CMX: IEEE Clean File Metadata Exchange

Mark Kennedy, Symantec
Dr. Igor Muttik, McAfee

Introduction

The malware problem is constantly growing—in both quantity and complexity. There is even a prediction that the count of malware samples will exceed the number of clean files [1]. At the same time, evasion methods employed by malware authors make it progressively harder to distinguish malware from clean software. The majority of contemporary malware is coded in high-level languages that makes them more similar to legitimate software than to software of 10 to 15 years ago, when most viruses were written in x86 assembly language. Compilers, linkers, and standard libraries anonymize the programmer's coding style, whereas low-level coding makes exposes it.

One other major factor complicating the job of differentiating malware from clean software is obfuscation. Legitimate software employs software protection techniques (such as packers, protectors, and software envelopes) to prevent reverse engineering and hide the intellectual property of the creators. Malware writers use software protection to complicate and delay recognition and detection of malware; this widens the window of opportunity for them to deploy malware.

Main job of antivirus software is to recognize malware, including new (a.k.a. unknown or zero-day) malware. This has to be done while maintaining a very low level of false alarms (false positives); otherwise antimalware software would be of little use.

To detect new malware, antimalware companies use extensive generic and heuristic methods. Moreover, to cover the new and evasive types of malware new heuristic antimalware technologies are being constantly introduced and their sensitivity has to be frequently tuned to proactively pick up new malware. As a result, all detection methods produce occasional false positives. In many cases this is not a problem, as antimalware companies maintain extensive test sets of clean software, perform testing of detection technologies, and can apply mitigations to reduce the false positive–false negative rate. So these false positives are not visible to users.

Indeed, the simplest false-positive mitigation is to exclude known clean software from being scanned and apply heuristic and generic methods only to the rest of software. Frequently such an exclusion list is called a whitelist, which can be deployed locally or even in the cloud. This approach produces great results. In fact if we imagine that one manages to compile a 100 percent–complete whitelist, then heuristics can be set to the most paranoid level and they would still never produce a false positive. In the "reductio ad absurdum" case, we can see that an antimalware product will not require any technology other than a whitelist. It will be able to safely block any software that does not appear in the whitelist!

The reality, of course, is not so simple. There are many factors why a whitelist can never cover all clean software and why it cannot guarantee classification accuracy for all entries:

1. It is operationally very hard to collect clean software from all its sources

    - Even trusted whitelist sources can be compromised (we see a constant increase in the number of stolen authenticode certificates used to create digitally signed malware that appears to come from a "clean" source)
    - Some software sources may be operated by malware authors (who have sufficient resources to maintain a pretense of being "trusted")

2. There is always a delay between the moment when a clean software is produced and when the whitelist is updated

3. Certain programs (for example, .NET executable files with JIT) are intrinsically variable and reliably whitelisting them is problematic

4. A whitelist is a representation of black-or-white classification; there are always shades of gray

    - Some legitimate software can be misused (like potentially unwanted programs/apps, for example, remote-desktop software can be abused as backdoor Trojans)
    - Even software from a trusted source may have hidden malicious functionality ("Software Easter Egg") that is not known (discovered later or never)

The need to have a better whitelist has been recognized by all security companies participating in IEEE Industry Connections Security Group (ICSG [2]). We believed that a cooperative system would have a much better chance of being successful than all previous attempts, which were predominantly limited to single-vendor efforts.

We decided to focus on the hardest problem we have discussed: collecting clean software in a standard way, opening access to the system for trusted software publishers, and making the data consumable by all legitimate security companies. But, at the same time, we built in tracking of the metadata source to allow dynamic trust levels for the metadata providers.

Current Whitelisting Approaches

Each security product has its own version of whitelisting, each with varying degrees of effectiveness. Most of these will be bolstered by the IEEE clean file metadata exchange, or CMX. Let us look at some of these, and how CMX will help.

