンロードし、ソースコードからコンパイルして使用することも可能です：

注意: `main` ブランチは開発中のバージョンです。まだ正式にリリースされていない新機能が使えるかもしれませんが、バグがある可能性もあるので、テスト版だと思って下さい。

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

アドバンス: ソースコードからのコンパイル（任意）

Nimがインストールされている場合、以下のコマンドでソースコードからコンパイルできます:

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

コマンド一覧

Extractコマンド

- `extract-scriptblocks` : PowerShell EID 4104 スクリプトブロックログからPowerShellスクリプトを抽出して再構築する

Listコマンド

- `list-domains` : `vt-domain-lookup` コマンドで使用する、重複のないドメインのリストを作成する
- `list-hashes` : `vt-hash-lookup` で使用するプロセスのハッシュ値のリストを作成する
- `list-ip-addresses` : `vt-ip-lookup` コマンドで使用する、重複のない送信元/送信先のIPリストを作成する
- `list-undetected-evtx` : 検知されなかったevtxファイルのリストを作成する
- `list-unused-rules` : 検知されなかったルール of リストを作成する

Splitコマンド

- `split-csv-timeline` : コンピューター名に基づき、大きなCSVタイムラインを小さなCSVタイムラインに分割する
- `split-json-timeline` : コンピューター名に基づき、大きなJSONLタイムラインを小さなJSONLタイムラインに分割する

Stackコマンド

- `stack-logons` : ユーザー名、コンピューター名、送信元IPアドレス、送信元コンピューター名など、項目ごとの上位ログオンを出力する

Sysmonコマンド

- `sysmon-process-tree` : プロセスツリーを出力する

Timelineコマンド

- `timeline-logon` : ログオンイベントのCSVタイムラインを作成する
- `timeline-suspicious-processes` : 不審なプロセスのCSVタイムラインを作成する
- `timeline-partition-diagnostic` : partition diagnostic イベントのCSVタイムラインを作成する

TTP Commands

- `ttp-summary` : コンピュータ毎に検知されたTTPsの要約を出力する
- `ttp-visualize` : TTPs を抽出し、MITRE ATT&CK Navigator で視覚化するためのJSONファイルを作成する

VirusTotalコマンド

- `vt-domain-lookup` : VirusTotalでドメインのリストを検索し、悪意のあるドメインをレポートする
- `vt-hash-lookup` : VirusTotalでハッシュのリストを検索し、悪意のあるハッシュ値をレポートする
- `vt-ip-lookup` : VirusTotalでIPアドレスのリストを検索し、悪意のあるIPアドレスをレポートする

コマンド使用方法

Extractコマンド

`extract-scriptblocks` コマンド

PowerShell EID 4104 スクリプトブロックログからPowerShellスクリプトを抽出して再構築します。

注意: PowerShellスクリプトは、コード構文が強調表示された「.ps1」ファイルとして開くのが最適ですが、悪意のあるコードが誤って実行されるのを防ぐために「.txt」拡張子を使用します。

- 入力: JSONL
- プロファイル: すべて
- 出力: ターミナルとPowerShellスクリプトのディレクトリ

必須オプション:

- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-l, --level` : 最小のアラートレベルを指定 (デフォルト: `low`)
- `-o, --output` : PowerShellスクリプトを保存するディレクトリ (デフォルト: `scriptblock-logs`)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

`extract-scriptblocks` コマンドの使用例

HayabusaでJSONLタイムラインを出力する:

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

PowerShell EID 4104 スクリプトブロックログから抽出し、 `scriptblock-logs` ディレクトリに保存します:

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

`extract-scriptblocks` スクリーンショット

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.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-c3103e9bbc3a	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コマンド

list-domains コマンド

`vt-domain-lookup` で使用する重複のないドメインのリストを作成します。現在は、Sysmon EID 22ログでクエリが記録されたドメインのみをチェックしますが、ビルトインのWindows DNSクライアント・サーバーログも今後サポート予定です。

- 入力: JSONL
- プロファイル: `all-field-info` と `all-field-info-verbose` 以外すべて
- 出力: テキストファイル

必須オプション:

