Mandiant

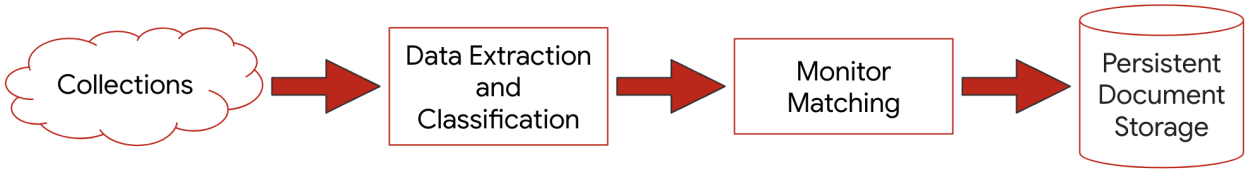
MONITOR MATCHING METHODOLOGY

Monitors in Digital Threat Monitoring (DTM) let you define conditions for searching artifacts (called Documents) collected from the deep and dark web for mentions of designated Topics or Entities. DTM uses a sophisticated process to transform Documents into standardized formats that are more likely to match the queries defined in Monitors and generate Alerts..

For more information about creating Monitors, see **Create DTM Monitors (https://docs.mandiant.com/home/dtm-creating-monitors)**.

Overview

The Monitor matching process to extract meaningful Alerts from Documents involves various data ingestion and transformation pipelines as represented in the following diagram:



1. Documents are collected from various public and private sources from the internet and transformed to a Mandiant-specific data model.
2. Documents are then sent to the Data Extraction and Classification pipeline.
    a. The extraction process attempts to identify any specific Topics found in that Document. It then attempts to classify the Document by applying labels indicating what the Document is likely about.
3. Documents and the extracted and labeled information from the pipeline are sent together to the Monitor Matching pipeline.
    a. The matcher contains a set of Monitors (or queries) that specify what is being looked for in any given Document.
    b. If a Document or any of its extracted data matches a Monitor, an Alert is created. The Alert contains a reference to the Monitor and Document, and information about what caused the Document to match.
4. The set of Alerts is sent to a persistent data store, from which they are displayed in the DTM web console and accessible through the DTM API.

DTM provides an intuitive workflow to create your own custom Monitor as well as customizable templates for common Monitor use cases. This article provides insight into the methodology used to match Monitor conditions and trigger Alerts for monitored entities.

For more information, see **Build Effective Monitors (https://docs.mandiant.com/home/dtm-monitor-scenarios)**.

Terminology

To understand the processes described in this article, it's important to know the terminology used throughout.

DTM Terminology

- **Document**: Source material collected from searches of the deep and dark web. Documents contain the following:
    - Structured data in JSON format from the collected source material.
    - Any extracted Topics from the Document (including people, places, objects, products, and identifiers).

- Any labels applied to the Document (including threat classifications, Document type, and language).

  For a list of Document types, see **DTM Monitor & Research Tools Fields (https://docs.mandiant.com/home/dtm-monitor-fields)**.

- **Monitor**: Structured query based on a set of nested conditions.
  - A structured set of criteria to search for in Documents.
  - Allows any part of the Document to be searched for matches.
  - Conditions can be nested and combined to create complex queries.
  - Full text (keyword) searches and Lucene queries are fully supported.

  For more information, see **Create DTM Monitors (https://docs.mandiant.com/home/dtm-creating-monitors)**.

- **Alert**: An indication that a Monitor matched content found in a Document. Alerts contain the following:
  - A reference to the original Document.
  - Information about which Topics, labels, and text were matched from the Monitor configuration.

  For more information, see **Working with Alerts (https://docs.mandiant.com/home/dtm-alerts)**.

Monitor Matching Terminology

- **Flatten**: With JSON content, this process converts nested objects and arrays into key:value (KV) pairs for easier processing and analysis.
- **Field**: A flattened piece of the original Document to search.
- **Token**: The smallest set of characters or chunk of useful information contained in a field.
- **Tokenize**: The act of reducing a field to its component tokens.
- **Normalize**: Transform data into a standardized structure or format for improved processing.
- **Analyzer**: A process specific to the field being tokenized that reduces a field to its tokens. Analyzers can change the value of the source text to produce a token that more accurately matches its meaning.

