

The Concepts of the Malware Attribute Enumeration and Characterization (MAEC) Effort

Ivan A. Kirillov, ikirillov@mitre.org
Desiree A. Beck, dbeck@mitre.org

Melissa P. Chase, pc@mitre.org
Robert A. Martin, ramartin@mitre.org

Abstract

Malware Attribute Enumeration and Characterization (MAEC) is a standardized language and format being formulated in cooperation with industry, government and academia for use in attribute-based malware characterization. MAEC is composed of a set of attribute enumerations, a schema, and a standard output format for the transport and communication of MAEC-encoded data. MAEC is being developed by MITRE under the sponsorship of DHS NCSA and others and will be part of MITRE's Making Security Measurable (MSM) effort.

Motivation

Malicious software represents one of the most prevalent threats to cyber security today. The enormity of this threat stems in part from the ever-increasing complexity of software, something that ultimately results in a growing number of vulnerabilities that can be exploited by an attacker. Clearly, any vulnerability capable of being exploited by a human attacker can also be exploited by a malicious program explicitly written to do so.

Thus, the protection of computer systems from malware is currently one of the most important information security concerns for organizations and individuals, since even a single instance of uncaught malware can result in damaged systems and compromised data. Being disconnected from a computer network does not completely mitigate this risk of infection, as exemplified by the recent wave of malware that uses USB as its insertion vector. As such, the main focus of the majority of anti-malware efforts to date has been on preventing damaging effects through early detection.

There are currently several common methods utilized for malware detection, based mainly on physical signatures and heuristics. These methods are effective in terms of their narrow scope, although they have their own individual drawbacks, such as the fact that signatures are unsuitable for dealing with zero-day, targeted, polymorphic, and other forms of emerging malware. Similarly, heuristic detection may be able to generically detect certain types of malware while missing those that it does not have patterns for, such as kernel-level rootkits. Therefore, it would be safe to say that these methods, while still useful, cannot be exclusively relied upon to deal with the current influx of malware.

However, there are a number of new tools being utilized in the battle against malicious binaries. Static and dynamic analyses are methods that the security community is converging upon in order to determine the malware attributes that would be helpful in combating this emerging threat. In this sense, automation of dynamic malware analysis through the use of sandboxes and similar methods allows for fast response to new malware instances based

on run-time behavior, while static analysis allows for a deeper understanding of malware, including its relationship to other malware. Yet, such techniques are hampered by the non-existence of a widely accepted standard for unambiguously characterizing malware.

The lack of such a standard means that there is no clear method for communicating the specific malware attributes detected in malware by the aforementioned analyses, nor for enumerating its fundamental makeup. Several major problems result from this, including non-interoperable and disparate malware reporting between organizations, disjointed or inaccurate malware attribution, the duplication of malware analysis efforts, increased difficulty in determining the severity of a threat, and an increased period of time between malware infection and detection/response, among others. On this basis, it is clear that a standard for describing malware in terms of its attack patterns, artifacts, and actions is needed to address such issues and allow for the clear communication of the information gained using static and dynamic analysis, which is the approach being taken by the Malware Attribute Enumeration and Characterization (MAEC) effort.

History

Malware Identification

AV product vendors and others commonly identify viruses and other forms of malware through the CARO (Computer Anti-virus Researcher Organization) naming scheme¹, which was first adopted in 1991. Although the value of knowing the specific malware threat that one is dealing with is significant, the fact remains that the CARO naming scheme is not an official standard, and is not applied consistently among its adopters. As a result, it is often the case that a particular malware instance has multiple, disparate CARO-based identifiers, thus furthering confusion regarding malware identity and reducing the value of the identifier.

A large part of the problem with a system like the one created by CARO is that it attempts to encode malware attributes as part of the identifier. It is very difficult to include all of the important information regarding a malware instance in its identifier without making the identifier too large and cumbersome to effectively utilize. Likewise, there is no way of determining which critical attributes should be present in the identifier and which to leave out, as certain attributes are relevant only to certain parties. It appeared that a better solution would be the use of standardized, non-attribute based malware identifiers.

CME

In the fall of 2004, MITRE began work on the Common Malware Enumeration effort². The goal of CME was to provide single, common identifiers for new and prevalent virus threats, in order to reduce public confusion during malware incidents. Like other MITRE security standards efforts, the intent was to collaborate with industry and develop a consensus approach. This community effort was not an attempt to replace the vendor names used for

¹ <http://www.people.frisk-software.com/~bontchev/papers/naming.html>

² <http://cme.mitre.org>

viruses and other forms of malware, but rather was intended to facilitate a shared, neutral indexing capability for malware.

In the first quarter of 2005 an initial capability was stood up. Anti-virus vendors submitted malware samples to a submission server with some metadata, the CME Board decided when a CME identifier should be generated, and the CME team identified the mapping between this sample and vendor names and created the content for the CME web site. For a while, CME seemed to fulfill its purpose, with identifiers being issued for prevalent malware incidents, as well as referenced in reporting about such malware.

However, the malware landscape experienced a paradigm shift in late 2006. Instead of a few widespread threats there was a move towards a large number of targeted malware instances, greatly reducing the need for a set of common identifiers for identifying malware. Accordingly, with the increasing reliance of AV vendors on heuristic-based detection methods, this meant that the sample-based mapping approach employed by CME no longer made sense, as a large number of new malware instances were being detected generically. Therefore, the last CME ID was issued in early 2007, and the CME-related efforts transitioned to supporting the DHS/DoD/NIST Software Assurance Malware Working Group.