1. On-machine whitelists
    - On-machine whitelists have existed for as long as signature databases have existed. Generic signatures will often have some number of false positives, and these whitelists prevent them for future customers. These are reactive whitelist entries (meaning some customers have already seen false positives).
    - CMX can help in this case because clean files provided by CMX can be scanned with current signature sets, and any false positives can be suppressed and the clients updated (or the signature modified). This can lead to proactive false-positive protection.

2. Cloud whitelists
    - A relatively recent addition. Many security vendors now use the cloud to help prevent false positives. These have expanded beyond the simple whitelists we have described because companies actively seek clean files to populate their cloud. This does allow some false positives to be proactively prevented.
    - CMX can help tremendously. Currently, security vendors must seek clean files. Some security vendors make special arrangements with third-party software vendors to gain access to these files. This is extra work for both the security vendor and the third parties. Moreover, this work is being done in parallel by multiple security vendors, which increases the complexity for the third parties.
    - With CMX, third-party software vendors will have a single point of contact with all subscribing security vendors. They need one interface. This greatly simplifies their process, and will make this type of interaction more attractive to other third parties. Additionally, the work of the security vendors is made simpler, because they no longer have to process multiple streams (likely using different interfaces) to obtain their information.

The CMX system

CMX aims to provide timely information about clean files in the form of metadata. Initially the metadata should be provided for three types of Windows files: executables (PE), .CAB, and .MSI files. The metadata extraction should be done after the final version is created, and all digital signing is complete. It should be for all software that is released to the public. Nightly builds and limited betas are not the best candidates because they will not reside on many machines. Public betas should definitely be included.

What type of metadata should be included?

The type of metadata generated will vary by file type. In general the list includes (but need not be limited to):

- Hashes. Minimally, these should be MD5, SHA-1, and SHA-256. SHA-512 and SHA-3 can also be considered.
- Filename. The name as it will appear once installed on a user's machine.
- Path. The path where the file will appear once installed, using CSIDL normalized paths.
- Signature information. If the file is digitally signed, information about the signing certificate.

- File version information. The various fields from the file version record.

Why don't we share files?

Many have asked why we are building a new system to share metadata when there are existing mechanisms for sharing files. There is a key difference between some clean files and malware files: copyright. Many clean files are copyrighted, and their distribution is controlled—even for whitelisting purposes. By sharing only metadata we are able to circumvent this problem.

It is also convenient that most cloud whitelisting solutions do not actually require the file; the hashes included in metadata (presuming we capture all we need) are sufficient. This will drive requirements on the metadata collection tool, to ensure we are capturing all the hashes and fingerprints that various consumers of the metadata will need.

Single point of contact for third-party developers

As we have mentioned, some security vendors have arrangements with some third-party software vendors to exchange files or metadata. These are separate agreements, and each one can be different. If a third-party software vendor provides this data, it must replicate the effort for each vendor it shares with. This can become cumbersome.

With CMX, these third parties must have a single process and deal with a single entity with a single interface. This will simplify their efforts, and lower the cost to the organizations. With lower costs, more third parties can be enticed into joining the effort. The more third parties that join the effort, the better consumers are served. Everybody wins (except the malware authors, but that is the point).

How the system works

Let us now explore in more detail how the system works, the types of users, and some of the use cases.

Types of users

There are two types of users in the CMX system: providers and consumers. Consumers will also be providers if they are software companies with their own products.

