

STANDARDS AND POLICIES ON PACKER USE

Samir Mody

Sophos, The Pentagon, Abingdon Science Park,
Abingdon, OX14 3YP, UK

Email samir.k.mody@googlemail.com

Igor Muttik

McAfee Avert, Alton House, Gatehouse Way,
Aylesbury, Bucks HP19 8XD, UK

Email mig@mcafee.com

Peter Ferrie

Microsoft, USA

Email peferrie@microsoft.com

ABSTRACT

Packers, whether third-party or bespoke, are still widely used by malware authors in an attempt to evade detection. Conficker, FakeAV, Bredolab and TDSS are but a few examples of malware which make extensive use of packers.

The wide variety of packers used for both legitimate and malicious purposes pose a challenge for the anti-virus industry. The anti-virus community has decided, within the framework of the Malware Working Group (MWG) within the Industry Connections Security Group (IEEE ICSG <http://standards.ieee.org/prod-serv/indconn/icsg>), to address the issue of packers with a common voice.

In addition, the stigma and the anti-virus detections associated with the use of legitimate packers by malware, along with the performance impact related to scanning benign packed files, are likely to lead to an impact on both the reputation and revenue of the packer vendors involved.

Therefore it is in the best interests of both parties to work together to identify and implement solutions to the core issues associated with packers.

One of the fruits of the collaborative IEEE ICSG sessions, involving representatives from across the anti-virus industry, is a document describing various packer properties and standards for their use. This document is intended to provide a yardstick for the formulation of policy on how to treat different packers and a potential set of best practice guidelines for packer vendors. The specific contents of the document are subject to the outcome of negotiations with packer vendors.

It is hoped that the guidelines can be used to improve end-user security through the concerted efforts of the anti-virus industry when dealing with packers, and via cooperation and information exchange with packer vendors. Thus it is expected to facilitate a more robust approach to the generic static flagging of suspicious packed files for the benefit of all (other than the malware authors, of course).

GLOSSARY OF SALIENT TERMS

Certain core terms are used, many frequently, in subsequent sections and therefore deserve concise definitions in the context of this document.

- **Packer:** a packer is a piece of software that creates a 'software envelope' around an executable object, modifies its natural executable form and retains the original functionality of this object at runtime solely in memory.¹
- **Packing application:** see 'Packer'.
- **Target:** an executable object in its natural form prior to being operated on by a packer.
- **Packed file:** a target which has been operated on by a packer. Also referred to as 'Packed object' or 'Packed target'.
- **Software envelope:** may be an unpacking layer created around a target by a packer. The code of the unpacking layer represents an important part of a packer's 'DNA'. Another form of a software envelope consists of instructions from a target file transformed into p-code interpreted at runtime. This list of examples is not exhaustive and there could be other implementations of software envelopes.
- **Modification of the natural executable form:** examples would be modifying the target's original instructions (or their combinations) into an equivalent but different form, or patching the target and/or adding 'do-nothing' instructions (such as NOPs or JMPs). This list of examples is not exhaustive and there could be other implementations of software envelopes.
- **Obfuscator:** a packer which employs code obfuscation techniques to deliberately render more difficult the analysis of the packed object via a certain class of methods or tools for analysing code.²
- **Compressor:** a packer whose sole purpose is to make the target smaller on disk.
- **Anti-emulation:** a class of techniques intended to obfuscate code against emulators.
- **Anti-debugging:** a class of techniques intended to obfuscate code against debuggers.
- **Junk code:** code patterns which serve no practical purpose other than as an obfuscation technique. Polymorphic code patterns can sometimes form a subset of junk code.
- **Packer characteristic:** individual descriptive feature/aspect of a packer.
- **Packer property:** descriptive functional aspect of a packer formed out of a superset of packer characteristics.
- **Watermark:** encoded unique identifier permeated through a packed target.

¹ From 'IEEE ICSG Definition of Terms' – a draft document internal to the IEEE ICSG.

² Based on a definition given in the 'IEEE ICSG Definition of Terms' document.