DHS/DoD/NIST Software Assurance Malware Working Group

Shortly after the issuance of the last CME ID, the DHS/DoD/NIST Software Assurance Malware Working Group³ was stood up with representatives from MITRE and the Anti-Spyware Coalition (ASC)⁴ as co-chairs. The Malware Working Group was created to develop a framework of descriptive attributes of malware in order to:

- Improve communication by characterizing patterns (attributes and behaviors) to identify and describe malware types and instances
- Enable users to make informed decisions, i.e., lead to more stringent user acceptance criteria if potentially malicious code is to be installed with user knowledge
- Enable legal definitions of malware, spyware, adware
- Provide objective criteria for anti-malware tool assessments

MAEC

As we embarked on the Malware Attribute Enumeration and Characterization (MAEC) effort, we initially focused on a single use case from the Malware Working Group, namely attempting to develop a legally defensible definition of malware. This was driven by the desire to control the installation of spyware and adware and the ASC's success in using the concept of "potentially unwanted behaviors" to put the discussion in a more neutral setting. In talking to members of the security community involved with malware, we saw the need to broaden the Malware Working Group's set of use cases in order to provide greater uniformity in reporting malware, standardize the results of malware analysis, and support the development of databases describing adversary tactics, techniques, and procedures.

³ <https://buildsecurityin.us-cert.gov/swa/malact.html>

⁴ <http://www.antispywarecoalition.org/index.htm>

Objectives

Attribute-based Characterization

Under this broader concept, MAEC's main function will be to serve as a standard method of characterizing malware based on its behaviors, detritus, and attack patterns (see the Glossary for a definition of these and other terms). This will allow for the identification of malware based on patterns of distinct attributes rather than a single metadata entity (which is essentially what is used in signature based detection). Likewise, the use of such a method will mitigate the challenges of developing physical signatures posed by polymorphism, metamorphism, encryption, packing, and other obfuscation and armoring techniques.

Unambiguous Malware Typing

Malware has been binned into types since its inception, since knowing the type of a malware instance is useful for making a quick assessment of its behavioral nature and the threat that it possesses. However, malware types have always been defined fairly arbitrarily, and with little consensus. As a result, vendors and researchers have been identifying identical malware instances as different types, leading to confusion about what a malware type truly entails, and subsequently diminishing the value of malware typing.

Due to the fact that the various types of malware in existence (e.g., trojans, worms, viruses, etc.) have particular attributes in common, one aim of MAEC will be to permit accurate and unambiguous malware typing based on these attributes. For instance, computer worms are often typed on the basis of having the attribute of being able to propagate from one machine to another without any sort of user intervention. The analysis of large collections of malware and their corresponding MAEC-characterized attributes will permit the identification of the specific attributes necessary for defining malware based types and subtypes.

Malware Pedigree Tracking

As with malware typing, the usage of MAEC in describing malware instances will permit the identification of the minimum set of attributes necessary and sufficient for characterizing a particular malware family. This will enable the accurate identification of new instances of existing malware families through the comparison of their MAEC-characterized attributes with the unique set of attributes specific to a malware family. In this manner, MAEC should enable accurate and unambiguous malware pedigree tracking.

Legal Malware Definitions

MAEC's enumeration of malware attributes and their corresponding end-goals will permit the establishment of the behaviors and attributes that have been commonly observed as belonging to malicious software. Along with further analysis of individual malware samples, this will enable the creation of legal definitions of malware.

For example, back-up software that copies files from a user's hard disk to an off-site web server is typically classified as benign if it is installed by the user and the aforementioned

actions are performed at the behest of the user. A program that performs the same actions can be classified as malicious if it is intended to steal files by being installed and executed transparently, without the authorization and permission of the user. Thus, in this second case, transparency of insertion and non-user initiated execution are the attributes that differentiate benign versus malicious intent, and these important attributes would be included in the MAEC characterization of the program.

However, the majority of software behaviors can be classified as being either benign or malicious, with the context of their use quite often being the sole differentiator between the two. A program, with the same exact MAEC characterization as the malicious example described above, could also be used for benign reasons by a system administrator to back-up a user's system without the user's knowledge.

Clearly, MAEC could be used to define the groups of behaviors and attributes that have the *potential* to be malicious based on past observation. If a behavior has been observed multiple times in malware and has been identified as being consistently used for information theft, chances are that any other program that implements that same behavior is malicious too, as long as it implements the other known malicious behaviors. However, the concrete establishment of the intent of a piece of software based on its attributes and behaviors is a complex problem and one that is outside the scope of MAEC.

Scope

MAEC is being developed to address all known types, variants and manifestations of malware. Although initial work focuses on supporting the characterization of the most commonly discussed types of malware (such as, trojans, worms, kernel rootkits, etc.), MAEC will be applicable to the characterization of any of the more esoteric malware types. This can range from malware that is currently only a proof-of-concept, such as a hypervisor rootkit, to firmware and malware introduced into a product through its development activities and components, tools and library supply-chain.

As a formal language for attribute-based characterization of malware, MAEC's singular purpose is to facilitate the creation and communication of such data. Therefore, it is not intended to serve as an analogue or replacement for a unique malware identifier such as CME.

Likewise, MAEC clusters defined for malware instances are not canonical and will not be treated as such. This is because, unlike OVAL, MAEC's only component is the language itself, and as such we will not be maintaining a database or keeping track of content developed using the language.

Approach

In developing MAEC, we will engage industry, academia, government researchers, and developers so as not to focus exclusively on those with a single use case. Such collaboration

will allow for the establishment of an accepted enumeration of malware attributes, and will permit the definition of a schema with a broad set of applications. Likewise, it will enable the refinement of MAEC's features as needed to suit a broader audience.

For the purpose of discussing MAEC and facilitating interactions with industry we will utilize public discussion lists, announcement lists and private email exchanges. Accordingly, we will hold meetings, present conferences talks and papers, and perform other outreach activities in order to promulgate the MAEC effort and engage the community in its evolution.

The initial stage of development for MAEC will focus on the creation of the enumeration of low-level malware attributes, followed by the schema. Likewise, we plan on modeling various well-established types of malware (viruses, rootkits, etc.) using semantic web technologies such as RDF and OWL. Such modeling should help further refine the specifics of the language and allow for the establishment of initial namespaces and common relationships for use in the schema.

Proposed Framework

As a language and format for attribute-based malware characterization, MAEC's core components include a vocabulary, grammar, and form of standardized output (Figure 1, below).

