Issues Involved in the "Creation" of Samples for Testing

Anti-Malware Testing Standards Organization

Notice and Disclaimer of Liability Concerning the Use of AMTSO Documents

Issues Involved in the “Creation” of Samples for Testing

Introduction

This document discusses the issues involved in the “Creation” of samples for testing. The document outlines additional issues involved in best practice testing of such products, above and beyond other AMTSO guidelines and best practices. This document is not a comprehensive listing of all such issues.

Unless otherwise defined herein, all terms included in this document are used with their common meaning. The following document should be read in conjunction with AMTSO’s *Fundamental Principles of Testing, AMTSO Best Practices for Dynamic Testing, AMTSO Best Practices for Validating Samples, AMTSO Best Practices for Testing In-the-Cloud Security Products*, and other information available at www.amtso.org.

Overview

One of the most hotly-debated issues in the anti-malware industry today is the question as to whether it is ever right to create a new piece of Malware for the purpose of testing anti-malware software. There are fiercely held positions on both sides of the issue, some of the points of contention dating back decades. This document will attempt to take a clinical approach to the issue and examine the issues and ramifications for Sample Creation.

The format will be in the form of a debate. For each point, an argument in favor and in opposition will be made, followed by a rebuttal to each position where there is a case for such a rebuttal. This will provide the reader with the information he needs to make an informed decision.

This document discusses “Creation” solely in the context of testing. Under **no** circumstances does AMTSO support or condone the public release of malware samples.

> **Note: The arguments in favor of an item and the rebuttals to the “arguments in opposition” shall not be taken as meaning that AMTSO does or doesn’t support these positions. They are offered to frame the debate and to place the arguments into context**

Safe Handling

Whenever handling samples in a testing lab – whether they be “Created” or not – care must be taken that these samples not leave the lab. This is especially important given that many security products require an active internet connection to function.

What is Creation?

The first issue to tackle is a very basic one: what is meant by "Creation"? Much of the heat the problem stems from this question. To some, Creation means writing from scratch new Malware – "malicious" programs never seen before. To others, Creation means a trivial modification such as the changing of a single bit. Let us examine whether the types of modifications listed below are "Creation".

Archiving (ZIP, ZIP-Self Extractor, Installer)

There are numerous archiving utilities, and some of these support "self-extraction" and "self-execution." The question discussed here is whether taking an existing sample of malware and placing it into an archive – with any of the options described – is "Creating" malware.

Arguments in Favor of Archiving as Creation:	Arguments in Opposition to Archiving as Creation:
If the archiver supports installation type scripting then this adds functionality, and therefore is Creation. If self-extraction leads to execution, whether immediate or delayed (as may happen when an executable is dropped to an autostart location), then this is also Creation.	Malware is more than a specific sequence of bytes. It includes functionality and capability. Since a sample placed into an archive – even one with self-extraction – adds no additional functionality or capability, it is not Creation. The usefulness of archiving a sample is questionable, as only the most rudimentary anti-malware solution would fail to detect it.

Packing/Repacking

Binary packers (programs to compress executables) are used by the majority of malware. The question discussed here is whether taking an existing sample of malware and packing (or repacking) with a binary packer is "Creating" malware.

Arguments in Favor of Packing as Creation:	Arguments in Opposition to Packing as Creation:
If the packer in question were to add functionality – such as anti-debugging or active protection – then it is creation.	Malware is more than a specific sequence of bytes. It is functionality and capability. Since a sample repacked with a binary packer adds no additional functionality or capability, it is not "Creation". The usefulness of repacking a sample is questionable, as the test objective would be addressed only if the solution supports that packer. It should be noted that if the packer in question were to add functionality – such as anti-debugging or active protection – then it is creation.
	Rebuttal to Arguments in Opposition of Packing as Creation:
	If a virus spreads by copying the on-disk image, then repacking would be creating a new variant

Using a Malware Generation Kit

There are any number kits available to package a "payload". These tools can contain exploits to get the code onto the target box, and options to disable security software. The question discussed here is whether using a Malware creation kit to generate samples – regardless of payload – is "Creation".

Arguments in Favor of the Use of a Malware Generation Kit as Creation:	Arguments in Opposition to the Use of a Malware Generation Kit as Creation:
This results in a new variant, and is therefore creation.	This is a more complicated question. If the payload implemented with the kit is pre-existing then new functionality is not created. In the case where a new payload is crafted for the kit, then this would constitute "Creation".