- Taggant: set of cryptographically encrypted unique identifiers placed at a designated location in a packed target. Taggants can be considered to provide the same information as a watermark in a cryptographically secure way.

HIGH-LEVEL CHECKLIST FOR SECURITY VENDORS

This section describes the sequence of checks which security vendors are likely to apply when evaluating whether a file generated by a packer should be blocked.

The flow chart in Figure 1 depicts the actions and decision logic which may be summarized as follows:

1. Apply digital signature checks and exclude files signed by trusted vendors.
2. Identify the packer family/version (should not require unpacking).
3. Match against rule data based on statistics which show how frequently this packer is misused in order to obfuscate malware.
4. If statistics warrant it, detect the file or apply a security policy.
5. Extract the user licence key information (should be reasonably quick and should not require unpacking).
6. If statistics indicate a significant share of benign programs obfuscated by the packer, analyse its licensing model, and if the file is packed with 'leaked'/'misused' keys, block as potential malware.
7. If all the above checks fail, analyse the properties of the packer and decide if the combination of the obfuscation methods yields a decision to detect the file or apply a security policy.

This decision-making process, especially with regard to any specific packer, may be subject to continuous evolution. There is likely to be an appeal process. If a legitimate complaint arrives from a packer vendor or a user of a specific blocked packer, it is likely to be investigated in a reasonable time frame, with appropriate action being taken, including giving feedback.

USE OF DIGITAL SIGNATURES TO AUTHENTICATE PACKED OBJECTS

Verified digital signatures certified by a known, trusted organization provide a robust way to trace the source of files.

Based on the fact that responsible creators of software and trusted certification authorities are unlikely to digitally sign malware, the presence of verified digital signatures would allow a much higher degree of confidence in a file regardless of its contents. Wide use of valid digital signatures will aid security vendor scanners to make quick but robust decisions on all files, including those which are packed.

Caveats:

1. The mere presence of digital signatures is insufficient. They need to be actively verified since some malware can and do

