VB2020 localhost

30 September - 2 October, 2020 / vblocalhost.com

STANDARDIZED REPORTING WITH THE MALWARE BEHAVIOR CATALOG

Desiree Beck

The MITRE Corporation, USA

dbeck@mitre.org

www.virusbulletin.com

ABSTRACT

The *Malware Behavior Catalog* (*MBC*) is a publicly available catalog of malware objectives and behaviors, developed to support malware analysis-oriented use cases, such as labeling, similarity analysis, and standardized reporting. MBC content is available on *GitHub* [1].

INTRODUCTION

While there is no formal relationship between MBC and the *MITRE ATT&CK*® knowledgebase[1], adversary behaviors and malware behaviors overlap because adversaries often use malware as a means of carrying out their attacks. Consequently, ATT&CK is used in reporting to capture malware behaviors. For example, several commercially available sandboxes map behaviors to ATT&CK techniques.

Many malware behaviors can be mapped directly into ATT&CK. For example, the behavior 'starts cmd.exe for commands execution' maps to ATT&CK's 'Command-Line Interface' technique. However, ATT&CK is oriented toward adversaries, not malware, and it focuses on behaviors identified during an intrusion, rather than those discovered during analyses, as is done for malware. As a result, some behaviors unique to malware cannot be mapped to ATT&CK. The need to define these behaviors motivated the development of MBC.

MBC *expands* upon ATT&CK by defining behaviors that support malware analysis use cases, and it also *restricts* ATT&CK content by only referencing ATT&CK techniques that are applicable to malware.

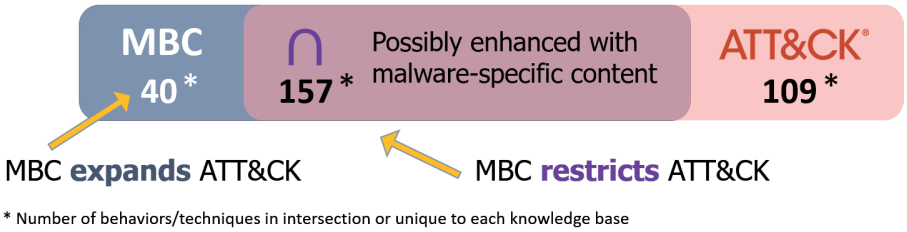

Figure 1: MBC's relationship to ATT&CK.

As shown in Figure 1, MBC currently defines 40 new behaviors and leverages (possibly enhancing) 157 ATT&CK techniques. One hundred and nine (109) ATT&CK techniques are not used in MBC.

MBC draws upon ATT&CK's success by applying its philosophy and methodology to malware. Namely, MBC maintains a malware analysis-oriented perspective; focuses on real-world use of behaviors through empirical and publicly documented malware examples; and sustains a level of abstraction appropriate for supporting malware analysis use cases.

MBC CONCEPTS

To best reflect terms used in malware analysis, MBC uses 'objective' (instead of ATT&CK's use of 'tactic') and 'behavior' (instead of ATT&CK's use of 'technique'). These and other MBC concepts are discussed below.

Objectives

An MBC *objective* captures 'why' malware does something. As shown in Table 1, 13 objectives are defined for MBC. ANTI-BEHAVIORAL ANALYSIS and ANTI-STATIC ANALYSIS are specific to malware and are not defined in ATT&CK. The other 11 are based on ATT&CK tactics, where their definitions have been tailored for malware analysis use cases.

Behaviors

An MBC *behavior* captures 'what' malware does to achieve an objective. MBC aims to directly and explicitly define malware behaviors and code characteristics to support malware analysis-oriented use cases. MBC references existing ATT&CK techniques, when possible, and defines its own set of new, malware-focused behaviors. MBC behaviors are defined in one of four ways:

- ATT&CK technique reference: MBC provides a short description and a link to an existing ATT&CK technique. MBC is intended to be used in combination with ATT&CK; by design, ATT&CK content is not duplicated. For example, ATT&CK's 'Audio Capture' technique is simply referenced.
- Enhanced ATT&CK technique: MBC extends ATT&CK's description to cover specific aspects of malware. For example, ATT&CK's 'Execution Guardrails' technique is enhanced to include details on how malware may use environmental conditions to constrain execution.