Server-Side Polymorphic

A tester downloads samples from a public server which serves different versions periodically. Does this constitute creation?

Arguments in Favor of Server-Side Polymorphic as Creation:	Arguments in Opposition to Server-Side Polymorphic as Creation:
This is the same as using a malware generation kit, and should be considered the same.	If the system generating the samples is public, then this is not creation.

Patching

Patching is defined as making a binary change to an existing file. This can involve changes to the PE header, appending bytes, making changes in empty portions of the file, etc. The question here is does patching an existing malware sample constitute "Creation"?

Arguments in Favor of Patching as Creation:	Arguments in Opposition to Patching as Creation:
Any modification to a binary is creating a new variant. If the file is self-replicating, it would be creating a new variant.	Malware is more than a specific sequence of bytes. It includes functionality and capability. Presuming the patching adds no additional functionality or capability, it is not "Creation". Since what will load and execute on a given operating system is always evolving, this can be an effective technique to determine if a given security product has weaknesses in its PE parsing.
	Rebuttal to Arguments in Opposition of Patching as Creation:
	If a virus spreads by copying the on disk image, then patching would be creating a new variant.

Using a Pre-Existing Polymorphic Engine

This would be a specific case of the Malware Kit scenario discussed above.

Using a Custom Packer or Polymorphic Engine

Testers could write their own custom polymorphic engine to obfuscate code. This could involve anti-emulation tricks or other such items. The question here is whether the processing of existing malware samples by a custom packer or polymorphic engine constitutes "Creation"?

Arguments in Favor of Custom Polymorphic Engines as Creation:	Arguments in Opposition to Custom Polymorphic Engines as Creation:
This creates a new variant.	If this does not produce new behaviors, then it is not creation. If it does exhibit new behavior – such as anti-emulation – then it could be construed as "Creation".
	Rebuttal to Arguments in Opposition of Custom Polymorphic Engines as Creation:
	If a virus spreads by copying the on disk image, then patching would be creating a new variant.

Writing a New Sample Using Existing Techniques

There are known mechanisms by which malware may perform its actions or spread. The question here is, does writing a new sample using existing techniques constitute “Creation”?

Arguments in Favor of a New Sample Using Existing Techniques as Creation:	Arguments in Opposition to a New Sample Using Existing Techniques as Creation:
This is undoubtedly “Creation”.	No argument. This is undoubtedly “Creation”.

Writing a New Sample Using New or Previously Unknown Techniques (Proof of Concept)

Some security products claim to protect against broad classes of existing and future malware. New techniques are sometimes anticipated by security vendors, who may put them into practice as Proof of Concept (PoC) code, in order to test their own or competitors' products. The question here is whether writing new samples using previously unknown techniques constitutes "Creation."

Arguments in Favor of a New Sample Using Previously Unknown Techniques as Creation:	Arguments in Opposition to a New Sample Using Previously Unknown Techniques as Creation:
This is undoubtedly "Creation". Moreover, these samples could serve as guides for malware authors, and thus should never be publicly disclosed.	No argument. This is undoubtedly "Creation". Moreover, these samples could serve as guides for malware authors, and thus should never be publicly disclosed.

Reasons for Creating Samples

1) To test the heuristic/proactive capabilities of anti-malware products against malware for which there are no signatures.

A tester wants to test the heuristic/proactive capabilities of anti-malware against malware for which there are no signatures.

Arguments in Favor of Item 1:	Arguments in Opposition to Item 1:
Many new anti-malware products tout their ability to detect and block never before seen threats. This is an augmentation to their traditional signature based solutions. In order to test the claims and effectiveness of these products it is necessary to check them against malware for which one can guarantee there is no signature. The best way to guarantee this is for the tester to create the sample himself. It goes without saying (though it is being said) that extreme care must be taken with the samples created that they never leave the lab. The "malicious" functionality must be suitably contained to ensure that the malware not function outside the lab. This requires extraordinary efforts since many anti-malware products now rely on an active network connection in order to function.	There is no shortage of real malware in the world. If a tester wishes to test the proactive/heuristic capabilities of an anti-malware product (and we strongly encourage the testing of this functionality) he need only obtain fresh samples from the field. Computer users have little problem finding this new malware. The tester – who is significantly more sophisticated than the average user – should therefore have no trouble finding fresh samples (over 45,000 unique samples were discovered per day in 2009). The artificial creation of malware samples is unlikely to accurately reflect what is happening in the real world. The nature and type of the created malware can be a variance with the current malware space. Moreover, should the tester provide these samples to anti-malware companies as "missed samples", these companies might bloat their already large signature and collection sets with these files that no customer has seen – or will ever see. Additionally, so called "retrospective" tests, where products and signature sets are frozen, and tested against subsequent samples of malware can also exercise this functionality within a product.
	Rebuttal to Arguments in Opposition of Item 1: Given the advent of cloud based technologies, so called "retrospective" tests are either not possible, or cannot be proven to reflect whether or not a given product would have detected a given threat at the point it was introduced.