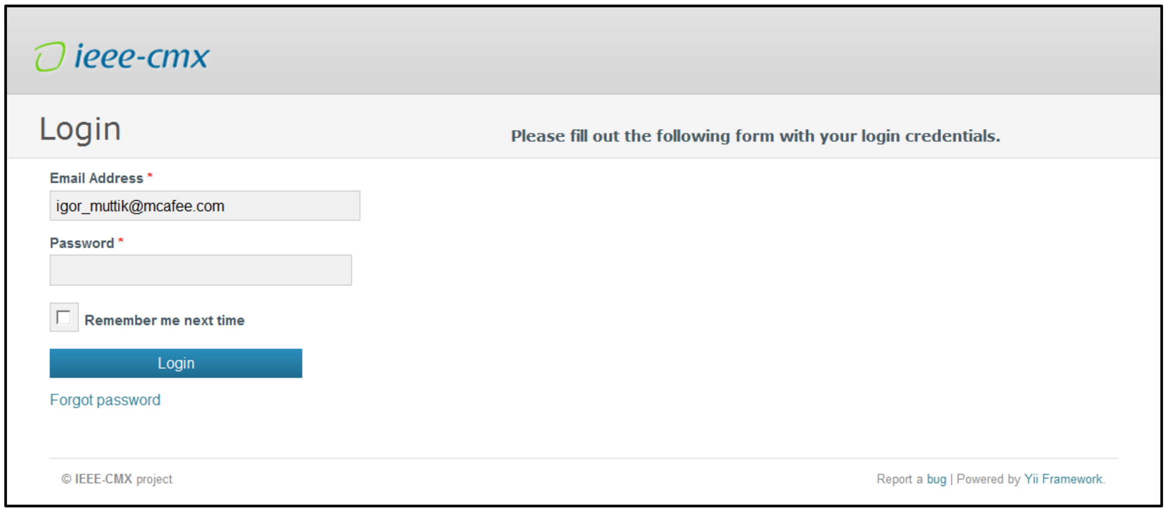
Providers create the metadata and submit it to the CMX system. It is formatted as the IEEE metadata XML [3] developed previously by Malware Working Group within IEEE ICSG [4]. A Python script is provided to assist in the extraction and formatting of the metadata. The data put into the system can be pulled out by the consumers. There is a mechanism for consumers to get only the data that has been added since the last time they pulled data.

The server will keep a window of the most recent data. The size of that window is a function of the amount of storage and the amount of inputted data. Periodically, older data will be archived off the system. Subscribers can have access to archives, but it will be provided in chunks. There is no processing of the archived data to remove data that the subscriber already has. (The consumers must sort that out.)

Current status

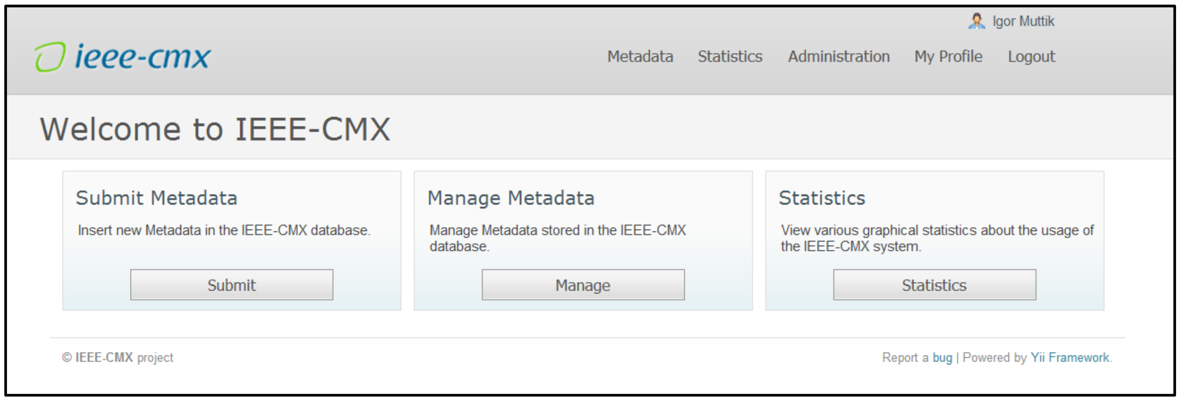
The system is now fully operational and hosted on servers owned by Avira in Germany (https://ieee-cmx.avira.com). CMX is somewhat similar to the MUTE system [5, 6], which was implemented by Avira to share malicious URLs. CMX required several modifications (including specific metadata extractors implemented currently in Python), but it is largely based on MUTE.