[1] *ATT&CK* is a curated knowledgebase and model for cyber adversary behavior, widely used to capture various aspects of the adversary's lifecycle [2].

Objective	Description
ANTI-BEHAVIORAL ANALYSIS	Malware aims to prevent, obstruct, or evade behavioral analysis done in a sandbox, debugger, etc.
ANTI-STATIC ANALYSIS	Malware aims to prevent static analysis or make it more difficult. Simpler static analysis identifies features such as embedded strings, executable header information, hash values, and file metadata. More involved static analysis involves disassembly of the binary code.
COLLECTION	Malware aims to identify and gather information, such as sensitive files, from a target network prior to exfiltration. This objective includes locations on a system or network where the malware may look for information to exfiltrate.
COMMAND AND CONTROL	Malware aims to communicate (receive and/or execute remotely submitted commands) with controlling or controlled systems within a target network (C2 servers, bots, etc.).
CREDENTIAL ACCESS	Malware aims to obtain credential access, allowing it or its underlying threat actor to assume control of an account, with the associated system and network permissions.
DEFENSE EVASION	Malware aims to evade detection or avoid other cybersecurity defenses.
DISCOVERY	Malware aims to gain knowledge about the system and internal network.
EXECUTION	Malware aims to execute its code on a system to achieve a variety of goals.
EXFILTRATION	Malware aims to steal data from the system on which it executes. This includes stored data (e.g. files) as well as data input into applications (e.g. web browser).
IMPACT	Malware aims to achieve its mission of manipulating, interrupting, or destroying systems and data.
LATERAL MOVEMENT	Malware aims to propagate through the infection of a system or is able to infect a file after executing on a system. The malware may infect actively (e.g. gain access to a machine directly) or passively (e.g. send malicious email).
PERSISTENCE	Malware aims to remain on a system regardless of system events.
PRIVILEGE ESCALATION	Malware aims to obtain a higher level of privilege for execution.

Table 1: MBC objectives.

- Refined ATT&CK technique: when an ATT&CK technique is too broad for malware analysis use cases, multiple MBC behaviors are defined. For example, ATT&CK's 'Software Packing' technique is broken down into *two* MBC behaviors: 'Software Packing' and 'Executable Code Obfuscation'.
- MBC-only behavior: new behaviors are defined as needed to support malware analysis use cases. The best examples are anti-analysis behaviors, such as 'Debugger Detection' and 'Dynamic Analysis Evasion'.

Example of content captured for MBC behaviors is shown in Figure 2.

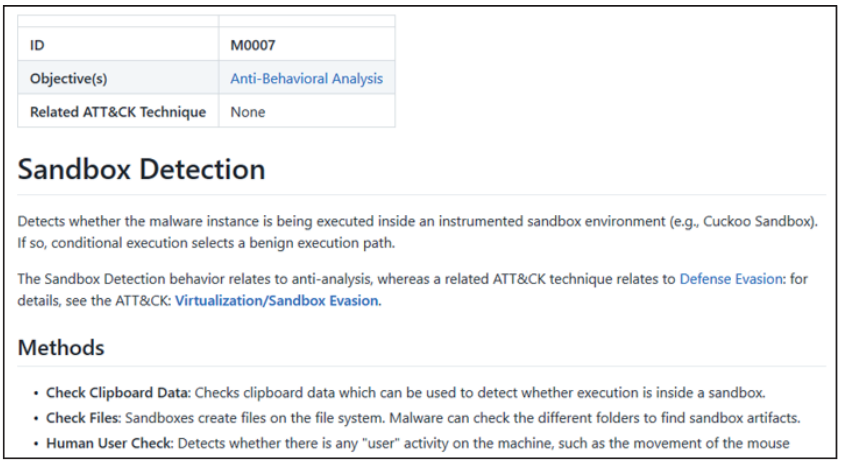

Figure 2: Content of MBC behavior 'Sandbox Detection'.