2) To test the support of anti-malware products for specific packers

Many anti-malware products contain specialized code to unpack known dual-use and malicious packers. If we take a sample that we know is detected by product X and re-pack it with packer P, we can then determine by scanning the new object whether product X supports packer P.

Arguments in Favor of Item 2:	Arguments in Opposition to Item 2:
Most malware today is packed. An anti-malware product’s ability to deal with these binary packers is an important part of its efficacy. A good way to test this is to take threats the product has demonstrated it can detect in their original form and re-pack them with known packers, using various packing options. If the product still detects these samples, great! If it does not, then that could be construed as an important deficiency.	The practicality of the test described above is low. Gathering fresh samples packed with the packers in question, tested against frozen definitions, and would yield equivalent results.
Rebuttal to Arguments in Favor of Item 2:	**Rebuttal to Arguments in Opposition of Item 2:**
Generating “new” samples that are actually combinations of base code and packer that may never actually come into being otherwise results in additional and unnecessary sample glut.	It might not always be practical to scour the internet for samples packed with a particular packer in order to test a product’s proficiency with a given packer. This can more quickly be determined by specifically packing original samples the product is known to detect.

3) To “Mimic” what the bad guys do.

Packing a piece of Malware is valid because “the bad guys are going to pack it, so this simulates the real world.”

Arguments in Favor of Item 3:	Arguments in Opposition to Item 3:
As stated above, most malware is packed today. If the first versions of some new threat are not packed, they soon will be. Packing is the easiest way to implement server side polymorphism – a common technique utilized by malware authors to defeat signature based solutions. Simulating in the lab what happens naturally in the field is a valid approach, provided all the possible precautions are taken to ensure the malware never leaves the lab.	This item is basically self-defeating. If the bad guys are going to do this, then the tester need only wait for them to do so and use that sample. If the bad guys do not do it, then it is not really mimicking them. Additionally, much malware today is packed with custom packers. The tester would not have access to these, and thus cannot create samples with them. By not including these custom packers in a test will not accurately reflect the real world.
	Rebuttal to Arguments in Opposition of Item 3:
	Packing a piece of real malware and testing a product’s ability to detect the sample is a quick way to test out how well it handles the general detection of that malware. More timely results can be obtained, because the tester does not need to wait for a “bad guy” to do it, and then to obtain that sample.

4) To test detection of a malware kit's output.

There are any number of Malware Kits available on the internet. These can create specific features (stealth, exploits, etc.) around some specific payload. A tester may wish to determine a product's generic detection capability by creating packages and testing effectiveness (dynamically or statically).

Arguments in Favor of Item 4:	Arguments in Opposition to Item 4:
Much of the malware created today is generated from so called Malware Creation Kits. These kits will take a payload and package them up inside various exploits or functionality. Obtaining these kits, using them to create threats, and testing anti-malware products against them is a good way to simulate that product's expected performance in the field.	The practicality of the test described above is low. Gathering fresh samples generated with the kits in question, tested against frozen definitions, and would yield equivalent results. Not indicative of real world. Purchasing a kit provides support to the underground economy.
	Rebuttal to Arguments in Opposition of Item 4:
	It might not always be practical to scour the internet for samples generated with a particular kit in order to test a product's proficiency with a given kit. Plus, it might be difficult for a tester to know whether a given sample was created by a kit, whereas if he creates it himself he will be certain that this is the case.

5) To beef up a test set.

Have 1,000 samples? Want 10,000 samples? Why not use 10 packers and turn your 1,000 into 10,000?

Arguments in Favor of Item 5:	Arguments in Opposition to Item 5:
Given that some purchasers of tests desire a certain number samples, generating them yourself makes it easy to meet these numbers without gathering them.	While this might seem on the surface to be a valid thing to do, it really will demonstrate nothing more than the information gained in 2). To test the support of anti-malware products for duplicating samples adds no additional information.
Rebuttal to Arguments in Favor of Item 5:	
In reality, you only have the number of samples you started with, the others are functionally equivalent, and therefore of little value.	