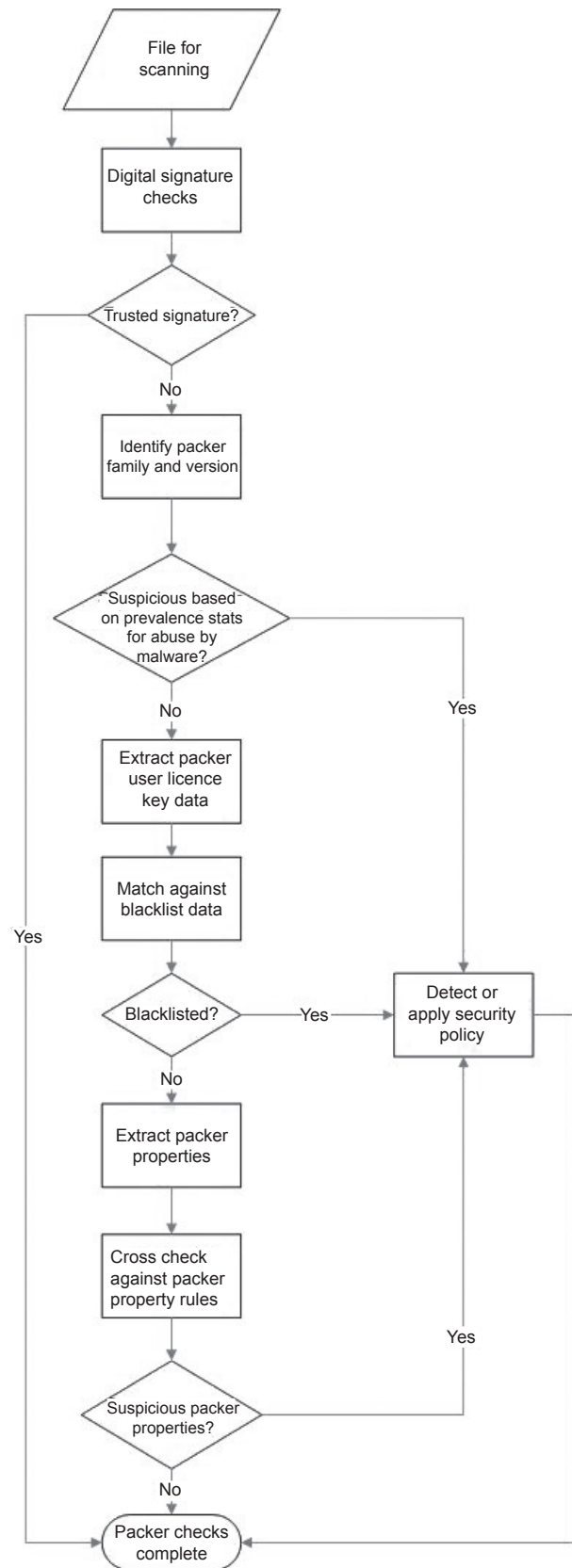


Figure 1: Flow chart depicting actions and decision logic.

spoof signatures (for example, malware frequently spoofs version strings like `CompanyName`, etc.). It is important to note that there have been known cases of W32/Induc infections being validly digitally signed by legitimate software manufacturers due to this virus embedding itself into programs at the compilation stage.

2. A high degree of confidence in the creators of the digital signatures is required. This level of confidence will need to be maintained ad infinitum.
3. The onus for applying digital signatures falls on the packer user.
4. Digital signatures tend to be most commonly applied to *Windows Portable Executable* (PE) files, although they can be used to authenticate various types of electronic media on multiple operating systems. However, given the nature of the threat landscape it is the use of digital signatures on PE files which is particularly relevant to this document.

USE OF WATERMARKS, TAGGANTS OR LICENCE INFORMATION TO AUTHENTICATE PACKED OBJECTS

As an alternative to the use of digital signatures, it is possible to authenticate a packed executable based on the credentials provided by the packer vendor. Packer vendors could include encoded or encrypted licence key information and share known leaked licences to permit the flagging of packed files containing blacklisted licences via either a taggant or watermarking solution. The concept of taggants, which are similar in purpose to watermarks, is rather new with an incipient design and implementation framework being mooted within the IEEE ICSG. However, watermarking solutions already exist in certain packer systems. For example, at least one responsible packer vendor includes two sets of watermarks in a packed executable, one to identify the packer (public watermark) and one to uniquely identify the licensee (private watermark). This packer vendor is also willing to share information related to leaked private watermarks.

Caveats:

1. Watermarking currently tends to apply only to commercial packers.
2. Active and in-depth packer-vendor/AV industry cooperation is necessary ad infinitum. AV vendors may need to constantly provide new malware samples to obtain blacklisted licence information.
3. It is possible for malware writers to obtain licence keys. This is somewhat mitigated by employing a policy of 'strike one, you're out' on licence keys so that only one known instance of a licence key in malware is sufficient to blacklist it.
4. Depending on the method of encoding the licence key information it may be possible for malware authors to spoof or sabotage them. This can be mitigated by ensuring that any tampering with the licence information results in a failure on the part of the packer to execute the target object.
5. This method may be difficult to apply to non-PE files.

The use of private and public watermarks, or taggant information, provides the security industry with a reliable, robust and potentially performance-efficient way to authenticate packed files. In fact, if a standard method of applying a taggant is adopted by many/all packer vendors, there are additional performance enhancements. Therefore it is worth digressing to explore the subject in some detail.

The typical scenarios for the potential abuse of commercial packers and the need for watermark or taggant verification by malware may be described as follows:

- Alice writes a packing application and sells a copy to Bob. Bob uses it to protect legitimate applications. Carol, an AV vendor, needs a way for her product to confirm automatically that these samples have been created with a legitimate packing application, and that neither the packing application nor the final executable has been tampered with.
- Dave, a malware author, obtains a copy of the packing application (maybe he stole Bob's copy, maybe he used stolen credit card details or maybe he was stupid enough to use his own). Dave starts releasing packed malware, and eventually one of Carol's customers is infected by a sample and sends it to Carol. Carol now needs a way for her product to spot any sample that has been created by this particular copy of the packing application.
- Dave notices that everything he makes with this copy of the packer is automatically detected, and starts modifying files – either the final samples or the packing application – so that whatever markers Carol was using for detection are no longer present, but the file will still run. Carol needs to be able to spot that these files are the result of tampering, and block them anyway. However, if Dave is persistent, eventually the files will no longer be recognizable as packed by Alice's packer, at which point Carol can detect them using a simple unknown-packer/obfuscation detection.