Names of MBC behaviors may or may not match the names of related ATT&CK techniques. Any content provided on behavior pages is *supplemental* to ATT&CK content. In other words, ATT&CK content is not duplicated in MBC, and MBC users are expected to reference ATT&CK while capturing malware behaviors.

Methods

Methods are associated with behaviors and serve different roles, depending on the behavior. In some cases, a method further refines a behavior; in other cases, a method is an implementation of a behavior. Method descriptions are included on behavior pages. For example, methods defined for the 'Debugger Detection' behavior include 'API Hook Detection', 'CheckRemoteDebuggerPresent', and 'CloseHandle'. A method cannot be used without a behavior.

Previously, methods had no ATT&CK counterpart, but beginning in April 2020, ATT&CK defines sub-techniques, which are similar to methods.

Micro-behaviors

Some malware behaviors are low-level, support many objectives and other behaviors, and are not malicious in and of themselves. Because they are often identified during malware analysis, they are captured in MBC. They are called micro-behaviors. Examples include 'HTTP Communication' and 'Inter-process Communication'; method examples for these micro-behaviors include 'GET Request' and 'Create Pipe', respectively.

Identifiers

As shown in Table 2, the beginning letter of an identifier relays information about the behavior.

Letter	Example	Description
T	*T1234*	Behavior is a stub that references an ATT&CK technique.
E	*E1234*	Behavior enhances an ATT&CK technique with malware-specific details.
M	*M1234*	Behavior is newly defined in MBC.
X	*X1234*	Behavior is an MBC micro-behavior.

Table 2: MBC identifiers.

When two or more MBC behaviors refine the same ATT&CK technique, the MBC behavior by the same name (if there is one) keeps the number portion of the ATT&CK identifier, and MBC behaviors with different names are given MBC identifiers. For example, the MBC behaviors 'Software Packing' [E1045] and 'Executable Code Obfuscation' [M0032] refine the ATT&CK technique 'Software Packing' [T1045].

When a new ATT&CK technique is defined *after* an equivalent MBC behavior is defined, the pre-existing MBC identifier is preserved and the new ATT&CK identifier is referenced. For example, the MBC behavior 'Virtual Machine Detection' [M0009] references the ATT&CK technique 'Virtualization/Sandbox Evasion' [T1497].

Method identifiers: If MBC defines a new method for an existing ATT&CK technique, an 'm' is included. For example, a new method defined on T1234 would be denoted T1234.m01 and is different from T1234.001. Identifiers of methods on newly defined MBC behaviors do not use an 'm' (e.g. M0008.009).

Canonical representation

The canonical representation for MBC content is OBJECTIVE::Behavior::Method. For example:

ANTI-BEHAVIORAL ANALYSIS::Debugger Detection::Process Environment Block.

Objectives and behaviors can be used together or independently, but a method *must* be associated with a behavior.

STANDARDIZED REPORTING

Standardized reporting enables consistent interpretation of behavior analysis data to improve detection, mitigation and remediation. We begin by showing how MBC can be expressed in STIX 2.1 format and then show how behaviors identified through analysis with *Cuckoo Sandbox* can be mapped to MBC.

MBC in STIX 2.1 format

Structured Threat Information Expression (STIX) is a language and serialization format used to exchange cyber threat intelligence (CTI). Readers not familiar with STIX should refer to [3]. STIX enables organizations to share CTI, including malware analysis information, in a consistent and machine-readable manner.

ATT&CK is available expressed in JSON-based STIX 2.0, making it machine-readable and accessible [4]. Similarly, MBC content is available expressed in a JSON-based STIX 2.1 format [5]. Table 3 maps MBC concepts to STIX 2.1 objects where the STIX object types are literal strings captured in the **`type`** property of the STIX object.