- `-o, --output <TXT-FILE>` : 結果を保存するテキストファイル
- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-s, --includeSubdomains` : サブドメインを含めるか (デフォルト: `false`)
- `-w, --includeWorkstations` : ローカルワークステーション名を含めるか (デフォルト: `false`)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

list-domains コマンドの使用例

HayabusaでJSONLタイムラインを出力する:

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

結果をテキストファイルに保存する:

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

サブドメインを含める場合:

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

list-hashes コマンド

`vt-hash-lookup` で使用するプロセスハッシュ値のリストを作成します (入力: JSONL, プロファイル: standard)

- 入力: JSONL
- プロファイル: `all-field-info` と `all-field-info-verbose` 以外すべて
- 出力: テキストファイル

必須オプション:

- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン
- `-o, --output <BASE-NAME>` : 結果を保存するベースファイル名

任意オプション:

- `-l, --level` : 最小のアラートレベルを指定 (デフォルト: `high`)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

list-hashes コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

ハッシュタイプ毎に異なるファイルに結果を保存する:

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

たとえば、MD5 、 SHA1 、 IMPHASH がSysmonログに保存されている場合、次のファルが作成されます:

```
case-1-MD5-hashes.txt , case-1-SHA1-hashes.txt , case-1-ImportHashes.txt
```

list-ip-addresses コマンド

`vt-ip-lookup` で使用する重複のない送信先/送信元IPアドレスのリストを作成します。すべての結果から送信先IPアドレスの `TgtIP` フィールドと送信元IPアドレスの `SrcIP` フィールドが抽出され、重複のないIPアドレスをテキストファイルに出力します。

- 入力: JSONL
- プロファイル: `all-field-info` と `all-field-info-verbose` 以外すべて
- 出力: テキストファイル

必須オプション:

- `-o, --output <TXT-FILE>` : 結果を保存するテキストファイル
- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-i, --inbound` : インバウンドトラフィックを含めるか (デフォルト: `true`)
- `-O, --outbound` : アウトバウンドトラフィックを含めるか (デフォルト: `true`)
- `-p, --privateIp` : プライベートIPアドレスを含めるか (デフォルト: `false`)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

list-ip-addresses コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

結果をテキストファイルに保存する:

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

インバウンドトラフィックを除外する:

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

プライベートIPアドレスを含める:

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

list-undetected-evtx コマンド

Hayabusaで検知するルールがなかったすべての .evtx ファイルをリストアップします。これは、[hayabusa-sample-evtx](#) リポジトリ内の evtx ファイルなど、悪意のあるアクティビティの証拠を含むすべての evtx ファイルをリストアップすることを目的としています

- 入力: CSV
- プロファイル: `verbose` , `all-field-info-verbose` , `super-verbose` , `timesketch-verbose`

まず、`%EvtxFile%` を出力するプロファイルを使用し、Hayabusaを実行、結果をCSVタイムラインに保存する必要があります [こちら](#)でHayabusaがプロファイルに従って、どのカラムを保存するかを確認できます。

- 出力: ターミナル または テキストファイル

必須オプション:

- `-e, --evtx-dir <EVTX-DIR>` : Hayabusaでスキャンした .evtx ファイルのディレクトリ
- `-t, --timeline <CSV-FILE>` : HayabusaのCSVタイムライン

任意オプション:

- `-c, --column-name <CUSTOM-EVTX-COLUMN>` : evtxのカラム名を指定 (デフォルト: Hayabusaの規定値の EvtxFile)
- `-o, --output <TXT-FILE>` : 結果を保存するテキストファイル (デフォルト: 標準出力)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

list-undetected-evtx コマンドの使用例

HayabusaでCSVタイムラインを出力する:

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

結果を標準出力に表示する:

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

結果をテキストファイルに保存する:

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

list-unused-rules コマンド

何も検出されなかったすべての `.yaml` ルールをリストアップします。これは、ルールの信頼性を判断するのに役立ちます。つまり、どのルールが悪意のあるアクティビティを検出するか、またどのルールがまだテストされておらずサンプル `.evtx` ファイルが必要かの判断に使えます。

- 入力: CSV
- プロファイル: `verbose` , `all-field-info-verbose` , `super-verbose` , `timesketch-verbose`

