

= What is the purpose of this work? =

Why is it necessary to share meta-data? The industry has long-established sharing of samples. A malware sample is always very useful for analysis but additional information about the source, the commonality and timeframe when it was relevant can be very valuable too. For any malware sample there are multiple characteristics that, if shared, would help enormously in providing users with timely protection. Such data would also help in achieving higher quality of testing and making it more relevant. For example, the meta-data can help in:

- * Prioritizing samples, URLs, domains, etc. in research queues of security companies to protect users quicker and better
- * Ranking threats in test sets for comparative testing to make tests more relevant
- * Reducing the number of sample requests between security companies
- * Transmitting the information about relationship of a sample and, say, spammed URL
- * Better research due to deeper understanding of how malware operates
- * Faster reaction to important events
- * Quicker fixing of false alarms, especially those known as “snowball” false alarms (when the same false detection propagates between different products, usually due to automation)
- * Increased cooperation of security vendors to protect users and global computer ecosystem better

Even though it is most important to share the meta-data for malware samples, sharing meta-data for clean or “grey” files within the context of malware data sharing makes a lot of sense too. This, for example, can help flush common false alarms quicker. So we speak about XML with meta-data related to malware research but we do not mean sharing of meta-data related only to malware!

= Why use XML? =

To make parsing of meta-data simple the format should be:

- * Standard (so that there are ready tools to perform the task)
- * Extendable (so that new types of records can be added when necessary)
- * Easy to read by humans
- * Easy to parse by automated tools

A natural suggestion for the format to hold the meta-data is XML format as it satisfies all of the above requirements (<http://en.wikipedia.org/wiki/XML>). When needed we would be able to add additional elements to this XML and make it richer and more useful.

When shared, XML should be placed inside a ZIP or RAR archive. That would add the following benefits:

- * possibility to chunk huge files into small pieces (e.g. 5MB for easier Emailing or 2GB to fit onto FAT32 volumes or allow the use of software that has 2GB limitation on the file size)
- * external verification of file integrity via a CRC check built-in into ZIP/RAR software

XMLs described in this document must be well-formed and obey the XML standard.

= Can you give me some background to the schema? =

The XML schema is fully described in [<http://grouper.ieee.org/groups/malware/malwg/Schema1.1/metadataSharing.xsd>] and is accompanied by [<http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc> documentation] and [<http://grouper.ieee.org/groups/malware/malwg/Schema1.1/examples>].

The general idea is to describe “atomic” (simple, significant, independent and predominantly static) objects and link these objects with a set of relationships. As a set of atomic objects we identified the following:

- * File
- * Registry
- * URI
- * IP or IP range (can be IPv4 or IPv6)
- * Domain
- * ASN
- * Classification (e.g. a detection name of a certain security product or a verdict of a common tool)
- * Entity (for example, a malware-writing group responsible for creation of certain family of viruses)

Each of these objects can be referenced (via an “xpath” in XML; see “ref=” in examples) by another object by using a relationship link. To allow referencing files we use a hash (or a set of hashes) while symbolic objects (e.g. URLs and domains) – by quoting the entire object as a string (see Appendix A about normalizing URLs). The relationships reflect a wide variety of how objects are related to each other (e.g. “parent”, “child”, “”) and they provide the richness and flexibility. Relationships may change so they may carry a timestamp.

The following relationships, for example, can be used (for the complete list please check the full schema in the [<http://grouper.ieee.org/groups/malware/malwg/Schema1.1/metadataSharing.xsd>]).

- * isClassifiedAs
- * hosts
- * installed
- * isParentOf
- * causesToInstall
- * downloads
- * exploits
- * runs
- * usesCNC (malware uses a specific command-and-control center)
- * resolvesTo
- * isNameServerOf
- * verifiedBy

- * isServerOfService
- * downloadedFrom
- * operatedByEntity
- * hasAssociatedConfiguration

Using this approach one can, for example, describe the following common use cases:

- * a bunch of samples detected under one name by an AV scanner can be a series of samples (presented as hashes) linked via “isClassifiedAs” relationship to a classification entry stating which product detects these samples under a specific name (BTW, this example covers detection of polymorphic viruses and
- * a series of URLs serving the same malicious Trojan file can be linked via “downloadedFrom” to the hash of that sample (and supplied with an appropriate timestamp because downloaded file may change over time)
- * a group of domains can be linked to an IP address via “resolvesTo” (with a timestamp, of course)
- * a series of IP addresses (or ranges) can be linked to a malicious group by using “operatedByEntity”
- * a file is classified as clean by using a “verifiedBy” link to a research lab or a person