6) Because packing samples are “extra fresh”.

They’re fresh! I made them myself!

Arguments in Favor of Item 6:	Arguments in Opposition to Item 6:
If samples are newly made then a tester can guarantee that no vendor has seen them.	The arguments against this have already been covered in 1) To test the heuristic/proactive capabilities of anti-malware products against malware for which there are no signatures. These “fresh” samples are artificial, and as such are likely not representative of what is happening in the real world. Real world samples can be easily obtained, obviating the need to artificially create “fresh” fake ones.
Rebuttal to Arguments in Favor of Item 6:	Rebuttal to Arguments in Opposition of Item 6:
In reality, you only have the samples you started with, the others are functionally equivalent, and therefore of little value.	Packing a piece of real malware and testing a product’s ability to detect the sample is a quick way to test out how well it handles the general detection of that malware. More timely results can be obtained, because the tester does not need to wait for a “bad guy” to do it, and then to obtain that sample.

7) To gain independence.

Testers get their many of their samples by sharing with vendors. This source of samples, when samples come from the vendors whose products are being tested, can call into question the independence of test results. If those samples are modified, then they are no longer the same ones provided by that vendor and have thus achieved independence.

Arguments in Favor of Item 7:	Arguments in Opposition to Item 7:
It is quite common for anti-malware vendors and testers to share samples. This could result in tests that are somewhat biased in favor of one or more vendors. One way to offset that is create a set of samples – thus guaranteeing that no vendor has them in their collections. This could allow for a more independent test.	While the concern expressed within this item is valid, this is not the proper way to address it. Accepting samples and/or URL's from a particular vendor might bias the test set towards that vendor. The better way to avoid this potential is to obtain samples from an independent 3rd party (for example, independent security research firms) or to obtain the samples and/or URLs yourself.
	Rebuttal to Arguments in Opposition of Item 7:
	While this argument may be true in an academic sense, not every tester will have the resources to obtain a representative set of URL's. Since the anti-malware vendors may also be obtaining samples from the same 3rd parties while others do not, this bias might still be present.

8) Because it is cheap and fast.

Modifying existing samples through the techniques mentioned is relatively cheap and fast.

Arguments in Favor of Item 8:	Arguments in Opposition to Item 8:
Some testers have very limited resources. Additionally, when testers are trying to establish themselves, they will not have access to the sample sharing discussed in Item 7.	While it is relatively cheap and quick to artificially create certain types of samples, the reasons for not doing so have been extensively covered in the Items above.
Rebuttal to Arguments in Favor of Item 8:	
A credible tester (for example, a member of AMTSO) will have access to sample sharing efforts within the **a**nti-malware industry. The convenience of the testing organization should not be seen as more important than its responsibility to provide an accurate test.	

9) To make it easy to keep a test focused.

Product X claims that it heuristically detects/blocks, for example, Keyloggers. To determine the full scientific validity of the claim one could create a series of Keyloggers and test whether product X can detect/block them.

Arguments in Favor of Item 9:	Arguments in Opposition to Item 9:
Products often make broad claims, such as “proactively protects against all Keyloggers”. In this specific case, Keyloggers use a variety of techniques to grab the keys, and new techniques are often thought of. If a new technique is invented, it might take some time before some malware author decides to employ it. Moreover, Keyloggers are among the stealthier malware, so finding samples may take a while even after a real threat emerges. Some proof of concept ideas may be discussed in publications. It is legitimate to test the product claims by implementing these techniques and verifying if the product does, in fact, protect against it. Rootkits are another area where claims have been made and are valid to test. As always, extreme care must be taken with the handling of these samples. Moreover, release of details regarding successful PoCs may violate Principle 1: Testing must not endanger the public.	While this test may be valid, the conclusions must be limited to the focused area. Certain Proofs of Concept (PoCs) might be so improbable that companies might divert resources to a threat that might never emerge.
Rebuttal to Arguments in Favor of Item 9:	Rebuttal to Arguments in Opposition of Item 9:
There could be a high degree of risk involved in developing PoC’s, particularly if there is self-replication.	If anti-malware vendors address PoCs early, the actual threat might never emerge.

____

This document was adopted by AMTSO on October 13, 2009