まず、`%RuleFile%` を出力するプロファイルを使用し、`Hayabusa`を実行、結果をCSVタイムラインに保存する必要があります [こちら](#)で`Hayabusa`がプロファイルに従って、どのカラムを保存するかを確認できます。

- 出力: ターミナル または テキストファイル

必須オプション:

- `-r, --rules-dir <DIR>` : `Hayabusa`で使用した `.yaml` ルールファイルのディレクトリ
- `-t, --timeline <CSV-FILE>` : `Hayabusa`のCSVタイムライン

任意オプション:

- `-c, --column-name <CUSTOM-RULE-FILE-COLUMN>` : ルールファイルのカラム名を指定 (デフォルト: `Hayabusa`の規定値の `RuleFile`)
- `-o, --output <TXT-FILE>` : 結果を保存するテキストファイル (デフォルト: 標準出力)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

`list-unused-rules` コマンドの使用例

`Hayabusa`でCSVタイムラインを出力する:

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

結果を標準出力に表示する:

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

結果をテキストファイルに保存する:

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

Splitコマンド

`split-csv-timeline` コマンド

コンピューター名に基づき、大きなCSVタイムラインを小さなCSVタイムラインに分割します。

- 入力: 複数行モード(-M)でないCSV
- プロファイル: すべて
- 出力: 複数のCSV

必須オプション:

- `-t, --timeline <CSV-FILE>` : `Hayabusa`のCSVタイムライン

任意オプション:

- `-m, --makeMultiline` : フィールドを複数行で出力する (デフォルト: `false`)

- `-o, --output <DIR>` : CSVを保存するディレクトリ (デフォルト: `output`)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

split-csv-timeline コマンドの使用例

HayabusaでCSVタイムラインを出力する:

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

1つのCSVタイムラインを複数のCSVタイムラインに分割して `output` ディレクトリに出力:

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

フィールドを改行文字で区切って複数行のエントリを作成し、 `case-1-csv` ディレクトリに保存:

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

split-json-timeline コマンド

コンピューター名に基づき、大きなJSONLタイムラインを小さなJSONLタイムラインに分割します。

- 入力: JSONL
- プロファイル: すべて
- 出力: 複数のJSONL

必須オプション:

- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-o, --output <DIR>` : JSONLを保存するディレクトリ (デフォルト: `output`)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

split-json-timeline コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

1つのJSONLタイムラインを複数のJSONLタイムラインに分割して `output` ディレクトリに出力:

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

`case-1-jsonl` ディレクトリに保存:

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

Stackコマンド

stack-logons コマンド

`Target User`、`Target Computer`、`Logon Type`、`Source IP Address`、`Source Computer` によってログインのリストを作成します。デフォルトでは、ローカルIPアドレスのソースIPアドレスはフィルタされません。

- 入力: JSONL
- プロファイル: `all-field-info` と `all-field-info-verbose` 以外すべて
- 出力: ターミナルまたはCSVファイル

必須オプション:

- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-l, --localSrcIpAddresses` : ソースIPアドレスがローカルIPアドレスであっても結果に含む
- `-o, --output <CSV-FILE>` : テキストの結果のベースネームを指定する
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

stack-logons コマンドの使用例

デフォルトの設定で実行する:

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

ローカルログオンを含む:

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

Sysmonコマンド

sysmon-process-tree コマンド

不審なプロセスや悪意のあるプロセスなど、特定のプロセスのプロセスツリーを出力します。

- 入力: JSONL
- プロファイル: `all-field-info` と `all-field-info-verbose` 以外すべて
- 出力: テキストファイル

必須オプション:

- `-o, --output <TXT-FILE>` : 結果を保存するテキストファイル
- `-p, --processGuid <Process GUID>` : SysmonのプロセスGUID
- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

sysmon-process-tree コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

結果をテキストファイルに保存する:

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

sysmon-process-tree スクリーンショット

```
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\wininit.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-B776-5EA4-0000-0010A74D0000 / 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-B776-5EA4-0000-0010A74D0000)

Elapsed time: 0 hours, 0 minutes, 1 seconds
```