Matching process

Matching starts by transforming (or flattening) the original Document into elements called fields. For example, consider the following original Document:

```
{
 "__id": "00000000-0000-0000-0000-000000000000",
 "__type": "message"
 "body": "Hello from the Internet!"
 "channel": {
   "name": "Happy Thoughts"
 }
 "sender": {
   "identity": {
     "name": "John Smith"
   }
 }
}
```

The matching process flattens this JSON into the following set of fields:

```
          __id: 00000000-0000-0000-0000-000000000000
        __type: message
          body: Hello from the Internet!
  channel.name: Happy Thoughts
sender.identity.name: John Smith
```

We then tokenize each field to produce tokens using an Analyzer that is appropriate to that field.

Analyzer approach

The analyzing process helps to ensure that Monitor queries have the best possible chance of matching Documents, even if the original Document doesn't exactly match the query term. In their original form, Documents collected by DTM often contain emojis or unexpected syntax and punctuation. These irregularities are specifically intended to thwart attempts to detect mentions of compromised data on the deep and dark web. Our Analyzers use a sophisticated approach to overcome these efforts, starting with a three-phased process to convert fields into tokens:

1. **Character Filter**: Remove anything from the field that's not useful. Examples could include whitespace, emoji, or punctuation.
2. **Tokenizer**: Split the remaining characters into chunks and then transform those chunks into tokens.
3. **Token Filter**: Remove any tokens that are not ultimately useful. For example, it could be useful to remove common words like articles or conjunctions such as "a" or "the." This phase is where Unicode normalization typically occurs along with case folding (making all characters lowercase).

Simple Analyzer Example

The following is a simple example:

`The  quick brown fox JuMps over the lazy(?) dog.`

The Analyzer process starts with the character filter. The result appears as follows:

`The quick brown fox JuMps over the lazy dog`

The next steps are to tokenize and transform, resulting in the following:

the quick brown fox jumps over the lazy dog

Lastly, we filter the resulting tokens:

quick brown fox jumps over lazy dog

More Complicated Example

This example is closer to what we actually see in the wild:

Ⓣⓗⓔ 🆀🆄🅸🅲🅺 (b)(r)(o)(w)(n) 🦊 j̲u̲m̲p̲s̲ ɹǝʌo the ɭa̲z̲ʏ 🐕.

First we apply the character filter. Given the complexity of this example, the character filter only removes the period at the end:

Ⓣⓗⓔ 🆀🆄🅸🅲🅺 (b)(r)(o)(w)(n) 🦊 j̲u̲m̲p̲s̲ ɹǝʌo the ɭa̲z̲ʏ 🐕

The next step is to tokenize. Note the significant changes resulting from this step:

the quick brown 🦊 jumps revo the lazy 🐕

Finally, we filter the resulting tokens:

Tokenization

Tokenization is the process of splitting text into tokens and transforming them into the smallest bit of useful information possible. This process involves three steps:

1. **Normalization**: Takes the text and transforms it such that semantically identical text gets converted into a common form (for example, becomes QUICK).
2. **Segmentation**: Splits words into tokens based on a set of rules based on the Unicode Standard (see **Unicode TR#29 (https://unicode.org/reports/tr29/)**).
3. **Transformation**: Takes the resulting tokens and makes any final changes that are required for the tokens (such as making all of them lowercase).

Tokenizer Example

Start from our previous example:

Perform Unicode Normalization:

Segment the resulting text into tokens:

Perform lowercase transformation on the tokens:

Analyzer types

There are several types of Analyzers used by DTM, including the following common examples:

- **Full Text**: Splits text into tokens as shown in the previous examples. Normalization and segmentation are performed and common English words are removed from the token stream.
- **Keyword**: Used for text that should only be searchable in its entirety, such as identifiers like hostnames, usernames, or email addresses. This type of Analyzer always produces exactly one token and very little normalization is performed. Values are lower cased.
- **Domain**: Splits domains into word segments (for example, `applebatterystapler.com` becomes `apple|battery|stapler|.|com)`.
- **IP Addresses** (IPv4/IPv6): Produces a single token with a normalized IP address. Used for searching by specific address or CIDR range.
- **Hex**: Used for fields that use hexadecimal values such as MAC addresses.
- **Bin**: Bank Identification Numbers (BINs) used for numeric things like phone numbers and credit card numbers.
- **Hash**: Removes all characters that aren't hash components (for example, `[0 to 9]` and `[a to f]`) for fields containing hashes.
- **Numeric**: Removes all characters that aren't digits (0 to 9) for fields such as phone numbers and credit cards.