The following screenshots show the CMX portal:


ieee-cmx
Login
Please fill out the following form with your login credentials.
Email Address *
Password *
Remember me next time
Login
Forgot password
© IEEE-CMX project
Report a bug | Powered by Yii Framework.

igor_muttik@mcafee.com

Login screen.

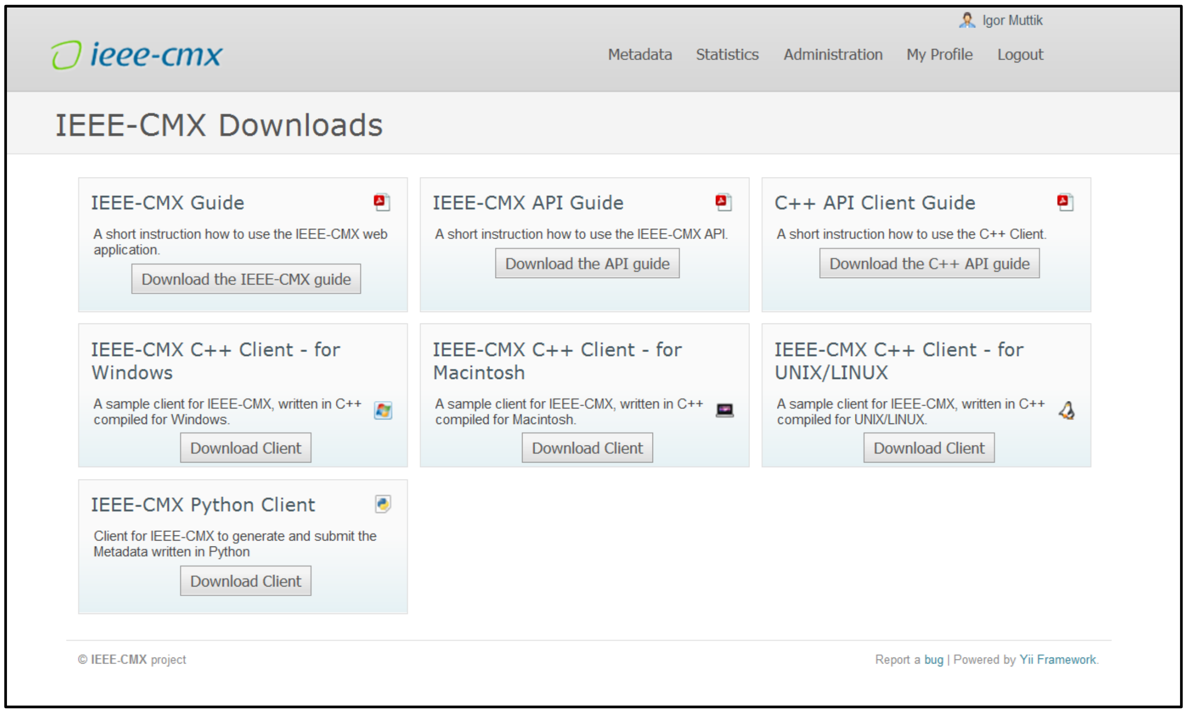


Insert new Metadata in the IEEE-CMX database.

Manage Metadata stored in the IEEE-CMX database.

View various graphical statistics about the usage of the IEEE-CMX system.

Main menu.