Timelineコマンド

timeline-logon コマンド

このコマンドは、次のログオンイベントから情報を抽出し、フィールドを正規化し、結果をCSVファイルに保存します:

- 4624 - ログオン成功
- 4625 - ログオン失敗
- 4634 - アカウントログオフ
- 4647 - ユーザーが開始したログオフ
- 4648 - 明示的なログオン
- 4672 - 特権ログオン

これにより、ラテラルムーブメント、パスワードスプレー、権限昇格などを検出しやすくなります。

- 入力: JSONL
- プロファイル: all-field-info と all-field-info-verbose 以外すべて
- 出力: CSV

必須オプション:

- -o, --output <CSV-FILE> : 結果を保存するCSVファイル
- -t, --timeline <JSONL-FILE> : HayabusaのJSONLタイムライン

任意オプション:

- -c, --calculateElapsedTime : 成功ログオンの経過時間を計算する (デフォルト: true)
- -l, --outputLogoffEvents : ログオフイベントを別のエントリとして出力する (デフォルト: false)
- -a, --outputAdminLogonEvents : 管理者ログオン イベントを別のエントリとして出力する (デフォルト: false)
- -q, --quiet : ログを出力しない (デフォルト: false)

timeline-logon コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

ログオンタイムラインをCSVに保存する:

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

timeline-logon スクリーンショット

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	IEIOWin7	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		127.0.0.1	IEIOWin7	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	IEIOWin7	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		127.0.0.1	IEIOWin7	2 - INTERACTIVE
2016-08-18 23:47:38.430 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	IEIOWin7	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		127.0.0.1	IEIOWin7	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	IEIOWin7	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		127.0.0.1	IEIOWin7	2 - INTERACTIVE
2016-08-18 23:49:40.000 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	IEIOWin7	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		127.0.0.1	IEIOWin7	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	IEIOWin7	2 - INTERACTIVE
2016-08-19 00:29:29.859 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser		127.0.0.1	IEIOWin7	2 - INTERACTIVE
2016-08-19 00:29:29.859 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser	Yes	127.0.0.1	IEIOWin7	2 - INTERACTIVE
2016-08-19 00:29:29.859 +09:00	Sec	4624	Successful Logon				IEIOWin7	IEUser		127.0.0.1	IEIOWin7	2 - INTERACTIVE
2016-08-19 03:46:19.937 +09:00	Sec	4624	Successful Logon				IEIOWin7	SYSTEM		-	-	0 - SYSTEM
2016-08-19 03:46:19.937 +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	JcDcZTc		ghgtnVrrFuW065	192.168.198.149	3 - NETWORK
2016-09-20 01:50:06.477 +09:00	Sec	4625	Failed Logon			Unknown - UNKNOWN USER	DESKTOP-MSSND4R	JcDcZTc		ghgtnVrrFuW065	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		gC4ym5KxwVGScMgY	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		gC4ym5KxwVGScMgY	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		f2q1tdAUlXHGQH6	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		f2q1tdAUlXHGQH6	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		AbwesMP10Rs4h1Wl1	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		AbwesMP10Rs4h1Wl1	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		EEcdqcpq5xQ4RgPx	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		EEcdqcpq5xQ4RgPx	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		ngdIRwzXXhARxGY	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		ngdIRwzXXhARxGY	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		BbCFzw5q9gU7rQ9W	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		BbCFzw5q9gU7rQ9W	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		SXv7IA3MvV6xK36f	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		SXv7IA3MvV6xK36f	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		IVF51kR0AuOutnml	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		IVF51kR0AuOutnml	192.168.198.149	3 - NETWORK

timeline-partition-diagnostic コマンド

partition diagnosticイベントのCSVタイムラインを作成します。Windows 10の Microsoft-Windows-Partition%4Diagnostic.evtx を解析し、現在および過去に接続されたデバイスのボリュームシリアル番号を出力します。この処理は [Partition-4DiagnosticParser](#)を参考にして作成されました。

- 入力: JSONL
- プロファイル: すべて
- 出力: CSV

必須オプション:

- t, --timeline <JSONL-FILE> : HayabusaのJSONLタイムライン

任意オプション:

- o, --output <CSV-FILE> : 結果を保存するCSVファイル
- q, --quiet : ログを出力しない(デフォルト: false)

timeline-partition-diagnostic コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

接続されたデバイスのCSVタイムラインを作成する:

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

timeline-suspicious-processes コマンド

不審なプロセスのCSVタイムラインを作成します。

- 入力: JSONL
- プロファイル: all-field-info と all-field-info-verbose 以外すべて
- 出力: CSV

必須オプション:

- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-l, --level <LEVEL>` : 最小のアラートレベルを指定 (デフォルト: `high`)
- `-o, --output <CSV-FILE>` : 結果を保存するCSVファイル
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

timeline-suspicious-processes コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

アラートレベルが `high` 以上のプロセスを検索し、結果を標準出力に表示:

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

アラートレベルが `low` 以上のプロセスを検索し、結果を標準出力に表示:

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

結果をCSVに保存:

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

timeline-suspicious-processes スクリーンショット

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 nbtscan.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コマンド

ttp-summary コマンド

このコマンドは、Sigmaルールの「tags」フィールドで定義された MITRE ATT&CK TTP に従って、各コンピュータで見つかった戦術とテクニックの要約を出力します。

- 入力: JSONL

- プロファイル: %MitreTactics% と %MitreTags% フィールドを出力するプロファイル (例: `verbose` , `all-field-info-verbose` , `super-verbose`)
- 出力: ターミナル または CSV

必須オプション:

- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-o, --output <CSV-FILE>` : 結果を保存するCSVファイル
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

ttp-summary コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

TTPsの要約を表示する:

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

結果をCSVに保存:

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

ttp-summary スクリーンショット

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 コマンド

TTPsを抽出し、[MITRE ATT&CK Navigator](#)で視覚化するための JSON ファイルを作成します。

- 入力: JSONL
- プロファイル: %MitreTactics% と %MitreTags% フィールドを出力するプロファイル (例: `verbose` , `all-field-info-verbose` , `super-verbose`)
- 出力: JSON

必須オプション:

- `-t, --timeline <JSONL-FILE>` : HayabusaのJSONLタイムライン

任意オプション:

- `-o, --output <JSON-FILE>` : 結果を保存するJSONファイル (デフォルト: `mitre-attack-navigator.json`)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

ttp-visualize コマンドの使用例

HayabusaでJSONLタイムラインを作成する:

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

TTPsを抽出し、`mitre-attack-navigator.json` に保存する:

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

<https://mitre-attack.github.io/attack-navigator/>を開き、`Open Existing Layer` をクリック、JSONファイルをアップロードする。