MBC concept	STIX object type	Notes
Objective	x-mitre-tactic	MBC objectives are captured using a custom object of type x-mitre-tactic, which was defined to capture ATT&CK tactics. Using it instead of defining a new 'x-mitre-objective' object enables ATT&CK users to more easily use MBC.
Behavior	attack-pattern	MBC behaviors (like ATT&CK techniques) are captured using the STIX Attack Pattern object. MBC micro-behaviors are also captured with Attack Pattern objects.
Malware	malware	MBC malware examples, which are associated with MBC behaviors are captured using the STIX Malware object.

Table 3: MBC concepts as STIX objects.

Capturing objectives

As shown in Table 4, the properties of an MBC objective are captured in a custom STIX object of type x-mitre-tactic. An example STIX 2.1 object is given after the table (strings have been snipped for brevity).

Note that the STIX `external_references` property is used to capture both the MBC objective identifier and external references by setting *external_references.source_name* to either 'mitre-mbc' or 'external_source', respectively. The STIX `x_mitre_shortname` custom property (defined and used by ATT&CK) will be set to a lowercase, hyphenated version of the MBC objective name.

MBC property	STIX object property
Name	**`name`**
---	**`x_mitre_shortname`**
Identifier	**`external_references`** property where *external_references.source_name* == "mitre-mbc" *external_references.external_id* == MBC identifier *external_references.url* == URL for the MBC objective (GitHub page)
Description	**`description`**
External reference(s)	**`external_references`** property where *external_references.source_name* == "external_source" *external_references.description* == reference description *external_references.url* == URL for the reference

Table 4: MBC objective as STIX object of type x-mitre-tactic.

Example:

```
{
    "type": "x-mitre-tactic",
    "spec_version": "2.1",
    "id": "x-mitre-tactic--eb6166b0-f3c9-4124-aeb9-662941baa19e",
    "created": "2020-02-05T20:28:15.061Z",
    "modified": "2020-02-05T20:28:15.061Z",
    "name": "Anti-Behavioral Analysis",
```

```
    "x_mitre_shortname": "anti-behavioral-analysis",
    "description": "Behaviors that prevent, obstruct, or evade <snip>",
    "external_references": [
        {
            "source_name": "external_source",
            "description": "Unprotect Project, a database about <snip>",
            "url": "http://unprotect.tdgt.org/index.php/Unprotect_Project"
        },
        {
            "source_name": "external_source",
            "description": "InDepthUnpacking, course content for <snip>",
            "url": "https://github.com/knowmalware/InDepthUnpacking"
        },
        {
            "source_name": "mitre-mbc",
            "url": "https://github.com/MBCProject/mbc-markdown/blob/<snip>",
            "external_id": "M9001"
        }
    ]
}
```


Capturing behaviors

The properties of an MBC behavior are captured in a STIX Attack Pattern object (attack-pattern), as shown in Table 5. An example STIX 2.1 object is given after the table (strings have been snipped for brevity).

MBC property	STIX object property
Behavior name	**`name`**
Identifier	**`external_references`** property where *external_references.source_name* == "mitre-mbc" *external_references.external_id* == MBC identifier *external_references.url* == URL for the MBC behavior (GitHub page)
Description	**`description`**
Associated MBC Objective(s)	**`kill_chain_phases`** property where *kill_chain_phases.kill_chain_name* == "mitre-mbc" *kill_chain_phases.phase_name* == MBC objective name
Related ATT&CK technique(s)	**`external_references`** property where *external_references.source_name* == "mitre-attack" *external_references.external_id* == ATT&CK identifier *external_references.url* == URL for the ATT&CK technique
Method(s)	**`x_mitre_methods`** custom property where *x_mitre_methods.name* == method name *x_mitre_methods.description* == description of the method
External reference(s)	**`external_references`** property where *external_references.source_name* == "external_source" *external_references.description* == reference description *external_references.url* == URL of the reference

Table 5: MBC behavior as STIX Attack Pattern object.