IEEE-CMX Downloads

IEEE-CMX Guide
A short instruction how to use the IEEE-CMX web application.
Download the IEEE-CMX guide

IEEE-CMX API Guide
A short instruction how to use the IEEE-CMX API.
Download the API guide

C++ API Client Guide
A short instruction how to use the C++ Client.
Download the C++ API guide

IEEE-CMX C++ Client - for Windows
A sample client for IEEE-CMX, written in C++ compiled for Windows.
Download Client

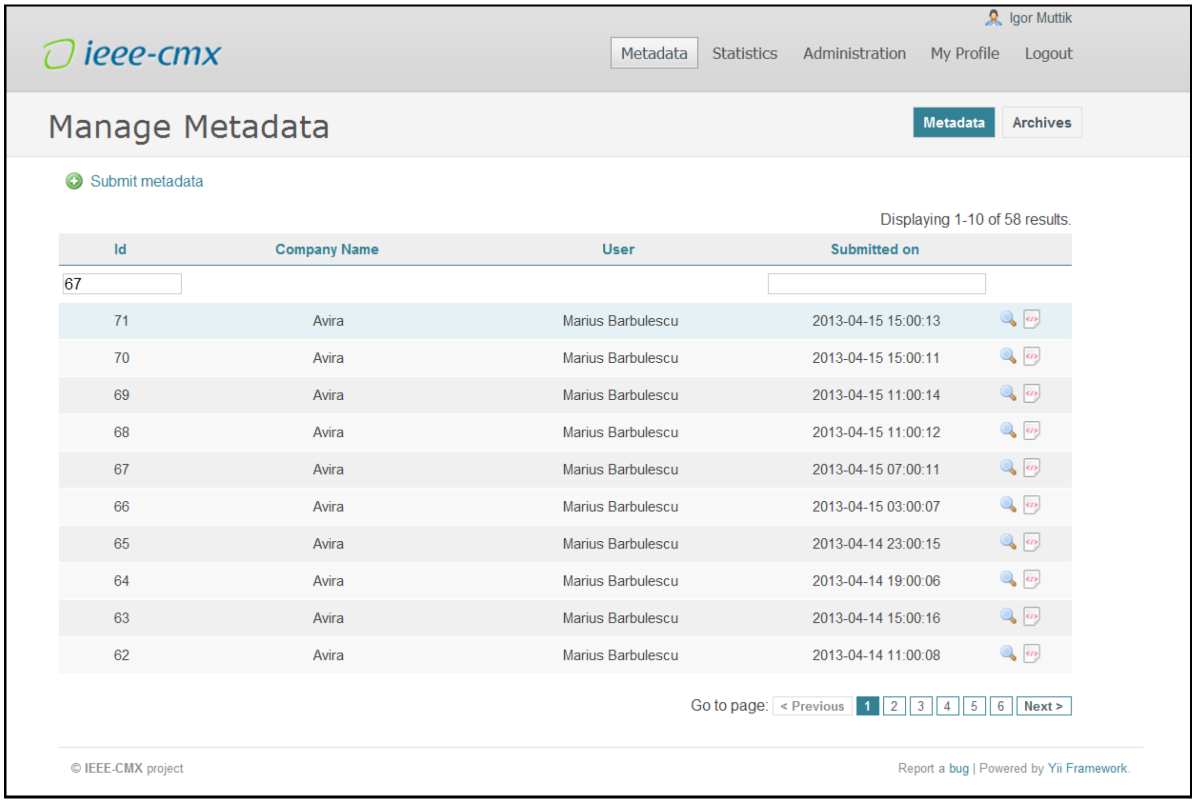
IEEE-CMX C++ Client - for Macintosh
A sample client for IEEE-CMX, written in C++ compiled for Macintosh.
Download Client

IEEE-CMX C++ Client - for UNIX/LINUX
A sample client for IEEE-CMX, written in C++ compiled for UNIX/LINUX.
Download Client

IEEE-CMX Python Client
Client for IEEE-CMX to generate and submit the Metadata written in Python
Download Client

© IEEE-CMX project
Report a bug | Powered by Yii Framework.

Downloads (documentation, tools and examples) [7].



Manage Metadata

Displaying 1-10 of 58 results.