The high-level requirements for watermarking or taggant information requested from packer vendors are as follows:

1. A way to identify that a file is packed with said packer without any need for unpacking the packed file.
2. A way to identify the licence information format and its location without any need for unpacking the packed file.
3. A continuous provision of up-to-date blacklisted licence information.

The desirable requirements for the watermarks or taggants can be divided into those which are general purpose and those which are specifically related to authentication by security software.

General purpose requirements:

1. Unique to the user using the product.
2. Cannot be easily spoofed, which means that tampering with the protected application – especially with the watermarking or taggant content itself – ought to render the application inoperable. It may also mean that it should not

be obvious to malware authors where this information can be found, although ideally there ought not to be any reliance on obscurity to maintain the integrity of the infrastructure. Taggants encrypted using contemporary cryptographic methods provide a verification solution without the need for obscurity.

Security software-related requirements:

1. Cryptographically secure.
2. Can be accessed and verified without needing to penetrate any part of the protection layer.
3. Should not preclude the application of standard *Authenticode* digital signatures to the protected application, i.e. should be designed in such a way that it remains valid after adding a digital signature. The digital signature will, of course, protect the watermark (or potentially a taggant) from being tampered with.

The recognition of the need for unique-identifier-based authentication is so strong amongst security vendors that there are plans under way within the IEEE ICSG to sponsor the framework and software to implement a taggant infrastructure in a standard way, in cooperation with our packer vendor partners³. Taggants have been chosen over watermarks since the entire package of design, implementation, administration, and perhaps even security, for a taggant-based solution is deemed to be more

³Specific details are currently unavailable.

favourable than that for watermarks. The taggant framework may also be applicable to non-commercial packers. Perform the final implementation is likely to be based on a negotiated agreement amongst the parties involved.

PROPERTIES OF PACKERS

This section summarizes several properties of packers deemed to be of a suspicious or unusual nature. The definitions of a 'suspicious' or an 'unusual' nature in the context of packer properties, and whether the properties fall into these categories have been debated and ratified by the MWG. The list of properties is not intended to be exhaustive.

There is a plethora of different individual packer characteristics which are designed to effect the same resultant packer property, e.g. the packer property of anti-debugging can be achieved via multifarious packer characteristics. Hence this section will focus on properties of a packer which form a superset of the individual characteristics. Note, packer properties are not mutually exclusive insofar as there may be considerable overlaps in terms of their constituent individual characteristics.

Table 1 lists several packer properties. For each property, where appropriate, examples of constituent characteristics and known packers which exhibit the corresponding property have been provided (note, some packers have several aliases). Additional comments have been made where relevant.

Packer property	Example packer characteristics	Example packers	Comments
Polymorphic/junk code		Morphine, TeLock, ACProtect, Obsidium	Used by some obfuscators.
Excessive branching	Parallel branching 'Spaghetti code' joined by unconditional jumps	Execryptor, Obsidium, PESpin	Used by some obfuscators. Execryptor (and probably some others) use the 'parallel branching' method, i.e. multiple functionally equivalent branches that may or may not be taken depending on the state of irrelevant flags, to make it difficult to place breakpoints. More (e.g. Obsidium, PESpin) use 'spaghetti code', where the functions are chopped up and put all over the place, but are connected by unconditional jumps.
Excessive looping	Use of 'do-nothing' loops	Morphine	Used in some obfuscators. May be seen as an anti-emulation trick. However, delay/timing loops are a legitimate use of 'do-nothing' loops.
Unnecessary branching into the middle of another instruction	Unconditional jump into the middle of another instruction	PESpin, SVKProtect, Exe32Pack	One or two occurrences may not necessarily be an issue, e.g. if it is done to minimize the code size or instruction count. UPX might do it.
Unnecessary use of unusual instructions	Use of MMX and floating point instructions	NTKrn1, XTreamLok, KKrunchy	Not necessarily obfuscation every time. Some legitimate packers (e.g. KKrunchy) use MMX/SSE/SIMD instructions because they are smaller and faster than the regular x86 instruction set when it comes to decompressing data. The context of use is relevant.