ttp-visualize スクリーンショット

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 (52)	Acquire Access (52)	Content Injection (52)	Cloud Administration Command (52)	Account Manipulation (52)	Abuse Elevation Control Mechanism (15)	Abuse Elevation Control Mechanism (15)	Adversary-In-the-Middle (15)	Account Discovery (24)	Exploitation of Remote Services (15)	Adversary-In-the-Middle (15)	Application Layer Protocol (15)	Automated Exfiltration (52)	Account Access Removal (52)
Gather Victim Host Information (24)	Acquire Infrastructure (52)	Drive-by Compromise (52)	Command and Scripting Interpreter (52)	BITS Jobs (52)	Access Token Manipulation (38)	Access Token Manipulation (38)	Brute Force (15)	Browser Information Discovery (15)	Internal Spearphishing (15)	Archive Collected Data (15)	Communication Through Removable Media (15)	Data Transfer Size Limits (52)	Data Destruction (52)
Gather Victim Identity Information (52)	Compromise Accounts (52)	Exploit Public-Facing Application (52)	Container Administration Command (52)	Boot or Logon Autostart Execution (41)	Account Manipulation (38)	Account Manipulation (38)	Credentials from Password Store (15)	Cloud Infrastructure Discovery (15)	Remote Service Hijacking (22)	Audio Capture (15)	Content Injection (15)	Exfiltration Over Alternative Protocol (52)	Data Encrypted for Impact (52)
Gather Victim Network Information (52)	Compromise Infrastructure (52)	External Remote Services (52)	Deploy Container (52)	Boot or Logon Initialization Scripts (52)	Browser Extensions (52)	Browser Extensions (52)	Exploitation for Credential Access (15)	Cloud Service Dashboard (15)	Remote Service Hijacking (22)	Browser Session Hijacking (22)	Data Encoding (15)	Disk Wipe (52)	Data Manipulation (52)
Gather Victim Org Information (24)	Develop Capabilities (14)	Hardware Additions (52)	Exploitation for Client Execution (52)	Browser Extensions (52)	Boot or Logon Autostart Execution (41)	Boot or Logon Autostart Execution (41)	Forced Authentication (15)	Cloud Service Discovery (15)	Replication Through Removable Media (15)	Dynamic Resolution (52)	Data Obfuscation (52)	Endpoint Denial of Service (52)	Defacement (52)
Phishing for Information (52)	Establish Accounts (52)	Phishing (52)	Inter-Process Communication (52)	Compromise Client Software Binary (52)	Boot or Logon Initialization Scripts (52)	Boot or Logon Initialization Scripts (52)	Direct Volume Access (15)	Cloud Storage Object Discovery (15)	Clipboard Data (15)	Encrypted Channel (52)	Exfiltration Over Other Network Medium (52)	Financial Theft (52)	Inhibit System Recovery (52)
Search Closed Sources (52)	Obtain Capabilities (52)	Replication Through Removable Media (52)	Native API (52)	Create Account (52)	Execution Guardrails (52)	Execution Guardrails (52)	Input Capture (15)	Container and Resource Discovery (15)	Software Deployment Tools (15)	Data from Configuration Repository (52)	Failback Channels (52)	Firmware Corruption (52)	Network Denial of Service (52)
Search Open Technical Databases (52)	Stage Capabilities (52)	Scheduled Task/Job (25)	Serverless Execution (52)	Create or Modify System Process (14)	File and Directory Permissions Modification (52)	File and Directory Permissions Modification (52)	Multi-Factor Authentication Process (52)	Debugger Evasion (15)	Taint Shared Content (15)	Data from Information Repositories (52)	Ingress Tool Transfer (52)	Resource Hijacking (52)	Service Stop (52)
Search Open Websites/Domains (52)	Valid Accounts (52)	Supply Chain Compromise (52)	Shared Modules (52)	Event Triggered Execution (52)	Domain Policy Modification (52)	Domain Policy Modification (52)	Multi-Factor Authentication Interception (52)	Device Driver Discovery (15)	Use Alternate Authentication Material (15)	Data from Local System (52)	Non-Application Layer Protocol (52)	System Shutdown/Reboot (52)	
Search Victim-Owned Websites (52)		Trusted Relationship (52)	Software Deployment Tools (52)	External Remote Services (52)	Escape to Host (52)	Escape to Host (52)	Multi-Factor Authentication Request Generation (52)	File and Directory Discovery (15)	Group Policy Discovery (15)	Data from Network Shared Drive (52)	Protocol Tunneling (52)		
		Valid Accounts (52)	System Services (12)	Hijack Execution Flow (212)	Event Triggered Execution (52)	Event Triggered Execution (52)	Log Enumeration (15)	Network Service Discovery (15)	Log Enumeration (15)	Non-Standard Port (52)	Transfer Data to Cloud Account (52)		
			User Execution (52)	Implant Internal Image (52)	Exploitation for Privilege Escalation (52)	Exploitation for Privilege Escalation (52)	Network Sniffing (15)	Network Share Discovery (15)	Network Sniffing (15)	Proxy (24)			
			Windows Management Instrumentation (52)	Modify Authentication Process (52)	Hijack Execution Flow (212)	Hijack Execution Flow (212)	OS Credential Dumping (52)	Network Sniffing (15)	OS Credential Dumping (52)	Remote Access Software (52)			
				Office Application Startup (52)	Process Injection (52)	Process Injection (52)	Steal or Forge Authentication Certificates (52)	Peripheral Device Discovery (15)	Steal or Forge Authentication Certificates (52)	Traffic Signaling (52)			
				Power Settings (52)	Scheduled Task/Job (25)	Scheduled Task/Job (25)	Steal or Forge Kerberos Tickets (15)	Permission Groups Discovery (25)	Steal or Forge Kerberos Tickets (15)	Web Service (52)			
				Pre-OS Boot (52)	Valid Accounts (52)	Valid Accounts (52)	Modify Registry (52)	Process Discovery (15)	Process Discovery (15)	Screen Capture (52)			