Id	Company Name	User	Submitted on
67			
71	Avira	Marius Barbulescu	2013-04-15 15:00:13
70	Avira	Marius Barbulescu	2013-04-15 15:00:11
69	Avira	Marius Barbulescu	2013-04-15 11:00:14
68	Avira	Marius Barbulescu	2013-04-15 11:00:12
67	Avira	Marius Barbulescu	2013-04-15 07:00:11
66	Avira	Marius Barbulescu	2013-04-15 03:00:07
65	Avira	Marius Barbulescu	2013-04-14 23:00:15
64	Avira	Marius Barbulescu	2013-04-14 19:00:06
63	Avira	Marius Barbulescu	2013-04-14 15:00:16
62	Avira	Marius Barbulescu	2013-04-14 11:00:08

Go to page: < Previous 1 2 3 4 5 6 Next >

© IEEE-CMX project
Report a bug | Powered by Yii Framework.

Time-stamped metadata submissions.

Metadata submissions are sorted by the time of the receipt. Consumers can list all additions made since their previous visit. This simple transactional approach allows consumers of metadata to pick up all new records (from all sources) and ignore what they have already downloaded. Metadata is kept in ZIP archives (providing integrity checking of the data by testing the ZIP validity) attributed to a specific source.

```
<cleanMetaData xmlns="http://xml/metadataSharing.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xml/metadataSharing.xsd file:metadataSharing.xsd" version="1.200000"
id="10000">
 <company>N/A</company>
 <author>ZIP 1</author>
 <comment>Test MMDEF v1.2 file generated using genMMDEF</comment>
 <timestamp>2011-08-19T13:50:21.721000</timestamp>
<objects>
<file id="c7ae4ffe33fc841aea2e0113afa05fdf">
 <md5>c7ae4ffe33fc841aea2e0113afa05fdf</md5>
 <sha1>25daac9d19f18b5ac19769dcf7e5abc154768641</sha1>
 <sha256>e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855</sha256>

<sha512>cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2
b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e</sha512>
 <size>1546790</size>
 <filename>ProcessExplorer.zip</filename>
 <MIMEType>application/zip</MIMEType>
</file>
<file id="d22ff2cc70fa2eec94aaa6c6f49e6eb0">
 <md5>d22ff2cc70fa2eec94aaa6c6f49e6eb0</md5>
 <sha1>2458a3d696698e2c4550b91e54ff63f4b964198d</sha1>
 <sha256>6ff22c87fb5ee105b33346dbb3f13f3049a292981e9df1eb8591e858ccf4d782</sha256>

<sha512>34e18bf9679c71189383bcd89c9f723383715bbf63f051edd659c57e14d012987c33ba67fbbb0f
aeca962b3ec7b12b0aa24b3c134ddbb9f905aa26604718f375</sha512>
 <size>7005</size>
 <crc32>1185414000</crc32>
 <filename>Eula.txt</filename>
 <filenameWithinInstaller>Eula.txt</filenameWithinInstaller>
 <MIMEType>text/plain</MIMEType>
 </file>
<file id="ae846553d77284da53abcd454b4eaedf">
 <md5>ae846553d77284da53abcd454b4eaedf</md5>
 <sha1>782333340d56a8f020a74bd2830e68f31310a5b7</sha1>
 <sha256>a8f4bd956816960691bc08bf94be342a6d62bf6d91c92f7e7506903ffda50b83</sha256>

<sha512>bf1fd7b27234d5605731d21358bba01738098cb363a6deee79ed88699a3927b3a09d9044767d
b24ff6ddd028fbe0f4a1572f9af4d4ab4996bfdefe2b950a9b49</sha512>
 <size>72268</size>
 <crc32>2807815698</crc32>
 <filename>procexp.chm</filename>
 <filenameWithinInstaller>procexp.chm</filenameWithinInstaller>
</file>
<file id="4edc50d3a427566d6390ca76f389be80">
 <md5>4edc50d3a427566d6390ca76f389be80</md5>
 <sha1>9cb1bd5dc93124f526a1033b1b3f37cc0224a77e</sha1>
 <sha256>e942d28c0e835b8384752731f1b430cb3fbd571381666ded7637a2db47fafcc0</sha256>

<sha512>3ceb1bd07af9e470ff453ef3dd4b97f9228856cb78eb5cddb7b81796b4b830368e3ed2f0c6a9ce
93009397e8158c68dba67e398f58df87137d8872cb0bb3b53b</sha512>
 <size>3412856</size>
 <crc32>1119775926</crc32>
 <filename>procexp.exe</filename>
 <filenameWithinInstaller>procexp.exe</filenameWithinInstaller>
 <MIMEType>application/octet-stream</MIMEType>
```

```xml
  </file>
<softwarePackage id="procexp">
  <vendor>Sysinternals</vendor>
  <product>Process Explorer</product>
  <version>14.11</version>
  <language>English</language>
  </softwarePackage>
  </objects>
<relationships>
<relationship type="createdBy" id="1">
<source>
  <ref>file[@id="c7ae4ffe33fc841aea2e0113afa05fdf"]</ref>
  </source>
<target>
  <ref>file[@id="d22ff2cc70fa2eec94aaa6c6f49e6eb0"]</ref>
  <ref>file[@id="ae846553d77284da53abcd454b4eaedf"]</ref>
  <ref>file[@id="4edc50d3a427566d6390ca76f389be80"]</ref>
  </target>
  <timestamp>2011-08-19T13:50:21.924000</timestamp>
  </relationship>
<relationship type="partOfPackage" id="2">
<source>
  <ref>softwarePackage[@id="procexp"]</ref>
  </source>
<target>
  <ref>file[@id="d22ff2cc70fa2eec94aaa6c6f49e6eb0"]</ref>
  <ref>file[@id="ae846553d77284da53abcd454b4eaedf"]</ref>
  <ref>file[@id="4edc50d3a427566d6390ca76f389be80"]</ref>
  <ref>file[@id="c7ae4ffe33fc841aea2e0113afa05fdf"]</ref>
  </target>
  <timestamp>2011-08-19T15:50:21.924000</timestamp>
  </relationship>
  </relationships>
  </cleanMetaData>
```