The following example STIX 2.1 Attack Pattern object captures the 'Sandbox Detection' behavior. For brevity, strings have been snipped. There are no related ATT&CK techniques for this MBC behavior.


```
{
    "type": "attack-pattern",
    "spec_version": "2.1",
    "id": "attack-pattern--43f3997b-88e5-4a2e-9b59-6af0892a89b2",
```

```
    "created": "2020-02-05T20:28:15.192Z",
    "modified": "2020-02-05T20:28:15.192Z",
    "name": "Sandbox Detection",
    "description": "Detects whether the malware instance is being <snip>",
    "kill_chain_phases": [{
        "kill_chain_name": "mitre-mbc",
        "phase_name": "anti-behavioral-analysis"
    }],
    "external_references": [
        {
            "source_name": "mitre-mbc",
            "url": "https://github.com/MBCProject/mbc-markdown/ <snip>",
            "external_id": "M0007"
        },
        {
            "source_name": "external_source",
            "url": "https://www.fireeye.com/blog/threat-research/ <snip>"
        },
        {
            "source_name": "external_source",
            "url": "http://labs.lastline.com/exposing-rombertik <snip>"
        },
        {
            "source_name": "external_source",
            "url": "https://github.com/LordNoteworthy/al-khaser"
        }
    ],
    "x_mitre_methods": [
        {
            "definition": "Checks clipboard data which can be used <snip>",
            "name": "Check Clipboard Data"
        },
        {
            "definition": "Sandboxes create files on the file <snip>",
            "name": "Check Files"
        }
    ]
}
```


Mapping Cuckoo community signatures to MBC

In early 2020, the MBC team mapped *Cuckoo* community signatures (developed for *Cuckoo Sandbox*) into MBC [6, 7]. Of the more than 560 signatures defined in the community repository, approximately 275 were appropriate for mapping into MBC (the others are anti-virus-related signatures that identify specific malware families and instances).

Approximately 140 of the signatures were already mapped into ATT&CK. We added new signatures, which was possible because MBC includes malware-related behaviors. We also used MBC's malware-focused content to revise 80 of the existing ATT&CK mappings for better accuracy.

The example below shows that the signature 'antisandbox_sleep.py' was mapped to the MBC 'Sandbox Detection' [M0007] behavior:

```
from lib.cuckoo.common.abstracts import Signature
class AntiSandboxFile(Signature):
    name = "antisandbox_file"
    description = "Looks for known filepaths where sandboxes execute samples."
    severity = 3
    categories = ["anti-sandbox"]
    authors = ["Cuckoo Technologies"]
    minimum = "2.0"
    ttp = ["M0007"]
    ...
```

A *Cuckoo Sandbox* report would then reference the behavior in its report, as shown below:


```
{
  "signatures": [{
    "families": [],
```

```
    "description": "Looks for known filepaths where sandboxes execute samples.",
    "severity": 3,
    "ttp": {"M0007": {
        "short": "Sandbox Detection",
        "long": "Detects whether the malware instance is being <snip>"
    }},
    "markcount": 1,
    "references": "...",
    "marks": "...",
    "name": "antisandbox_file"
  }]
}
```