Monitor Conditions

Monitors are built using conditions. These conditions can be nested and chained together to produce complicated decision trees that determine whether Alerts should be generated for any given Document. The following is a sample Monitor condition:

```
{
 "topic": "identity_name",
 "operator": "must_equal",
 "match": ["John Smith"]
}
```

Conditions are made up of Topics, Operators, and the specific Match Values to be searched for in the Document.

Monitor Condition Topics

Topics are specific elements in the Document that a Monitor can search. Topics generally fall into one of the following categories:

- Nested conditions (a condition made up of other of conditions)
- Field values from the Document
- Entities (or groups of Entities) extracted from the Document
- Classification labels applied to the Document
- Full Text (Keyword) Searches
- Lucene Queries

For a complete list of Monitor Topics, see **DTM Monitor & Research Tools Fields (https://docs.mandiant.com/home/dtm-monitor-fields)**.

Monitor Condition Operators

Operators determine the circumstances that will cause the Match Values in the condition to match the Topic value from the Document.

For nested and Lucene conditions, there are two operators:

- `any` : Produce an Alert if any of these conditions match.
- `all` : All conditions must match.

For all other conditions, there are four operators and their opposites:

- `must_equal (/must_not_equal)` : The tokens in the match value must exist in the Topic completely and in the same order.
- `must_contain (/must_not_contain)` : The tokens in the match value must exist in the Topic in the same order, but not necessarily completely.
- `must_start_with (/must_not_start_with)` : The Topic value from the Document must start with the Match Value.
- `must_end_with (/must_not_end_with)` : The Topic value from the Document must end with the Match Value token.

Contains versus Equals Operators

Both `contains` and `equals` behave similarly in most circumstances. The primary difference is that `contains` operators can split tokens when determining whether there's a match. For example:

`00000000-0000-0000-0000-DEADBEEF0000` would match `must_contain DEADBEEF` but would not match `must_equal DEADBEEF`.

This scenario also works for common text. For example:

`John Smith` would match `must_contain "hn sm"` but would not match `must_equal "hn sm"`.

The condition `must_equal` requires that any terms be wholly present in a given field *and* directly adjacent to each other for a match to be found and an Alert generated. The difference between this and `must_contain` is that `must_contain` can split terms and doesn't require them to be whole.

Topic Groups

To make it easier to search for similar kinds of entities, there are several Topic Groups that can be searched:

Topic Group	Entities Searched
`group_brand`	`identity_name`, `organization`, `product`, `brand`, `name`, and `batch_name`
`group_identity`	`email`, `identity_name`, `name`, `twitter_handle`, `telegram_user_name`, `phone_number`, and `client_identifier`
`group_network`	`ipv4_address`, `ipv6_address`, `domain`, `mac_address`, and `url`
`group_bin`	`bin`, `bin_foreign`, `bin_partial`
`group_hash`	`sha1`, `sha256`, and `md5`

Searches against Topic Groups will analyze Monitor search terms in the method most appropriate for searching the underlying grouped entities. As a result, these searches can occasionally produce some surprising results, especially with the `group_network` group. For example, the `mac_address` entity type uses the `hash` analyzer, so a search for `group_network must_contain “beefdinner.com”` would generate a match for `mac_address:"be:ef:de:c0:00:00"`.

Keyword Topic

The Keyword Topic is special because it performs a full text search over all tokens found in all fields, labels, and entities from the Document.

This Topic is useful for performing searches looking for very specific keyword terms. These types of searches can produce many results if the match conditions aren’t specific enough.

Each match condition for a Keyword Topic is automatically treated as a phrase in case multiple tokens are produced when analyzing the match term.

Keyword Topic values do not support quoting, escaping, or wildcards.

Lucene Topic

Apache Lucene is the accepted standard in the text searching technology space and is relatively well known. On the surface, the syntax looks simple, but using it correctly often requires careful consideration.

For more information about using Lucene queries in DTM, see **Lucene Queries in DTM (https://docs.mandiant.com/home/dtm-lucene-queries)**.