Apart from objects and relationship the schema also provides the historical timeline (as FieldData) and importance/priority. Field data carries the information on the field sightings and commonality:

- * From – user/desktop/network/gateway
- * Specific source - country/region/ipv4/ipv6
- * When the object was first and last seen (multiple “seen” tags are allowed to show history)

Classification allows to describe what an object or a group of object is (are):

- * Is it – clean/dirty/unknown/unwanted/neutral
- * Detection – product(s), detection name(s), etc. (detectiontype/product/version/defs/added/shipped/removed/name/malwaretype/parasitic/polymorphic/kernel/stealth”). Note – “detectiontype” can be av/behavioural/cloud/whitelist/blacklist.
- * Category of malware

= What hashes should be used for file objects? =

You should provide as many hashes as you can inside the file object. For the id, the rule is that you should use sha256 if you have it, failing that use sha1 if you have it, failing that use md5.

== Add information on a file object? ==

Create a

[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/fileObject.html file] object in the objects section filling in the various hashes if

available. Make the id of the object the sha256 if you have it, failing that the sha1 and failing that the md5.

E.g see [<http://grouper.ieee.org/groups/malware/malwg/Schema1.1/examples/minimal.xml>] or

```
<file id="2f437c1c8f73c2d6ffbb6214d3f1ccfe994151b3bd80fe2b3934a1bc89384599">
  <md5>8b31da6402d850ce94e7c19bc97effe1</md5>
  <sha1>850e5b037c799f86f04ee63da786f9ee139ebf57</sha1>

<sha256>2f437c1c8f73c2d6ffbb6214d3f1ccfe994151b3bd80fe2b3934a1bc89384599</sha256>

<sha512>e4e9cd843ae82f9c0b82d02d7f4ce049efcaa968130604ae8080d9e237a6c52baa6ca0cba7a7
33da383881df4ad345db50d620784def6bb6826d344715105ded</sha512>
  <size>32769</size>
  <crc32>34efdbca</crc32>
</file>
```

or

```
<file id="8b31da6402d850ce94e7c19bc97effe1">
  <md5>8b31da6402d850ce94e7c19bc97effe1</md5>
</file>
```

= Share filenames or paths for a file? =

Filenames and paths are recorded as

[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/objectProperty.html objectProperty] of a file. e.g.

[<http://grouper.ieee.org/groups/malware/malwg/Schema1.1/examples/illustratingProperties.xml> example]

```
<objectProperty>
  <references>
<ref>file[@id="sha256:2f437c1c8f73c2d6ffbb6214d3f1ccfe994151b3bd80fe2b3934a1bc89384599"]
</ref>
</references>
  <timestamp>2009-04-17T12:34:56</timestamp>
  <property type="filename">foo.exe</property>
  <property type="filename">bar.exe</property>
  <property type="filename">foobar.exe</property>
  <property type="filepath">%windir%\system32\foo.exe</property>
  <property type="filepath">%programfiles%\foo\bar.exe</property>
```

```
</objectProperty>
```

= Share the detection name for a file? =

This is expressed as a

[\[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/relationship.html\]](http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/relationship.html) relationship] between a

[\[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/fileObject.html\]](http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/fileObject.html) file] object and a

[\[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/classificationObject.html\]](http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/classificationObject.html) classification] object. The

[\[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/simpleType/RelationshipTypeEnum.html\]](http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/simpleType/RelationshipTypeEnum.html) relationship type] is "isClassifiedAs".

See this [\[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/examples/simple.xml\]](http://grouper.ieee.org/groups/malware/malwg/Schema1.1/examples/simple.xml) example].

= Share urls ? =

Urls are broken into the uri part (in a

[\[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/uriObject.html\]](http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/uriObject.html) uri] object) and the query string part, which is put in an

[\[http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/objectProperty.html\]](http://grouper.ieee.org/groups/malware/malwg/Schema1.1/xsddoc/http_xml_metadataSharing.xsd/complexType/objectProperty.html) objectProperty].