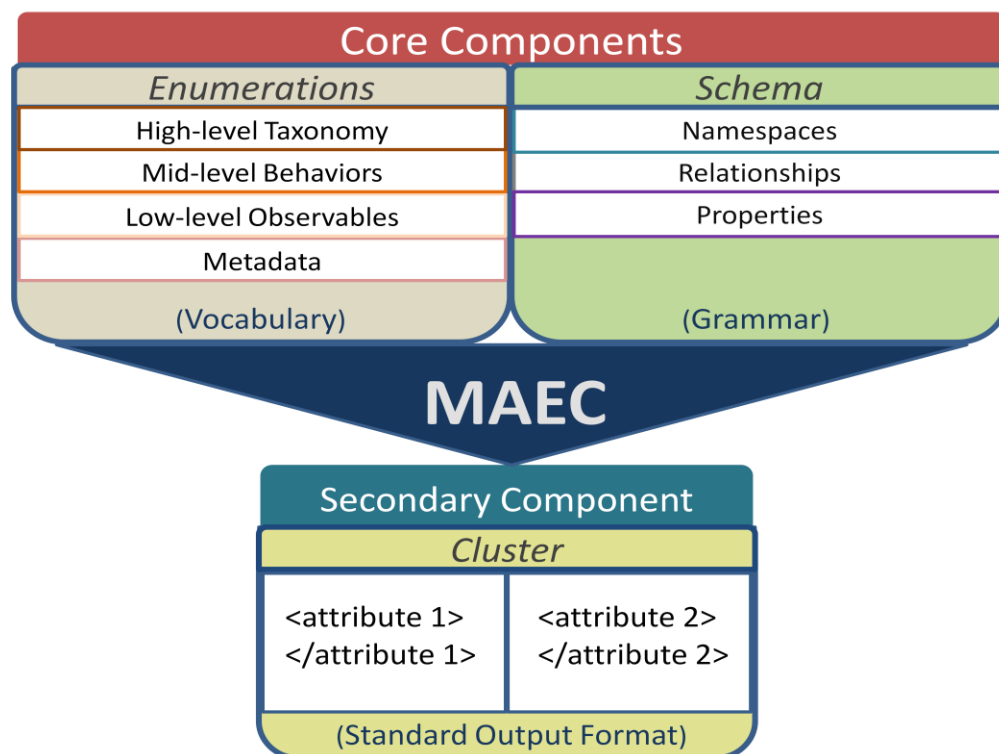


Figure 1: MAEC High-level Overview

The enumerated vocabulary is composed of three levels of malware attributes. MAEC's schema is effectively a grammar and defines the structure of the enumerated elements and the relationships between them. Finally, the MAEC cluster is a standardized format for the output of any MAEC characterized data.

MAEC's Enumerations

At its heart, MAEC will consist of a finite number of enumerations of malware attributes (Figure 2, below). Since malware is a form of software, and software can only perform the actions that can be achieved through execution of the instructions provided by the underlying system's Instruction Set Architecture (ISA), it follows that these attributes are finite and enumerable. Therefore, these enumerated attributes will include the detailed observables and behaviors performed by the software, as well as various metadata regarding the actions and the software in general.

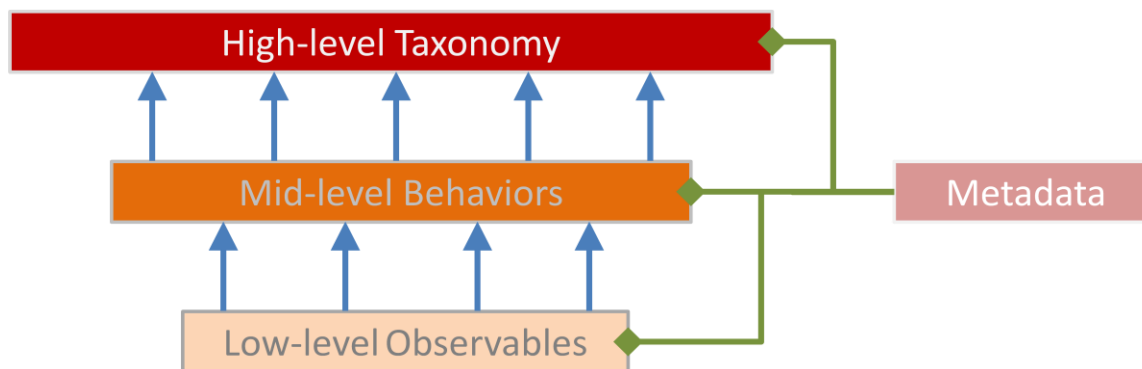


Figure 2: MAEC's Enumerated Attributes and their Linkages

MAEC's ability to communicate high-fidelity information about malware will be directly tied into its ability to accurately characterize the numerous types of malware in existence, as well as those created in the future. Therefore, at a minimum, MAEC must be able to describe any low-level actions performed by malware. However, its utility would be significantly degraded if it did not also have the ability to group such information into higher-level representations of malware behavior.

Low-level Observables

At the lowest level, MAEC will describe observable system state changes made by malware, including those which can be seen at the infrastructure level (i.e., the network). This can include changes to the file system and registry, modifications to active processes, the establishment of TCP/IP connections, etc. Likely sources of such data include automated dynamic analysis of malware binaries through sandboxes, host-based IDS, and IPS.

As the community defines MAEC, the enumeration and definition of these low-level observables will need to be done first, as it is likely that it will then be easier to gain consensus on mid-level behaviors. Accordingly, the current vision is to partition these enumerations into two discrete classes, namely those of host-based observables and network-based observables.

Mid-level Behaviors

At the middle level, MAEC's language will organize the aforementioned low-level observables into discrete clusters for the purpose of defining mid-level behaviors. This is to allow for the construction of a higher-level representation of malware behavior, thereby giving insight into the consequences of the actions performed by the low-level observables.

For instance, the description of a registry entry created or modified by malware can be useful in order to check for its presence on a system. However, it does not give any insight into *why* the malware created or manipulated the registry entry. Such a registry entry inserted or modified by malware could have many possible uses, including being used to ensure that the malware gets executed at system start-up, or as a simple flag to indicate that the system has been infected. Including the necessary components for characterizing such mid-level behaviors in the MAEC language will allow for the accurate description of the possible intent or goal that is behind the low-level system state changes being made by malware.