Example metadata.

Apart from submitting and retrieving metadata, the CMX portal can also manage users. Each company (either a submitter or a consumer) can appoint a superuser capable of adding or removing other users from the same company.

Each piece of submitted metadata has an associated company name that allows consumers to apply different trust levels depending on the source of the metadata. Naturally, some consumers will also be providers of the data. (For example, Microsoft, Symantec, and McAfee/Intel are big software publishers and all have their own security products too.)

The system provides API-level access so it is easy to integrate with all sorts of automation—both for providers and for consumers of the metadata.

Cost of the system

For providers, the system is free (to stimulate the participation of software publishers and quickly grow the CMX repository). Providers are selected by invitation of the CMX Board of Directors. A third party can request to participate, and the CMX Board of Directors will make the final decision. The reason for this filtering is because there is no way to vet the data at the time of input. The system relies on the trust in that third party and its practices. It is possible to revoke a previously submitted entry (if, for example, it is found to have been infected prior to the metadata's being gathered). Moreover, should a particular third party vendor be found to be unreliable, then its access to the system can be revoked.

For consumers there will be two costs. The first is for the real-time access to retrieve the additions to the CMX data. This price covers the expense of the bandwidth consumed by the system. The second cost is for access to archives that predate the consumer's joining the system and will be based on a per-megabyte basis. This allows consumers to catch up with data created prior to their joining.

The exact costs are not yet determined, but IEEE is a nonprofit organization; so we expect costs to be moderate. Moreover, they will go down as the number of consumers increases.

The Future

The system is currently being populated with metadata describing only clean programs. One possible extension to the system would be to also share metadata of malware. Technically, this would be a trivial step but, unfortunately, it is quite complicated due to the competitive nature of security business.