VirusTotalコマンド

vt-domain-lookup コマンド

VirusTotalでドメインのリストを検索します。

- 入力: テキストファイル
- プロファイル: `all-field-info` と `all-field-info-verbose` 以外すべて
- 出力: CSV

必須オプション:

- `-a, --apiKey <API-KEY>` : VirusTotalのAPIキー
- `-d, --domainList <TXT-FILE>` : ドメイン一覧のテキストファイル
- `-o, --output <CSV-FILE>` : 結果を保存するCSV

任意オプション:

- `-j, --jsonOutput <JSON-FILE>` : VirusTotalからのすべてのJSONレスポンスを出力するJSONファイル
- `-r, --rateLimit <NUMBER>` : 1分間に送るリクエスト数レート制限 (デフォルト: 4)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

vt-domain-lookup コマンドの使用例

はじめに、`list-domains` コマンドでドメイン一覧を作成し、その後 次のコマンドを使用してそれらのドメインを検索します:

```
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 コマンド

VirusTotalでハッシュのリストを検索します。

- 入力: テキストファイル
- プロファイル: `all-field-info` と `all-field-info-verbose` 以外すべて
- 出力: CSV

必須オプション:

- `-a, --apiKey <API-KEY>` : VirusTotalのAPIキー
- `-H, --hashList <HASH-LIST>` : ハッシュ値一覧のテキスト
- `-o, --output <CSV-FILE>` : 結果を保存するCSV

任意オプション:

- `-j, --jsonOutput <JSON-FILE>` : VirusTotalからのすべてのJSONレスポンスを出力するJSONファイル
- `-r, --rateLimit <NUMBER>` : 1分間に送るリクエスト数レート制限 (デフォルト: 4)
- `-q, --quiet` : ログを出力しない (デフォルト: `false`)

vt-hash-lookup コマンドの使用例

```
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 コマンド

VirusTotalでIPアドレスのリストを検索します。

- 入力: テキストファイル
- プロファイル: `all-field-info` と `all-field-info-verbose` 以外すべて
- 出力: CSV

必須オプション:

- `-a, --apiKey <API-KEY>` : VirusTotalのAPIキー

- `-i, --ipList <IP-ADDRESS-LIST>` : IPアドレスのテキストファイル
- `-o, --output <CSV-FILE>` : 結果を保存するCSV

任意オプション:

- `-j, --jsonOutput <JSON-FILE>` : VirusTotalからのすべてのJSONレスポンスを出力するJSONファイル
- `-r, --rateLimit <NUMBER>` : 1分間に送るリクエスト数レート制限 (デフォルト: 4)
- `-q, --quiet` : ログを表示しない (デフォルト: `false`)

vt-ip-lookup コマンドの使用例

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

貢献

どのような形でも構いませんので、ご協力をお願いします。プルリクエスト、ルール作成、evtxログのサンプルなどがベストですが、機能リクエスト、バグの通知なども大歓迎です。

少なくとも、私たちのツールを気に入っていただけたなら、GitHubで星を付けて、あなたのサポートを表明してください。

バグの報告

このプロジェクトは活発なメンテナンスを行っています。見つけたバグを[こちら](#)でご連絡ください。報告されたバグを喜んで修正します！

もし、Hayabusaで何かしらの問題(フォルスポジティブ、バグ、その他)を見つけましたら、[こちら](#) でご連絡ください。

もし、Hayabusaルールで何かしらの問題(フォルスポジティブ、バグ、その他)を見つけましたら、[こちら](#) でご連絡ください。

もし、Sigmaルールで何かしらの問題(フォルスポジティブ、バグ、その他)を見つけましたら、SigmaHQの[こちら](#)でご連絡ください。

ライセンス

Takajōは[GPLv3ライセンス](#)で公開されています。

Twitter

[@SecurityYamato](#)でTakajō`、Hayabusa、ルール更新、その他の大和セキュリティツール等々について情報を提供しています。