High-level Taxonomy

At the more conceptual and high level, MAEC's vocabulary will allow for the construction of a taxonomy that abstracts clusters of mid-level malware behaviors based upon the achievement of a higher order classification or grouping. We envision that such a taxonomy will have views (i.e. unique layouts) intended for different target audiences – for instance, forensic analysts may only be interested in looking at malware payload behaviors, etc.

To expound upon the example given for the mid-level behaviors, ensuring that malware is executed at start-up is a behavior that is typically part of a persistence mechanism. This behavior is often accompanied by the creation of a binary copy of the malware somewhere on the local hard disk. Therefore, in MAEC's top-level taxonomy, these two mid-level behaviors would be defined as belonging to the class of persistence mechanism.

Once MAEC's mid-level behaviors have been defined, the MAEC community will be in a position to begin the process of creating the high-level taxonomy and constructing the appropriate and necessary behavioral linkages.

Metadata

In order to include all pertinent information regarding malware and to fully describe the common actions of malware and the rationale behind them, MAEC will characterize malware-appropriate metadata. Such metadata can be thought of as additional information that can be used for better describing specific malware attributes. This can range from unique characteristics of malware behaviors, like the transparency of the insertion mechanism used, to metadata that can function as an attribute by itself, such as the type of packing mechanism utilized by the malware.

The MAEC Schema

MAEC's enumerations of behaviors and other attributes are necessary for the establishment of a vocabulary for characterizing malware. Therefore, the primary intent of MAEC's schema (Figure 3, below) will be to define a syntax for the discrete MAEC language elements. Likewise, the schema serves as an interchange format for the MAEC language,

and it can be utilized as a baseline for the creation of malware repositories or intermediate format for the sharing of information between repositories.

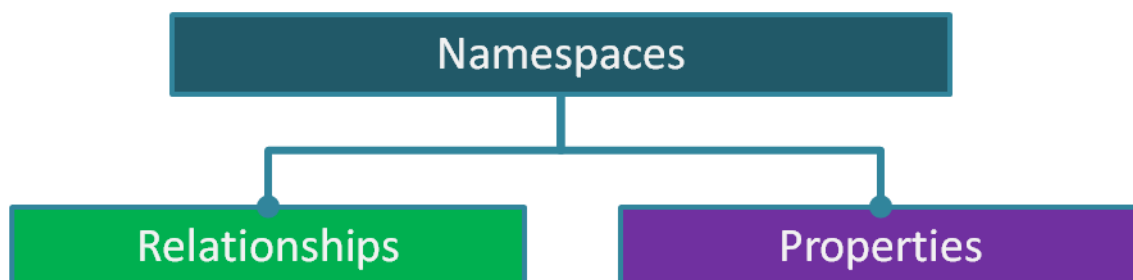


Figure 3: The Key MAEC Schema Components

At a minimum, the schema will define the following elements.

Namespaces

Namespaces in MAEC's schema will represent the grouping of the behaviors, observables, and other malware-related enumerated attributes of MAEC's language into well-defined classes. Where applicable, namespaces will follow those utilized by relevant Making Security Measurable (MSM) standards.

Properties

MAEC's schema will contain a general set of properties applicable to malware attributes and namespaces, with specific properties for behaviors. This can encompass things like the number of times a behavior occurs, whether it is the child or parent of another behavior, etc.

Relationships

MAEC's schema will include an established set of rules for defining the relationships among namespaces. For example, members of the namespace of mid-level behaviors can be composed of multiple members of the low-level observable namespace, while members of the low-level observable namespace can be associated with (but not composed of) multiple members of the mid-level behavior namespace.

The MAEC Cluster

The MAEC cluster (Figure 4, below) represents a standard output format of MAEC, with the purpose of encompassing any set of attributes obtained from the characterization of a malware instance. Therefore, it will serve as a container and transport mechanism for use in storing and subsequently sharing any MAEC-encoded information about malware. A MAEC cluster could be used to describe anything from a particular insertion method (composed of several low-level observables and mid-level behaviors), to all of the attributes of a malware instance, to the key behaviors common to an entire malware family.

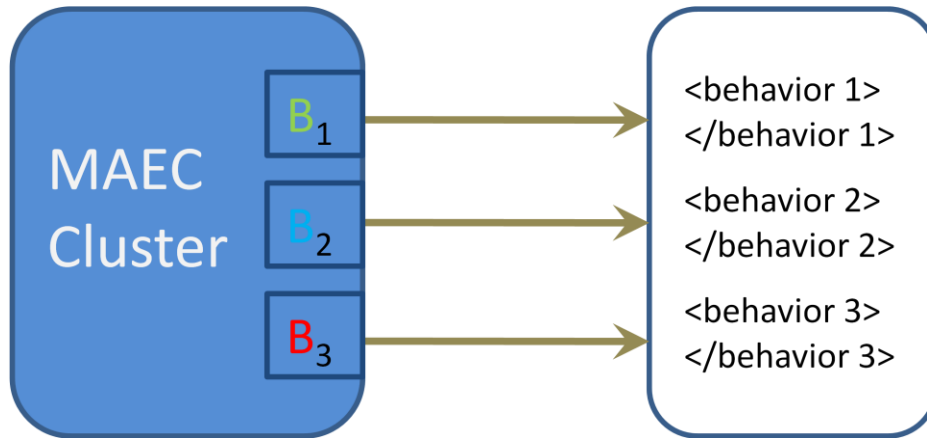


Figure 4: MAEC Cluster Overview

Although a MAEC cluster will be most useful when encompassing a set of malware attributes with a particular significance (like the insertion method or family behaviors mentioned above), it is intended to serve as a generic container for MAEC-characterized malware data. Therefore, it can be used with as little or as much information as desired; any further meaning beyond the explicit data stored in the cluster is defined by its producer.

```

<behavior id=1>
  <level>mid</level>
  <type>reconnaissance</type>
  <subtype>keyboardCheck</subtype>
  <attributes>
    <languageChecked>language:ukrainian</languageChecked>
    <successCondition>execution:stop</successCondition>
    <failureCondition>execution:continue</failureCondition>
  </attributes>
</behavior>

<behavior id=2>
  <level>low</level>
  <type>creation</type>
  <subtype>mutex</subtype>
  <attributes>
    <variableStringCreated>"Global\%u-%u"</variableStringCreated>
    <variable>"%u"</variable>
    <variabletype>datatype:decimal</variabletype>
  </attributes>
</behavior>

```

Figure 5: Prototype MAEC Cluster – XML Representation

In terms of the actual data-interchange format used for MAEC clusters, MAEC will initially support XML (Figure 5, above) due to its ubiquity and propensity for being human-readable as well as machine-consumable. In the future, it is likely that MAEC will support Resource Description Framework (RDF) as well as JavaScript Object Notation (JSON) and other lightweight formats.