E.g

```
<uri id="http://count.key5188.com/01/get.asp">
  <uriString>http://count.key5188.com/01/get.asp</uriString>
</uri>

<objectProperty>
  <references><ref>uri[@id="http://count.key5188.com/01/get.asp"]</ref></references>
  <timestamp>2009-04-17T12:34:56</timestamp>
  <property
type="urlParameterString"><![CDATA[mac=12:12:12:12:12:12&ver=123&os=MicrosoftWindowsXP&
temp=473]]></property>
  <property type="urlParameterString">getDetails=1</property>
  <property type="urlParameterString">id=34</property>
</objectProperty>
```

= How do you calculate commonality? =

We use two measures for prioritizing malware – objective (commonality) and subjective (importance). Commonality is calculated from the sightings of malware objects (and so such calculation is easier to automate). Importance is reserved for cases when “commonality” is not available or if there is a need to communicate the importance when commonality is low.

We define the commonality on a scale 0 to 100 (0 means “never found in the field” and 100 means “found very frequently”). Scaling commonality to 0..100 range instead of using actual sample counts is to avoid the effect of the user base size on the commonality. We derive commonality from the number of affected computers – not from the number of samples (for example, a hundred parasitic infections of the same virus on a single computer are to be counted as one).

To calculate the commonality we use two-stage approach and logarithmic scale:

- * If the number of affected users exceeds 0.1% of your user base (more frequent than 1 in a 1000) set commonality to “100”
- * Otherwise, calculate the ratio of infected computers amongst your user base by dividing the real number of affected computers ‘n’ by the total number ‘N’
- * Apply the following formula to get the commonality – $(\log_2(1+n*1000/N)) * 100$
- * Round to the closest integer

To illustrate, the following translation “commonality”->“users” would apply if we assume the user base of 100 million:

Commonality	Users affected out of 100 million
1	1...696
10	~7177
50	~41421
100	100,000..100,000,000

Obviously, the calculation above can only be applied to counting of malware sightings on desktops. If telemetry is collected from a fraction of such desktops then an appropriate correction should be used. For all other cases (e.g. sighting on gateways, in some network security appliance, on an ISP level, etc.) please exercise your best judgment and apply provided desktop guideline as an example to make sure the commonality factor is as comparable as possible.

For a URL object the commonality could reflect, for example, how widely it was spammed.

= How to calculate importance? =

“Importance” should not be used together with “commonality” (unless commonality=“0”) to avoid possible confusion. High “importance”, for example, can be assigned to samples that are over-hyped by media when their commonality is still “0”.

Use the following guidelines for “importance” which is also defined on a scale 0..100:

- 100 – you’d expect your CEO and/or media to call you any second about this object
- 80 – you might get a call from your CEO and/or media
- 60 – you’d expect your boss to call you any second

40 – you might get a call from your boss
20 – someone is very likely to contact you about this object
10 – you might get contacted about this object
0 – you’d be surprised if anyone would ever contact you about this object

= How should the metadata be shared? =

The standard does not dictate how metadata is shared, or who it is shared with. However, there are some industry standard methods described here.

Two methods of sharing XML meta-data wrapped inside archives can be used:

- * Email (SMTP)
- * FTP, HTTP or HTTPS

Email is more suitable for small and rapid updates because it is a “push”-based technology. FTP/HTTPS are “pull”-based and are suitable for automated retrieval of regular meta-data updates. The FTP/HTTP/HTTPS site must use a static domain name or IP address and fixed folder structure. It must also either have predictable filenames or support retrieval of the list of files.

It seems awkward to include integrity checking and authentication into individual records of the XML. If necessary this can be done later. At the moment PGP-signing the archive will be a better solution. So, the XML must be compressed (this provides integrity checking), PGP-encrypted (provides access control) and PGP-signed (provides authentication).

Use PGP software in its compatibility mode so that even the oldest command-line products can be used. Traditional method used for encrypting malware collections is to use free PGP 2.6.3 software with 1024-bit keys (Note – shorter keys must not be used!). (This PGP version is rather old but it is used for many years so a lot of scripts and automation have been built around it already. And it is free.

The reason is that AV companies have been using this PGP version for many years to exchange malware collections. Respective PGP keys have been exchanged, cross-signed, verified and key rings updated. So there is significant infrastructure already in place that we can use.

Note that one serious limitation of PGP 2.6.3 is that it cannot encrypt files exceeding 2GB in size properly. Such big files should be first sliced into pieces, for example, by using “RAR -v” option and then each piece can be encrypted separately.)

= How should uris be canonicalized? =

Uris (and query strings) should be canonicalized in accordance with [\[http://code.google.com/apis/safebrowsing/developers_guide.html#Canonicalization\]](http://code.google.com/apis/safebrowsing/developers_guide.html#Canonicalization) Google guidelines].