MMDEF-B v1.0 README/FAQ

Last Updated: 8/27/2013

Contents

What is MMDEF-B v1.0?

The Malware Metadata Exchange Format Behavioral (MMDEF-B) version 1.0 is a format for capturing some basic information about the behavior of a single malware instance. With a focus on simplicity rather than on expressivity and completeness, this work aims to facilitate the capture of simple behavioral information about malware. For example, such data can be used for the following purposes:

- Faster triage of malware instances, by providing a useful (but not comprehensive) glimpse into an instance's behavior to allow analysts to decide whether it's something that should be investigated for further analysis.
- A simple way of capturing the indicators which represent the minimum set of artifacts that signify the presence of the malware instance on a system.
- Transmission and exchange of the basic behavioral components of a malware instance, for helping organizations respond more quickly to malware outbreaks.

What's the relationship between MMDEF and MMDEF-B?

In terms of origin, both Malware Metadata Exchange Formats are the product of the IEEE's Industry Connections Security Group (ICSG) MMDEF working group. However, the original MMDEF format[1], at version 1.2 at the time of this writing, was produced first and intended to augment AV sample file sharing with the ability to supply additional metadata such as hashes, file names, and other useful entities. MMDEF-B, on the other hand, is a new format that focuses exclusively on characterizing, in a simplistic fashion, the behaviors of a malware instance.

Besides some overlap in necessary context (e.g., the use of hashes for the identification of a malware instance), MMDEF and MMDEF-B do not share a common data model and thus are not compatible with each other. This was done intentionally, since we feel that the two formats are intended to support different sets of use cases. Accordingly, the two formats will be developed and versioned independently, though there is a possibility that the two will be coupled via the import and usage of MMDEF-B in MMDEF (in a future version), if the community feels it worthwhile.

Data Model

The MMDEF-B 1.0 data model consists of two primary components:

A) The characterization of the malware instance, or subject, whose behavior is being captured.

B) A limited subset of observed behavior for the subject that provides *some* indication of what it's doing on the host or network. Again, this is not intended to capture a complete representation of the behavior of the malware instance, but instead is designed to capture some useful data that can be simply represented and similarly consumed. We refer to each atomic behavioral component, such as the creation of a file, as an action.

For the purposes of creating a simplistic but useful format, we have identified a subset of Windows-oriented actions that we should capture, based on the type of system entity that they operate on. These were chosen based on their utility in terms of capturing useful malware behavior, their ability to be used as simple indicators, and their ability to be expressed simply and succinctly. For each such class of

[1] http://grouper.ieee.org/groups/malware/malwg/Schema1.2/

actions, we have defined a minimum set of attributes used to describe the class (e.g. file names, registry key/hives, etc.) along with the set of actions that we wish to be able to express as operating on the class. This data is summarized in Table 1 below.

Action Class	Attributes	Possible Actions
File	count, pid, name, normalized_path, size_in_bytes, old_name, new_name, md5, sha1, sha256, sha3	create, delete, read, write, rename
Registry	count, pid, root_key, key, value_name, value_data	create, delete, read, write
Mutex	count, pid, name	create, delete
Process	count, pid, image_filename, image_file_normalized_path, target_pid	create, kill, write
Network	count, pid, url, ipv4, ipv6, final_url, port	connect, listen

Table 1: MMDEF-B v1.0 Action Classes, Attributes, and Corresponding Actions

Using this set, each action that we wish to capture can be expressed as the following notional tuple:

```
<class, action, attributes>
```

Due to the size of this set, we can explicitly define each attribute and also what each action means in the context of its class. This is defined in the appendices below.