Distribution

MAEC's attribute enumerations, schema, and associated content will be made available through the MAEC website⁵. New content releases will be announced on both the website and discussion list.

Primary Users

MAEC can be utilized by AV and other malware protection vendors, security organizations, researchers, analysts, and others who have the need for producing and/or consuming malware characterization information. It is also conceivable that tool vendors will implement MAEC for the purposes of malware assessment and the generation of MAEC clusters.

Ties to MSM Standards

As part of MITRE's Making Security Measurable (MSM)⁶ effort, MAEC will make use of other relevant MSM standards, where appropriate (Figure 6, below).

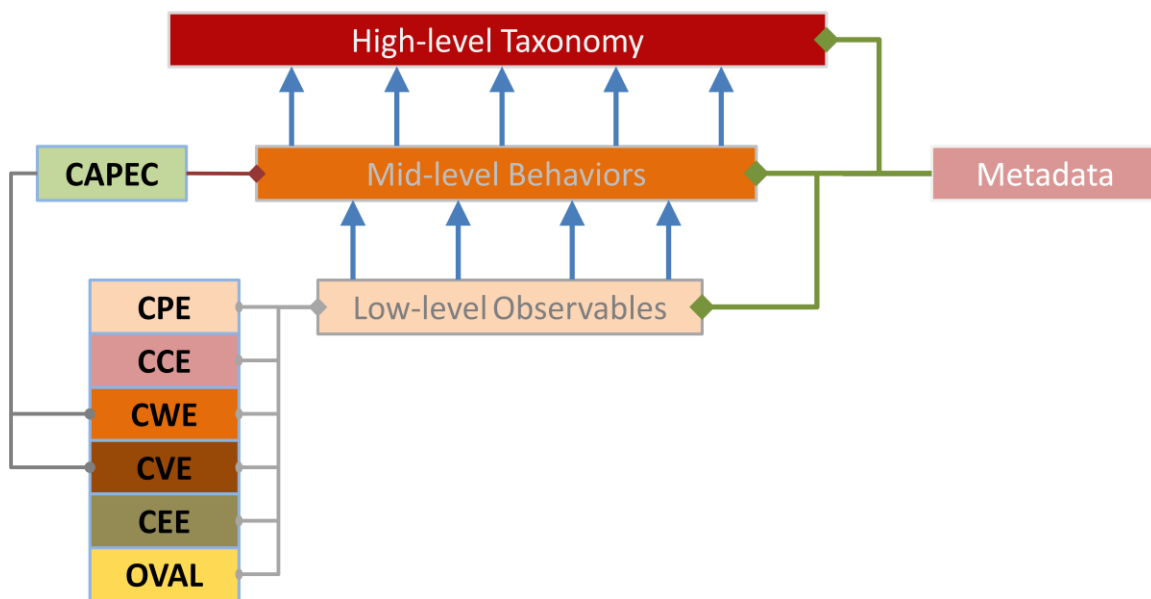


Figure 6: MAEC's Ties to MSM Standards

A number of these standards can be utilized for more accurately characterizing malware, especially with regards to standardized reporting and assessment of threats and vulnerabilities.

⁵ <http://maec.mitre.org>

⁶ <http://makingsecuritymeasurable.mitre.org>

CAPEC

MAEC will make use of the Common Attack Pattern Enumeration and Classification (CAPEC) ⁷ for describing the relevant attack patterns associated with the high-level malware taxonomy, such as those dealing with network reconnaissance, propagation, insertion, and command & control. The linkage of MAEC to CAPEC will allow for such behaviors to be defined through an industry standard attack pattern enumeration, thus ensuring that the attacker's perspective in implementing these behaviors is properly represented.

Likewise, this association will provide researchers with detailed information regarding the behavior's motivation (if included in the CAPEC entry). It is conceivable that such information could be utilized by researchers for determining the over-arching intentions of the malware author (by abstracting multiple CAPECs and other malware behaviors), as well as by developers for developing software with improved security against malware.

CPE

For a standardized description of the software and hardware platforms targeted by malware, MAEC will make use of the respective Common Platform Enumeration (CPE) ⁸ entry associated with the platforms, permitting the tool-based identification of potential victim machines by IT administrators.

CCE

The linkage of MAEC to the Common Configuration Enumeration (CCE) ⁹ will allow for the description of any of the vulnerabilities associated with malware that are not related to software flaws. This will allow for the host-based detection of specific configuration-related vulnerabilities exploited by malware, as well as the detection of general configuration issues that malware could potentially exploit. MAEC's link to CCE can also substantiate non-flaw based vulnerability threats by providing a concrete example of their exploitation, which will permit the prioritization of configuration vulnerability patching and associated threat assessment efforts.

CWE

If it is determined that a malware instance exploits a particular software weakness, MAEC will link to its corresponding Common Weakness Enumeration (CWE) ¹⁰ entry. This linkage will allow for the generation of statistics with regard to the most common types of weaknesses being exploited by malware, thereby highlighting the areas where better security-oriented coding practices need to be implemented. This linkage will also provide an attribute for correlation of malware when a specific CVE or CCE isn't being targeted by the malware.