Table 1: Packer properties.

Packer property	Example packer characteristics	Example packers	Comments
Unusual transfer of execution control	Setting Structured Exception Handlers (SEH) and forcing exceptions	Obsidium, Armadillo, ASProtect, Enigma, PECCompact	SEHs are used to catch runtime errors and should not be misused to obfuscate the execution flow.
Unusual structural features	Garbled Portable Executable (PE) section names or overlapping MZ and PE headers	MEW, UPack, FSG	Execryptor has garbled/randomized section names. UPack has non-aligned/overlapping sections to confuse the PE loader. This also includes the use of non-aligned/overlapping sections, and similar undocumented/non-obvious behaviours of the PE loader that put the entry point somewhere other than where a simple parser would expect to find it. However, Upack, KKrunchy, etc. may do it purely to save space so they can reuse header bytes for their own code or data, i.e. for the benefit, albeit minor, of compression. If there is no actual benefit from these structural changes, they may be considered suspicious.
Impersonation of another packer	Reproducing certain characteristics such as section names or EP code to pretend to be another packer	NSAnti	SVK-Protector may allow the user to choose section names and it may have been seen in other packer configuration GUIs.
Restricted environment compatibility	Use of hard-coded Virtual Addresses (VA) or artefacts of certain Operating Systems (OS) making the packer incompatible with other OSs Use of undocumented APIs and instructions Different execution outcome in native and emulated environments	ActiveMARK	Commercial packers are likely to be compatible across several OSs (generally <i>Windows</i>) and within VMs. Non-commercial packers are less likely to be widely compatible. However, ActiveMARK is a commercial DRM wrapper which may have assumed fixed memory locations so may be incompatible across all OSs.
Non-standard use of APIs	Doing something with APIs other than legitimately invoking them	SVK-Protector, Obsidium	
Destruction of target data	Destruction of digital signatures, version strings, etc. (without warning)		If a packer destroys a digital signature, or other important data structures such as version strings or header timestamp, etc., when packing a file this has to be considered data destructive and reduces the security of the packed object. This is unacceptable behaviour for a packer. In terms of digital signatures, packed files could be re-signed, or an explicit warning could be displayed by the packing application about the destruction of the target's digital signature. Commercial packers are likely to be more careful with target data. Non-commercial packers are less likely to preserve all important sections of target data.

Table 1: Packer properties contd.

Packer property	Example packer characteristics	Example packers	Comments
Virtual Machine (VM) detection	Querying of environment artefacts (registry strings, mutexes, processes, etc.) Incorrect/incomplete emulation of CPU behaviour (e.g. instructions that are too long in VirtualPC, undocumented instructions in Bochs and SandBox, writable read-only bits in eflags in SandBox, etc.)	ASProtect and SVK-Protector	Refusing to execute in a VM may be due to legitimate reasons such as enforced licensing, i.e. to prevent unlimited use.
VM disruption	Parallels crash via V86-mode SIDT with T flag set QEMU and VirtualBox hang via infinite double-faults		No examples of packers known to do this thus far.
VM rendering of target object	The original code is replaced with pseudo-code which is interpreted at runtime by a wrapper. At no time is the original code ever replaced or executed directly	VMProtect, Themida, ExeCryptor	Static obfuscation.
Anti-dumping	Structural modifications of the target in memory, e.g. header erasure, image size change, etc.	Armadillo, Yoda's Cryptor	
Anti-debugging	Querying fields modified by debugger, special APIs, hardware tricks Explicit targeted disruption of debuggers	ExeCryptor, Themida, MSLRH	
Anti-emulation	Spurious use of interrupts (INT 2e, INT 2d, INT 4, etc.)	NSAnti, Tibs, Lighty	Also includes the presence of calls to GetProcAddress for non-existent functions.
Unique packer identifier	Presence of watermark or licence key information	VMProtect	Tends to exist only for some commercial packers. Access to this information without recourse to unpacking is relevant.
Unique identifier for user of packer	Presence of watermark or licence key information	VMProtect, Themida	Tends to exist only for some commercial packers. Access to this information without recourse to unpacking is relevant.
Unprofessional coding standards	Buggy or poorly written code		Would tend to be exhibited only by non-commercial packers.