Example STIX 2.1 output

The example below shows malware analysis content captured in STIX 2.1 format (hash values are fabricated). Whether the behaviors are identified by *Cuckoo Sandbox* or another analysis tool, the STIX objects and relationships indicating MBC behaviors would be similar. For brevity, not all STIX objects are shown.


```
{
    "type": "bundle",
    "id": "bundle--94fdc239-0321-0544-0345-bdef253ac340",
    "objects": [
        {
            "type": "malware",
            "spec_version": "2.1",
            "id": "malware--8473bd84-03bb-47ff-a024-26de7e891bf1",
            "created": "2020-05-28T23:27:49.511Z",
            "modified": "2020-05-28T23:27:49.511Z",
            "malware_types": ["unknown"],
            "is_family": false,
            "sample_refs": ["file--35462d98-0234-c319-2c43-237dfe538564"]
        },
        {
            "type": "file",
            "id": "file--35462d98-0234-c319-2c43-237dfe538564",
            "spec_version": "2.1",
            "size": 96536,
            "name": "sample.exe",
            "hashes": {
              "MD5": "fed05678321dcfed98bf019fbbe3409c",
              "SHA-1": "6634de093451ff6cf441770d1b2b1253556c8d18"
            }
        },
        {
            "type": "malware-analysis",
            "spec_version": "2.1",
            "id": "malware-analysis--f7ed7453-5542-c531-6566-bdefa5450123",
            "created": "2020-05-28T23:54:45.000Z",
            "modified": "2020-05-28T23:54:45.000Z",
            "product": "cuckoo",
            "version": "2.0.7",
            "result": "suspicious",
            "sample_ref": "file--35462d98-0234-c319-2c43-237dfe538564"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--35462d99-0872-adfe-1288-65748cbd9433",
            "created": "2020-05-29T00:27:49.516Z",
            "modified": "2020-05-29T00:27:49.516Z",
            "relationship_type": "dynamic-analysis-of",
            "source_ref": "malware-analysis--f7ed7453-5542-c531-6566-bdefa5450123",
            "target_ref": "malware--8473bd84-03bb-47ff-a024-26de7e891bf1"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
```

```
            "id": "relationship--5342743f-034d-02bc-3829-1023543377d3",
            "created": "2020-05-29T01:01:46.369Z",
            "modified": "2020-05-29T01:01:46.369Z",
            "relationship_type": "derived-from",
            "source_ref": "attack-pattern--43f3997b-88e5-4a2e-9b59-6af0892a89b2",
            "target_ref": "malware-analysis--f7ed7453-5542-c531-6566-bdefa5450123"
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--84750981-cdfe-4637-2022-bd7af2534712",
            "created": "2020-05-29T01:01:46.370Z",
            "modified": "2020-05-29T01:01:46.370Z",
            "relationship_type": "uses",
            "source_ref": "malware--8473bd84-03bb-47ff-a024-26de7e891bf1",
            "target_ref": "attack-pattern--43f3997b-88e5-4a2e-9b59-6af0892a89b2"
        }
    ]
}
```


The relationship--5342743f-034d-02bc-3829-1023543377d3 indicates that the MBC 'Sandbox Detection' behavior (attack-pattern--43f3997b-88e5-4a2e-9b59-6af0892a89b2) was 'derived from' the *Cuckoo Sandbox* analysis. Also, relationship--84750981-cdfe-4637-2022-bd7af2534712 indicates that the malware 'uses' the 'Sandbox Detection' behavior.

FUTURE WORK

MBC content will continue to be refined to improve support for malware analysis use cases.

- As ATT&CK evolves to include sub-techniques and a modified set of techniques, MBC will be updated.
- MBC micro-behaviors will continue to be defined.
- The mapping between MBC and the *Cuckoo* community signatures will be updated, as new *Cuckoo* signatures are defined and as MBC content evolves.

Please see the MBC Project on *GitHub* for more information [1]. To join the MBC mailing list, please send a request to mbc@mitre.org.

REFERENCES

[1] Malware Behavior Catalog. https://github.com/MBCProject. Accessed 05/27/2020.

[2] MITRE ATT&CK®. https://attack.mitre.org. Accessed 05/18/2020.

[3] STIX Version 2.1, Committee Specification Draft 04 / Public Review Draft 03. 20 February 2020. https://docs.oasis-open.org/cti/stix/v2.1/csprd03/stix-v2.1-csprd03.pdf. Accessed 05/27/2020.

[4] ATT&CK expressed in STIX 2.0. https://github.com/mitre/cti. Accessed 5/31/2020.

[5] MBC expressed in STIX 2.1. https://github.com/MBCProject/mbc-stix2. Accessed 05/27/2020.

[6] MBC fork of cuckoosandbox/community. https://github.com/MBCProject/community. Accessed 05/27/2020.

[7] MBC fork of cuckoosandbox/cuckoo. https://github.com/MBCProject/cuckoo. Accessed 05/27/2020.