CVE

MAEC will link to the Common Vulnerabilities and Exposures (CVE) ¹¹ entry associated with a particular vulnerability exploited by malware. This will allow users to determine the

⁷ <http://capec.mitre.org>

⁸ <http://cpe.mitre.org>

⁹ <http://cce.mitre.org>

¹⁰ <http://cwe.mitre.org>

¹¹ <http://cve.mitre.org>

nature of the vulnerability being exploited by the malware, as well as for automated fix and patch assessment through CVE-compatible tools. Likewise, MAEC's link to CVE will substantiate vulnerability-based threats by providing a concrete example of their exploitation, which will permit the prioritization of software vulnerability patching and associated threat assessment efforts.

CEE

MAEC will use the ubiquitous event description language provided by Common Event Expression (CEE) ¹² to describe logged events associated with malware activity. Such entries can be linked to specific malware behaviors and used to determine the presence of malware.

OVAL

Certain low-level malware observables may represent attempts at software vulnerability exploitation, meaning that such entries can be linked to corresponding Open Vulnerability Assessment Language (OVAL) ¹³ definitions (if in existence). Such a connection would allow for improved malware threat mitigation, by tying in the ability to easily check for the host-based existence of a vulnerability that is directly associated with a particular malware instance. Likewise, it can narrow down the potential malware variants capable of infecting a system by correlating the un-patched vulnerabilities present on a system with those linked to by MAEC characterizations.

OVAL can also be used to determine malware presence based on comparison of multiple scans. A common malware behavior is to patch the particular vulnerability used to exploit a system after successful infection, so that detection of such a “silently” patched vulnerability can be used to establish the presence of malware.

Use Cases

At the highest level, MAEC is a domain-specific language for non-signature based malware characterization. Languages serve to provide a vocabulary and grammar for the encoding and decoding of information. It therefore follows that the majority of the use cases for MAEC are motivated by the unambiguous and accurate communication of malware attributes that it enables.

While there are a number of ways that MAEC-encoded information can be utilized in some automated form, the majority of MAEC's use cases are human-oriented. That is to say, while MAEC will provide a foundational basis and structure for use in characterizing malware, we believe that such characterizations will be interpreted and utilized by humans more often than by machines.

Uniform Malware Reporting Format

Malware reporting, while useful for determining the general type and nature of a malware instance, is inherently ambiguous due to its lack of a common structure and vocabulary.

¹² <http://cee.mitre.org>

¹³ <http://oval.mitre.org>

Likewise, it often excludes key malware attributes that may be useful for mitigation and detection purposes, such as the specific vulnerability exploited. Clearly, the current value of malware reporting to end-users is significantly degraded without an encompassing, common format.

The use of MAEC's standardized vocabulary and grammar in malware reporting will facilitate the creation of a separate, uniform reporting format. Such a format will reduce confusion as to the nature of malware threats through the accurate and unambiguous communication of malware attributes, while also ensuring uniformity between reports composed by disparate authors and organizations.

Multiple Source Integration

The majority of useful data in malware reports is currently made up of unstructured narrative text, making it very difficult to accurately compare or combine multiple reports. Therefore, the standard vocabulary and structure of the uniform malware reporting format enabled by MAEC will allow for accurate comparisons between multiple malware reports, without the need for converting each report to some intermediate representation (Figure 7, below).

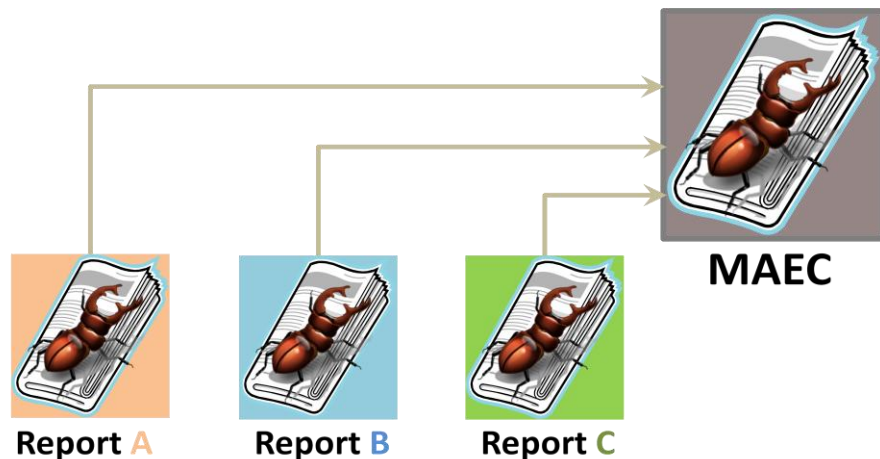


Figure 7: Multiple Report Integration with MAEC

As such, the use of MAEC in malware reporting will permit category-by-category comparison. This will not only allow for identification of differences and similarities among reports, but will also provide for the discernment of critical malware-related areas that need to be focused on or researched further.

Example Scenario

For example, say that an analyst is comparing multiple malware reports between two instances of the same malware family in order to determine the key attributes that differentiate the two. The usage of MAEC in these malware reports ensures that their data has a standard structure and uses a common vocabulary, even when considering their various creation dates and the fact that several different authors probably created them.

Therefore, the analyst is quickly able to contrast the two pieces of malware by locating the differing behaviors defined in their MAEC clusters. Since each cluster has a common structure, this amounts to simply determining the locations where they do not match, as there is no need to do any consistency checks or correlate differing descriptions of identical behaviors.

Malware Assessment & Detection

Characterizing malware based on its attributes with MAEC will permit the use of actionable information for malware assessment and detection. In this sense, low-level observables and mid-level behaviors will permit malware detection, while malware threat assessment can be carried out through MAEC's link to relevant MSM standards.



Figure 8: Malware Detection Based on Shared Attributes with MAEC

In terms of detection, a single MAEC characterization, represented by a MAEC cluster, can provide data that can be used to detect multiple malware instances, unlike with physical signatures. Because there are only a finite number of ways of accomplishing a single software behavior (for instance, malware insertion), particularly at the assembly level, it is statistically likely that there will be an intersection of such attributes between multiple malware instances. Therefore, the MAEC characterization of a single instance can permit the detection of malware families and even otherwise un-related malware that have certain attributes in common with the instance (Figure 8, above).