Table 1: Packer properties contd.

Reviewing the status of packers

It is possible to establish a set of statutes on the exhibition of properties in a packer and the actions for infringement, after taking into account any mitigating circumstances.

The case for determining the status of and action on any given packer can depend on:

- Properties of the packer: how well does the packer behave vis-à-vis adherence to the statutes?
- Willingness of the packer vendor to be a good citizen in terms of cooperation with the security industry to protect the public at large: this could entail information exchange, and accommodating modifications in the design and implementation of the packer properties.
- Prevalence of the packer in clean and malware files: what are the practical implications of exercising different actions on a packer?

The following caveats ought to be considered:

- The exact nature of specific actions based on statute infringement is the prerogative of each individual member of the security industry.
- It is possible to provide and distribute a list of security industry-endorsed, well-behaved packers to be utilized by responsible users amongst the general public.

SUSPICIOUS PROPERTIES OF PACKERS

The exhibition by a packer of certain packer properties in an individual capacity may be sufficient to raise the status of the packer to a suspicious threshold. However, in other cases, the exhibition of certain combinations of packer properties by the same packer may warrant a heightened level of suspicion and consequent action.

Table 2 shows individual packer properties or combinations of them which may be deemed to be suspicious together with a

Suspicious packer properties	Justification	Caveats
Polymorphic code at the entrypoint	Implies explicit intent to evade identification, especially if multiple invocations of the same packing utility on the same target, with the same options yield different entrypoint code.	Nops (90) are sometimes legitimately used as placeholders for future patch code.
Destruction of target data	The destruction of important target identification data is unacceptable.	Digital signatures for the target cannot be maintained post packing. However a warning ought to be explicitly displayed when a digital signature is going to be effaced.
Impersonation of another packer	Implies an explicit intent to evade identification. Could cast aspersions on well-behaved packers.	
Non-standard use of APIs	Implies explicit intent to trick emulators and sandboxes.	There may be genuine errors in API-calling code.
Pure compressor using any form of obfuscation	The main function of a self-professed pure compressor is to make the target smaller. The use of any packer code which leads to a larger overall packed object than it could have been raises suspicion.	
Exhibition of pre-emptively prohibited characteristics*	Use of undesirable disruptive techniques which have not been previously used in packers and have been explicitly banned a priori.	
Unprofessional coding standards in unidentified packer	Implies that the packer is non-commercial and unknown which may indicate a custom packer. Custom packers are regularly used to obfuscate malware.	Ascertainment of what constitutes 'unprofessional coding' is subjective but is likely to depend on professional experience and judgement.
Multiple layers of packing	Implies an intent to evade identification via obfuscation.	A paranoid user could have resorted to multiple layers of packing. Does not necessarily call into question any individual specific packer software envelope.

* A list of pre-emptively prohibited characteristics can be provided on request.

Table 2: Suspicious packer properties.

Suspicious packing application properties	Justification	Caveats
Lack of wide public availability	Difficulty in tracing lineage of packer. Implies that the packing utility may be custom (custom packers are often used to obfuscate malware).	
Malware-related nature of resource, e.g. website, hosting packing application	Implies an intent to be used for untoward purposes.	Well-known packers may also be hosted on these sites.
Malware-related nature of user interface for packing application	Implies an intent to be used for untoward purposes.	The perception of the look and feel of a user interface is likely to be subjective.