Data Model Implementation

The current data model is expressed through an XML Schema (mmdefb-1-0-schema.xsd). A sample output can be seen below in Figure 1:

```
<mmdefb
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="mmdef-v2"
 xsi:schemaLocation="mmdefb-v1 mmdefb-1-0-schema.xsd">
   <subject md5="35ed51749a8987b8dcda050647f6c8d7" size_in_bytes="18087"/>
   <action_findings>
      <file pid="352" action="create" name="i1ru74n4.exe" normalized_path="csidl_system"/>
      <registry_key pid="352" action="write" hive="HKEY_LOCAL_MACHINE" key="Software\Microsoft\
Windows\CurrentVersion\Explorer\Shell Folders" value_name="Common Desktop" value_data="C:\
Documents and Settings\All Users\Desktop"/>
      <process pid="440" parent_pid="352" action="create" filename="i1ru74n4.exe"/>
      <mutex pid="440" action="create" name="CTF.Asm.MutexDefaultS-1-5-21-1229272821-
1004336348-527237240-1003"/>
   </action_findings>
</mmdefb>
```

Figure 1: MMDEF-B v1.0 Output Example (XML)

While we feel that this implementation is lightweight and human-readable, we may look at other implementations, particularly JSON, in the future. However, creating the implementation as an XML schema provides built-in data validation that can be more difficult to do with other implementations.

FAQ

Q: How is MMDEF-B v1.0 data intended to be created? What types of tools would output such data?
A: We intentionally leave it open as to how MMDEF-B v1.0 instances are created and where the data for them is sourced. However, we envision that the most typical usage would be for an MMDEF-B v1.0 instance to be created from the output of a dynamic malware analysis tool, or sandbox.

Q: Why have we limited the actions and their corresponding attributes to such a small set?
A: For MMDEF-B, we've focused on creating a reasonable balance between simplicity and expressivity in terms of malware actions. From the input we've received from malware analysts and the broader anti-malware community, we feel that the set we have captures a useful subset of malware behavior that can be used for ascertaining whether further triage is needed and for the construction of indicators.

Q: Why are mutex actions broken out into their own class? Shouldn't they be considered in the context of a process?
A: While we agree that mutex objects are created and used by processes, they are often a useful indicator of malware infection on their own, and we therefore felt it worthwhile to break them out into their own class.

Q: What kind of dependencies is the Dependency element in the Subject intended to capture?
A: The Dependency element in the Subject is intended to capture data or configuration file dependencies for the malware instance being characterized. It is not intended to capture entities like installed software or other programs that the malware may need to run.

Q: Why can an MMDEF-B instance characterize only a single malicious binary and its behavior?
A: This was done for two primary reasons: simplicity, and to help achieve consistent content production and consumption. Following the established focus on simplicity for this effort, we felt that allowing for the characterization of a single malware binary in an instance document was simpler in terms of an implementation standpoint, as it requires less XML schema to achieve. This also gets us away from implying any implicit relationships (intentional or not) between the malware instances in a single document. Accordingly, this makes it easier to consume and produce content, since there is the knowledge that an MMDEF-B instance will always contain a single malware characterization.

Q: Are there any best practices for the creation of MMDEF-B instances?
A: Since an MMDEF-B instance characterizes a single malicious binary, we recommend the following naming convention for instance documents: the MD5 hash of the binary (if available) appended with '_mmdefb.xml'; for example: '3d23ec8b55840b95ea75197ce9446b6d_mmdefb.xml'. If the MD5 is not available, we recommend using the following hashes, in the order of availability: SHA1, SHA256, and SHA3. Also, we highly recommend using the XML schema at <insert URL here> to validate each instance document upon creation.

Q: What is the difference between MMDEF-B and Mandiant's OpenIOC?
A: MMDEF-B and OpenIOC differ greatly both in use case and scope: OpenIOC is intended to function as a language for the creation of malware indicators, and includes support for many more attributes than does MMDEF-B. On the other hand, MMDEF-B is focused primarily on capturing a simple set of behaviors for a malware instance, some of which can also double as indicators for the instance. Accordingly, MMDEF-B can also capture background noise as part of a malware instance's execution

that may not suitable for use as an indicator, whereas OpenIOC is intended to capture and convey analyst-derived data that serves exclusively as indicators for the malware instance.