Threat Assessment

MAEC's linkage to OVAL, CPE, CVE, and CWE, will provide system administrators with the necessary information for determining the vulnerabilities and weaknesses targeted by malware. Accordingly, MAEC's encoding of mid-level behaviors and a high-level taxonomy will allow for the accurate discernment of the threat that it represents to their organization and infrastructure.

Example Scenario

For example, say that a new malware instance has been seen in the wild, analyzed, and has had a report issued for it. The report uses MAEC for encoding the key attributes of interest, and therefore links to the CVE entry detailing the vulnerability exploited by the malware for insertion as well as the CPE entry for the software platform that it targets.

An IT administrator looks at the report, and determines based on its CPE linkage that several of their machines are running the platform targeted by the malware. However, these machines have already been patched against the vulnerability defined in the CVE entry linked in the report. Based on this information, the administrator is able to determine that this malware poses a low threat and detection of it based on the other MAEC-encoded information contained in the report is not critical.

Host-based Detection

MAEC's encoding of information regarding low-level system observables will allow for host-based malware detection. This can be accomplished through manual system inspection for MAEC-defined entities such as active processes that signal the presence of specific malware instances.

Likewise, utilization of such information contained in MAEC clusters will permit tool-based detection. In particular, a tool could read a MAEC cluster and use the data contained inside to automatically generate OVAL queries for the purpose of detecting the existence of entities such as registry keys and files.

Similarly, using MAEC for characterizing malware will permit the definition of patterns of low-level observables and mid-level behaviors utilized by malware. Such information can be used to establish new rule sets for heuristic-based detection, based on the observed sequences of behaviors.

Example Scenario

For example, say that two unrelated malware instances, one known and one unknown, use the same insertion mechanism. This mechanism and its low-level and mid-level attributes have been characterized using MAEC for the known instance. An administrator therefore uses this single MAEC cluster in his automated malware scanning and is able to successfully detect the presence of the unknown malware instance.

Similarly, say that one of the aforementioned malware instances has a command and control mechanism that attempts to connect to a server every five minutes. If characterized using MAEC clusters, this behavior could be detected heuristically at the host-level based on the temporal nature of the connection pattern.

Network-based Detection & Mitigation

Although the use of malware-specific signatures in network intrusion detection and prevention systems (IDS/IPS) has become fairly widespread, there is still no general method for characterizing the relationship between distinct malware attributes and network infrastructure. Such a systematic way of linking malware behavior with outgoing network data could be used for infrastructure-based detection as well as accurate identification of observed malware behaviors. Likewise, the accurate characterization of incoming traffic as being associated with a malicious behavior can be used drop packets associated with command and control behaviors employed by bots and related malware.

Therefore, MAEC's role as a standard language and format requires that it will be able to characterize any such applicable information regarding malware. In particular, MAEC will provide the capability for integrating and characterizing any observables and patterns related to malicious network activity, thereby permitting network-based malware detection in combination with this data and IDS/IPS.

Example Scenario

For example, say that a new malware instance has recently been introduced into the wild, and that it exploits an unknown network vulnerability as a means of insertion. However, several MAEC clusters of this malware have been prepared and integrated in a report describing the new instance. This report characterizes the specific types of network traffic generated by this malware in order to infect systems, as well as its other behavioral and low-level observable attributes.

Using the MAEC-encoded information contained in the report, an organization updates their IDS rule sets to block any traffic that looks similar to that of the malware insertion process. Likewise, this information is utilized to detect any machines that have already been infected by the instance. When the exploit is identified and the MAEC clusters for the malware instance updated to reflect this, the organization utilizes this information to patch all of their systems, thus making them immune to the insertion of the malware identified by the MAEC cluster.

Malware Analysis

The analysis of malware using static and dynamic/behavioral methods is becoming increasingly important for the purpose of understanding the inner workings of malware. Such information can be utilized for malware detection, mitigation, the development of countermeasures, and further analysis. However, the lack of a common vocabulary for analysis makes it difficult to compare and utilize the results of analyses performed by different people and tools.

The encompassing attribute enumerations provided by MAEC, containing all possible malware attributes capable of being characterized through malware analysis, will enable the convergence of malware analysis results upon a common vocabulary. Utilization of such a vocabulary for malware analysis should eliminate the confusion and ambiguity resulting from the current use of multiple disparate vocabularies for analysis results.

Behavioral/Dynamic Analysis Tool Wrapper

For dynamic analysis tools the MAEC schema could be integrated into a wrapper-type tool in order to create a standardized version of the specific tool output (Figure 9, below). In this way, MAEC would serve to convert the specific output of any tool into a standardized format, thereby ensuring data compatibility between divergent tools and facilitating the sharing of behavioral malware data.

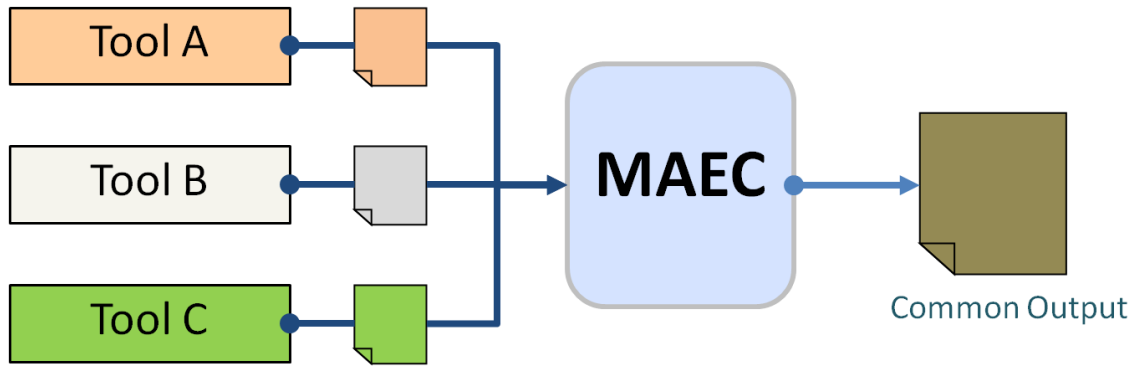


Figure 9: MAEC for Conversion of Disparate Tool Output to Common Format

Malware Repository

Malware repositories are commonly used to store artifacts associated with malware, along with malware analysis information. However, there is no standard for the structure of such information, making it very difficult to share malware-related information stored in repositories.