Another possible extension could provide actual files (including installation packages). This sounds even less likely given that there are legal restrictions on distributing, redistributing, and providing access to copyrighted or commercial software. Hosting installers would contradict the licensing agreements of many programs and is likely to be too complicated legally. Many software publishers would be reluctant to allow security vendors to allow their installers to be downloadable by multiple consumers, even if the list is limited to the designated employees of security companies. The risk of software piracy and associated tainting of the IEEE ICSG image may be too great.

We also intend to look into expanding the system to cover other software formats, for example, Android installation packages (APKs).

Acknowledgements

We are extremely grateful to the Avira team for their efforts in implementing the CMX system and hosting it on their servers. Special thanks go to Philipp Wolf and Thomas Wegele, who organized the development, coded the system, and provided documentation.

Appendix. IEEE ICSG: Who We Are

In this section we describe how antimalware companies work together to solve problems that no single company can achieve in isolation.

The IEEE ICSG was formed as a global group of computer security entities that have come together to pool experience and resources in combating the systematic and rapid rise in computer security threats. The key to the success of the ICSG is cooperation among competing security vendors. Why would competitors cooperate? To solve common problems affecting the global user community.

During the past few years, attackers have shifted away from mass distribution of a small number of threats to micro distribution of millions of distinct threats. ICSG was established, under the umbrella of the IEEE-SA Industry Connections program, to more efficiently address these growing threats in a coordinated fashion.

IEEE ICSG creates working groups (WGs) focused on problems that the majority find the most important. Representatives of security companies operate these WGs. There are multiple ICSG WGs operating:

- Malware WG
- Malware Metadata Exchange Format (MMDEF) WG
- Stop-eCrime WG
- Taggant System WG
- Privilege Management Protocol WG

The CMX system was developed by the IEEE ICSG's Malware WG [3]. The Malware WG's aim is to solve some of the malware-related challenges the industry faces today.

The ICSG's Malware WG has already completed two efforts. One is fully in use, while the other is just now coming online.

The first effort was the XML sample-exchange schema. Many security vendors exchange malware samples. (The IEEE is not the only place the security vendors cooperate.) However, these have been private relationships and were ad hoc in nature. As more and more vendors have started to share, this process has become somewhat unwieldy. The IEEE MMDEF XML allows vendors to standardize, which greatly simplifies automated sharing tasks, and leads to better internal efficiency. This Schema has been available to antimalware companies since 2009 and was published by IEEE in 2010. It has now been extended under the name MMDEF-B to cover behavioral attributes.

The next effort is the Software Taggant System. This system addresses the problem of binary packers, a favorite of malware authors implementing server-side polymorphism. This system has packers inserting a “taggant”[1] automatically into each file packed. The taggant is a cryptographically secure element that allows security companies to uniquely and positively identify the packer license key used to create the packed files. These license keys (associated with the individual packer installations) can be blacklisted, rendering that installation useless. As these packers are rather expensive, this system attacks the economics of malware authors.

References

[1] “Symantec Internet Security Threat Report: Trends for July–December 2007,” http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_exec_summary_internet_security_threat_report_xiii_04-2008.en-us.pdf

[2] http://standards.ieee.org/develop/indconn/icsg/

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

[4] http://standards.ieee.org/develop/indconn/icsg/malware.html

[5] MUTE group, http://mutegroup.org/

[6] “MUTE—Malware URL Tracking and Exchange,” http://mutegroup.org/MUTE_AVAR.pdf

[7] https://ieee-cmx.avira.com/documents

[1] A **taggant** is … a chemical or physical marker added to materials to allow various forms of testing. It is believed that they generally consist of microscopic particles built up in many layers, which are made of different materials. It is a somewhat secretive process, but products that may be affected include ink, paper, perfume, and medication. Taggants allow testing marked items for qualities such as lot number and concentration (to test for dilution, for example). In particular, taggants are known to be widely used in plastic, sheet, and flexible explosives. (http://en.wikipedia.org/wiki/Taggant)