Q: What is the difference between MMDEF-B and MITRE's MAEC?
A: There is some overlap in terms of context and capabilities of MMDEF-B and MITRE's Malware Attribute Enumeration and Characterization (MAEC) format, particularly in terms of the behavioral actions that both can capture. However, MAEC is intended to be a more fully-fledged language for the capture of malware attributes, and thus can represent many more actions and attribute types for a malware instance, at the expense of added complexity and verbosity. Thus, MMDEF-B can be considered a simplified subset, in terms of verbosity and complexity, of the behavioral characterization components of MAEC.

Q: Is there a broad comparison between the supported action classes of MMDEF-B, OpenIOC, and MAEC?
A:

Action Class (Windows)	**Support in Effort***		
	MMDEF-B	OpenIOC	MAEC
ARP Cache		**x**	
Disk/Volume		**x**	**x**
DNS Cache		**x**	**x**
Driver		**x**	**x**
File	**x**	**x**	**x**
GUI			**x**
Hooking		**x**	**x**
Inter-process Communication			**x**
Library/Module		**x**	**x**
Network (general)	**x**		**x**
Network (DNS)			**x**
Network (FTP)			**x**
Network (HTTP)			**x**
Network (IRC)			**x**
Network Route Entry		**x**	**x**
Network Share			**x**
Process (general)	**x**	**x**	**x**
Process (memory)		**x**	**x**
Process (thread)			**x**
Registry	**x**	**x**	**x**
Service		**x**	**x**
Socket			**x**
Synchronization	**x**		**x**
System Restore		**x**	**x**
System		**x**	**x**
Task Scheduler		**x**	**x**
URL History		**x**	
User		**x**	**x**

* = Support of *any* kind, however minimal

Appendix A: Subject Attributes

The subject represents the malware instance file being characterized, with the corresponding set of possible attributes.

Attribute	Definition	Datatype[2]	Example
name	The name of the file, including file extension but excluding any pre-pended path.	string	some_binary.exe
size_in_bytes	The size of the file, in bytes.	pos. integer	25563
md5	The MD5 hash of the file, as a hex string.	hex binary	(omitted)
sha1	The SHA1 hash of the file, as a hex string.	hex binary	(omitted)
sha256	The SHA256 hash of the file, as a hex string.	hex binary	(omitted)
sha3	The SHA3 hash of the file, as a hex string.	hex binary	(omitted)
analysis_run_length	For dynamic analysis, the length of time that the file was analyzed, in milliseconds.	pos. integer	60000
exit_code	The exit code recorded for the process spawned from the malware instance. There are two possible values: '0', which indicates that the malware process terminated itself, while a value of '-1' indicates that the malware process was terminated by some external means, such as the dynamic analysis environment after running for a set length of time.	Integer	0
dependency (0-n)	Any data or configuration files that the malware instance file may be dependent on in order to run.	(mmdefb) dependencyFileType	(omitted)

[2] Unless otherwise noted, the XML Schema datatype. See http://www.xml.dvint.com/docs/SchemaDataTypesQR-2.pdf for more information.

Appendix B: File Action Types and Attributes

Action Type	Definition	Possible Attributes
Create	The subject creates a new file on the system.	count, pid, name, normalized_path, size_in_bytes, md5, sha1, sha256, sha3
Delete	The subject deletes an existing file on the system.	count, pid, name, normalized_path, size_in_bytes, md5, sha1, sha256, sha3
Read	The subject reads some data from an existing file on the system.	count, pid, name, normalized_path, size_in_bytes, md5, sha1, sha256, sha3
Write	The subject writes some data to an existing file on the system.	count, pid, name, normalized_path, size_in_bytes, md5, sha1, sha256, sha3
Rename	The subject renames an existing file on the system.	count, pid, new_name, old_name, size_in_bytes, md5, sha1, sha256, sha3