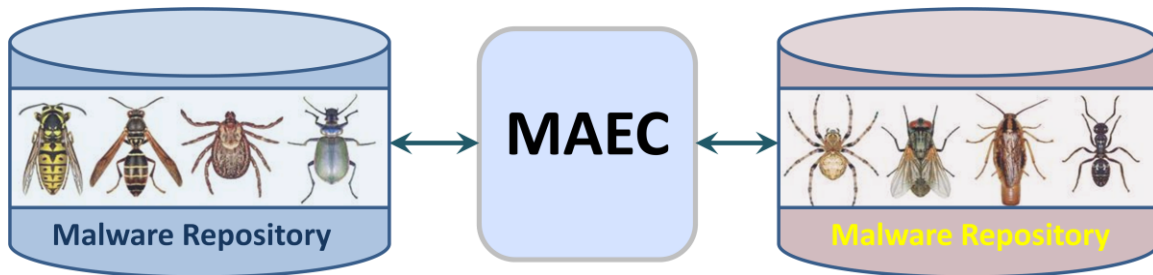


Figure 10: Sharing of Data among Disparate Repositories via MAEC

MAEC's schema could be used as a common intermediate format for mapping between the dissimilar schemas utilized in malware repositories. This would facilitate the sharing of analysis information stored in disparate repositories (Figure 10, above). Likewise, the usage of MAEC in malware repositories would permit improved data-mining due its structuring and labeling of malware attributes.

Enhanced Data-mining

By mapping a malware repository schema directly to MAEC's schema, any ambiguity between malware features and corresponding search queries could be eliminated. Since MAEC incorporates low-level observables, as well as mid-level behaviors, the integration of a malware repository schema with MAEC would allow for querying the repository based on discrete malware attributes and would provide for individual attribute-level comparisons between multiple malware instances.

Example Scenario

For instance, imagine searching a large unstructured repository for malware that propagates via spammed infected email messages by using the keyword text of "SMTP."

The results from such a query would include some relevant data relating to malware that does in fact use such a propagation mechanism, but could also have information about malware that connects to SMTP servers, attacks SMTP servers, or was originally received in an email message, just to name a few.

If the schema of the aforementioned repository were to be mapped to MAEC, querying the repository using the MAEC-defined category of (for example) "Propagation Vector: Email/SMTP" would retrieve the desired information, with completely accurate results. While this is a very simple query, it demonstrates the significant data mining capabilities provided by the usage of a domain-specific formal language.

Objective Criteria for Tool Assessments

MAEC's typing of malware based on discrete attributes can be utilized as objective criteria for use in the assessment of anti-malware tools. In this sense, a tool would be assessed on the basis of its support in detecting all of the attributes associated with a particular malware type. A tool that cannot detect certain MAEC-defined attributes associated with a particular malware type can miss any malware that contain such attributes, and therefore cannot objectively be defined as capable of detecting that type of malware.

Linking Malware TTPs

In cyber incident analysis, it is often useful to characterize the tools, techniques, and procedures (TTPs) used in the attack as being part of a set belonging to a particular attacker. When correlated across multiple attacks, such a connection can be helpful for the purposes of attribution.

Accordingly, with malware being one of the most prevalent tools used by attackers, it would be useful to characterize specific malware instances as belonging to a set of tools used by specific attackers. MAEC would provide this function, as its standard vocabulary and grammar will permit the accurate identification of the malware attributes observed in previous attacks, and with the "attacker" being defined as a metadata attribute, this would allow for the construction of an accurate link based on previously observed and characterized malware.

Concluding Remarks

The adoption of MAEC as an industry standard language for encoding high-fidelity information about malware will have several major implications for its stakeholders. First and foremost, it will allow for a vastly improved level of anti-malware related communication (including human to human, human to tool, tool to tool, and tool to human), due to its elimination of ambiguity and inaccuracy in malware description. This will positively impact all major stakeholders, including producers and consumers of malware analysis and related malware data, as well as the end-users of tools for malware prevention and mitigation.

Other benefits stemming from the adoption of MAEC include:

1. Reduced duplication of malware analysis efforts. A common way of characterizing malware along with a corresponding standard for malware analysis reporting will allow researchers and analysts to easily determine whether or not a particular malware instance has already been analyzed.
2. Improved general malware awareness. An adopted standard for characterizing malware will allow for increased public awareness of malware threats and activity due to its widespread usage throughout the entire anti-malware data producer->consumer chain.
3. Decreased overall response time to malware threats. The standard method of describing malware behavior provided by MAEC will allow for the faster development of countermeasures based upon those developed for previously observed malware instances.

Glossary

- 1) **Attribute:** simply, a characteristic of malware that can be used as a descriptor. For MAEC, attributes can include low-level observables, mid-level behaviors, high-level behaviors, and metadata.
- 2) **Attack Pattern:** a description of a common method for exploiting software, which can include the attacker's perspective and guidance on ways to mitigate their effect. Example: Exploiting incorrectly configured SSL security levels.
- 3) **Behavior:** the end result of the execution of a specific set of instructions by malware. In this manner, a behavior can be thought of as the consequence of an action. Example: the consequence of inserting a registry key (action) is allowing malware to be resident at system start-up (behavior).
- 4) **Observable:** any data that can be obtained by observing the host-level execution of malware on a system through some form of instrumentation. It is typically obtained through dynamic analysis methods and is dependent on the host operating system. Examples (broad): file system changes, GUI events, etc.
- 5) **Type:** any of the AV/security community's commonly referenced monikers for malware categories. Examples: virus, trojan, worm, backdoor, keylogger, rootkit, bot, etc.