Table 3: Suspicious packing application properties.

corresponding justification, and relevant caveats or mitigating circumstances to consider. Several entries could have exceptions and are potentially subjective. Ultimately it is the context of use for packer properties and its perceived intent based on professional experience which is likely to influence the final judgement.

The contents of Table 2 are by no means exhaustive or immutable. They are likely to evolve based on a variety of factors including the nature of current packed threats and negotiations with packer vendors.

SUSPICIOUS PROPERTIES OF PACKING APPLICATIONS

The status of a packer can be influenced strongly by the perceived nature of the packing application used to create it. A suspicious packing application is likely to cast aspersions on the packed target it creates, i.e. suspicion by association.

Table 3 shows properties of packing applications which may be deemed to be suspicious with a corresponding justification, and relevant caveats or mitigating circumstances that one might consider. As in the case of Table 2, the contents of Table 3 are subjective and are likely to evolve over time based on a variety of factors.

STATUS OF NON-COMMERCIAL VERSUS COMMERCIAL PACKERS

The fundamental principles on which the status of a packer would be determined remain the same regardless of whether the packer is commercial or not. However, there can be little doubt that the volume of known malware using non-commercial packers is significantly greater than of those using commercial packers.

There is a plethora of non-commercial packers, some of which are open source, e.g. UPX, RLPack, etc. Over the years these packers have been widely abused by malware for two obvious reasons, viz. they are effectively free to use and they are widely available. In addition, many aspects of certain non-commercial packers may be in breach of the statutes based on packer properties described earlier. Therefore it is possible that several non-commercial packers would appear to be candidates for blacklisting.

However, the specific status of, and response to, any particular packer is likely to be determined on a case-by-case basis. The fact that many non-commercial packers have a large user-base (e.g. UPX), and that many security products have the ability to peer through the packer software envelope of several non-commercial packers are valid considerations which would influence the decision-making process.

As previously mentioned it may also be possible to use a standard taggant-based authentication mechanism for both non-commercial and commercial packers. An effective taggant solution could render black or grey lists based on packer properties less relevant for all packers.

One important subset of non-commercial packers is composed of the custom packers which are exclusive to malware. These are likely to be blacklisted without controversy if they can be clearly identified as exclusively malware-related. A robust database infrastructure for known packers will help to ostracize custom malware packers.

SUMMARY OF SALIENT CONTENT

The heavy abuse of packers by malware authors continues to pose a challenge to the security industry, and thus represents a significant threat to millions of computer users globally. Security vendors are duty bound to protect their customers from threats, and have strategies and policies in place to deal with packed files. However, the security vendor response to packer abuse issues is likely to impinge on the commercial interests of packer vendors. Therefore security vendors, and hence the general public, and packer vendors are all affected by the packed malware problem.

It is within the remit of the Industry Connections Security Group (IEEE ICSG) to achieve wide agreement amongst security vendors on the future approach towards dealing with the packer issue, including compiling standards and policies for packers, and establishing a framework for cooperation and negotiation with packer vendors for mutual benefit.

In terms of standards and policies for packers, this document has described the use of digital signatures, watermarks, taggants and various aspects of packer properties to authenticate, block or apply security policies for packed files. It is important to reiterate that the details specified with regard to standards and

policies are likely to evolve based on negotiation with packer vendors and the nature of the malware threat, amongst other potential considerations.

The IEEE ICSG would like to make a solemn overture towards the commercial packing application industry and other legitimate packer creators in the fight against malware. It is highly likely that close and fruitful cooperation between the security industry and packer vendors would result in a more secure computing environment for all, which is certainly in the best interests of security vendors, packer vendors and the public at large.

ACKNOWLEDGEMENTS

- This document borrows from certain discussions held within the IEEE ICSG Malware Working Group. The input of Mark Kennedy (*Symantec*) and Mario Vuksan (*ReversingLabs*) is particularly appreciated.
- The allegory summarizing packer issues vis-à-vis watermarking and taggants is courtesy of Glyn Kennington (*Sophos*). Glyn also contributed to the information shown in Table 1.