Attribute	Definition	Datatype	Example
count	The number of times the action occurred, if it occurred more than once.	pos. integer	5
pid	The process ID (PID) of the process which initiated the file action.	pos. integer	135
name	The name of the file targeted by the action, including file extension but excluding any pre-pended path.	string	some_binary.exe
normalized_path	The normalized path to the file targeted by the action, using the appropriate pre-pended CSIDL constant[3]. E.g., for "C:\Windows", on Windows 7 it would be "csidl_windows". If no CSIDL constant is applicable, then the fully qualified path to the file.	string	csidl_windows\test
size_in_bytes	The size of the file targeted by the action, in bytes.	pos. integer	25563
old_name	For the rename action, the original name of the file targeted by the action before it was renamed.	string	old_name.exe
new_name	For the rename action, the new name of the file targeted by the action after it was renamed.	string	new_name.dll
md5	The MD5 hash of the file targeted by the action.	hex binary	(omitted)
sha1	The SHA1 hash of the file targeted by the action.	hex binary	(omitted)
sha256	The SHA256 hash of the file targeted by the action.	hex binary	(omitted)
sha3	The SHA3 hash of the file targeted by the action.	hex binary	(omitted)

[3] See http://msdn.microsoft.com/en-us/library/windows/desktop/bb762494(v=vs.85).aspx for a full listing and their definitions

Appendix C: Registry Action Types and Attributes

Action Type	Definition	Possible Attributes
Create	The subject creates a new registry key or a new value under an existing registry key.	count, pid, root_key, key, value_name
Delete	The subject deletes an existing registry key or an existing value under an existing registry key.	count, pid, root_key, key, value_name
Read	The subject reads some data from an existing value under an existing registry key.	count, pid, root_key, key, value_name, value_data
Write	The subject writes some data to an existing value under an existing registry key.	count, pid, root_key, key, value_name, value_data

Attribute	Definition	Datatype	Example
count	The number of times the action occurred, if it occurred more than once.	pos. integer	5
pid	The process ID (PID) of the process which initiated the registry action.	pos. integer	135
root_key	The abbreviated root key of the registry key targeted by the action. Abbreviations are: HKLM for HKEY_LOCAL_MACHINE, HKCC for HKEY_CURRENT_CONFIG, HKCR for HKEY_CLASSES_ROOT, HKCU for HKEY_CURRENT_USER, HKU for HKEY_USERS.	string	HKLM
key	The full registry key targeted by the action, not including the root key or value.	string	SYSTEM\ControlSet001\Control
value_name	The name of a registry value targeted by the action.	string	BootDriverFlags
value_data	The data contained in the registry value targeted by the action.	string	0

Appendix D: Mutex Action Types and Attributes

Action Type	Definition	Possible Attributes
Create	The subject creates a new named global mutex.	count, pid, name
Delete	The subject deletes an existing named global mutex.	count, pid, name

Attribute	Definition	Datatype	Example
count	The number of times the action occurred, if it occurred more than once.	pos. integer	5
pid	The process ID (PID) of the process which initiated the mutex action.	pos. integer	135
name	The name of the global mutex targeted by the action, without any prepended namespaces such as "global\".	string	someMutex

Appendix E: Process Action Types and Attributes

Action Type	Definition	Possible Attributes
Create	The subject spawns a new process on the system.	count, pid, image_filename, image_file_normalized_path, target_pid
Kill	The subject kills an existing process on the system.	count, pid, image_filename, image_file_normalized_path, target_pid
Write	The subject writes to an existing process on the system.	count, pid, image_filename, image_file_normalized_path, target_pid

Attribute	Definition	Datatype	Example
count	The number of times the action occurred, if it occurred more than once.	pos. integer	5
pid	The process ID (PID) of the process which initiated the process action.	pos. integer	135
image_filename	The name of the process image file targeted by the action.	string	ZCfgSvc7.exe
image_file_normalized_path	The normalized path to the process image file targeted by the action, using the appropriate pre-pended CSIDL constant, if applicable. If no CSIDL constant is applicable, then the fully qualified path to the image file.	string	csidl_programfiles\Intel\WiFi\bin
target_pid	The process ID (PID) value of the process targeted by the action.	pos. integer	1972

Appendix F: Network Action Types and Attributes

Action Type	Definition	Possible Attributes
Connect	The subject attempts to connect to an external network resource, namely a URL or an IP address.	count, pid, url, ipv4, ipv6, port, final_url
Listen	The subject listens for connections on a particular port/local IP address.	count, pid, ipv4, ipv6, port

Attribute	Definition	Datatype	Example
count	The number of times the action occurred, if it occurred more than once.	pos. integer	5
pid	The process ID (PID) of the process which initiated the process action.	pos. integer	135
url	The full URL requested by the subject.	any URI	http://some.bad.site
ipv4	The full IPv4 address requested by the subject.	string	145.22.61.200
ipv6	The full IPv6 address requested by the subject.	string	2001:0db8:85a3:0042:0000:8a2e:0370:7334
port	The particular TCP or UDP port number connected to or listened on by the subject.	pos. integer	80
final_url	For cases where the subject requested a URL by means of a referral, this specifies the final URL that resulted.	any URI	http://reallybadsite.com

Appendix G: Example Instances

Example 1: Complete Notional Instance (contains an example of every action)

```
<mmdefb
 xmlns="mmdefb"
 schema_version="1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="mmdefb mmdefb-1-0-schema.xsd">
    <subject md5="6519ED23FB22D0228E61A68D40F75C49"
sha1="7F42DF6B1318A5CC4633E33076629F95F59F1445">
        <dependency md5="79054025255fb1a26e4bc422aef54eb4" name="cfg.xml"/>
    </subject>
    <action_findings>
        <file action="create" name="badfile.dll"
normalized_path="csidl_system" pid="123"/>
        <file action="delete" name="winlogon.exe"
normalized_path="csidl_windows" pid="123"/>
        <file action="read" name="desktop.ini"
normalized_path="csidl_profile" pid="123"/>
        <file action="rename"  old_name="badfile.dll" new_name="kernel33.dll"
normalized_path="csidl_system" pid="123"/>
        <file action="write" name="desktop.ini"
normalized_path="csidl_profile" pid="123"/>
        <registry action="create" root_key="HKLM" key="SOFTWARE\Microsoft\
Windows\CurrentVersion\Internet Settings" value_name="UrlEncoding"
value_data="0x00000000" pid="123"/>
        <registry action="delete" root_key="HKLM"
key="SYSTEM\ControlSet001\Control\Safeboot" value_name="AlternateShell"/>
        <registry action="read" root_key="HKCU" key="Volatile Environment"
value_name="USERPROFILE"/>
        <registry action="write" root_key="HKCU" key="Environment"
value_name="TMP" value_data="C:\foo"/>
        <mutex action="create" name="SomeMutex" pid="123"/>
        <mutex action="delete" name="OtherMutex" pid="123"/>
        <process action="create" target_pid="442"
image_filename="kernel33.dll" image_file_normalized_path="csidl_system"
pid="123"/>
        <process action="kill" target_pid="1512"
image_filename="AeXAgentUIHost.exe" pid="442"/>
        <process action="write" target_pid="116"
image_filename="explorer.exe" pid="442"/>
        <network action="connect" ipv4="24.44.123.1" pid="116"/>
        <network action="connect" url="http://g000gle.su" port="80"
pid="442"/>
        <network action="listen" port="1444" pid="116"/>
    </action_findings>
</mmdefb>
```

Example 2: Netsky Worm Instance

```
<mmdefb
 schema_version="1.0"
 xmlns="mmdefb"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="mmdefb mmdefb-1-0-schema.xsd">
    <subject md5="3d23ec8b55840b95ea75197ce9446b6d"/>
    <action_findings>
        <file action="create" name="ranking_birth.zip"
normalized_path="csidl_windows" pid="229"/>
        <file action="create" name="winlogon.exe"
normalized_path="csidl_windows" pid="229"/>
        <registry action="read" root_key="HKLM" key="SOFTWARE\Microsoft\
Windows\CurrentVersion\Internet Settings" value_name="UrlEncoding"
value_data="0x00000000" pid="229"/>
        <mutex action="create" name="[SkyNet.cz]SystemsMutex"/>
    </action_findings>
</mmdefb>
```