deepinstinct™
BEFORE YOU KNOW IT

ENDPOINT SECURITY EVALUATION

June 2017

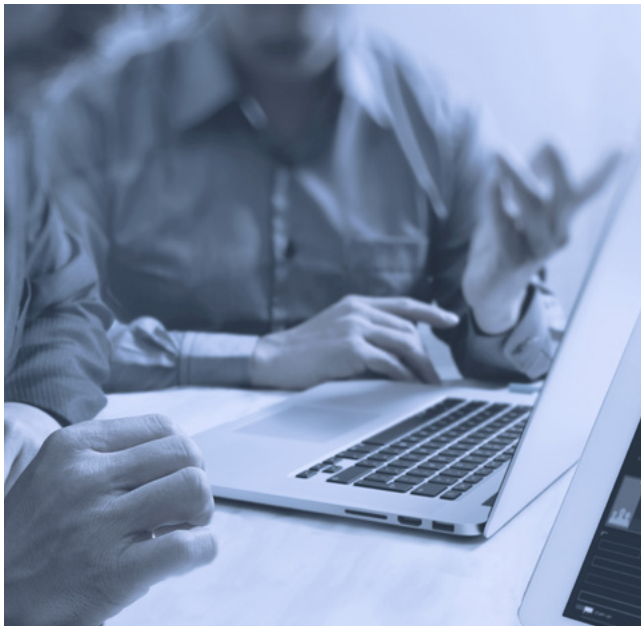
INTRODUCTION

Evaluation, in the endpoint security domain, has been undergoing rapid changes these past few years. In endpoint security solutions, evaluation has two main differentiators: detection rate and false positives. In most of the cases, the measurement of the detection rate test relies mostly on known malware that can be easily retrieved from public or private repositories.

Endpoint security solutions are progressively improving the accuracy of their detection rates, mainly on known malware, based on a wide range of capabilities: blacklists of hashes, signatures, heuristics, machine learning-based models and nowadays, also with deep learning-based models that scan files (statically) or look at the behavior of the processes or the machine (dynamically).

Having those capabilities in place raises, of course, the option that endpoint security solutions will perform well on such datasets from the wild; it is enough to sign them all and consequently, reach 100% detection.

Two points are important: the first is that the new malware, which can be signed easily and rapidly, was actually unknown prior to the publication. Therefore, if an endpoint security solution fails on this malware, it fails both on known and unknown. Surprisingly, during tests, we see that some of the most common solutions are failing easily on new threats.



The second point is that next-generation endpoint security solutions are targeted to detect unknown malware, which other non-next-generation solutions, might fail detection. Therefore, the test case should be different when testing next-generation solutions by using real unknown malware where there is a smaller chance for them to be signed by any means.

Let's talk a bit about the real challenge here: unknown malware and how the solutions act when they first face it.

UNKNOWN MALWARE

The number of unknown malware is constantly increasing. It is more common that new malware families are created (based on open-source malware or on leaked source codes – intentionally and unintentionally), new versions of current malware families are released (with new features or new sophisticated evasion techniques in parallel with the improvement of the detection capabilities) or just new mutations of known malware (that have already been signed). Therefore, these new variants, which are generated easily and rapidly, bypass current existing signatures.

There are a few different ways for creating mutations:

CHANGING THE HASH

A small change in the file itself, even by appending a byte, will change the hash of the file. Endpoint security solutions that rely on hash blacklisting (cloud reputation services in most of the cases) are vulnerable to such "mutations" because their existing hashing signatures will not match those new mutations' hashes.

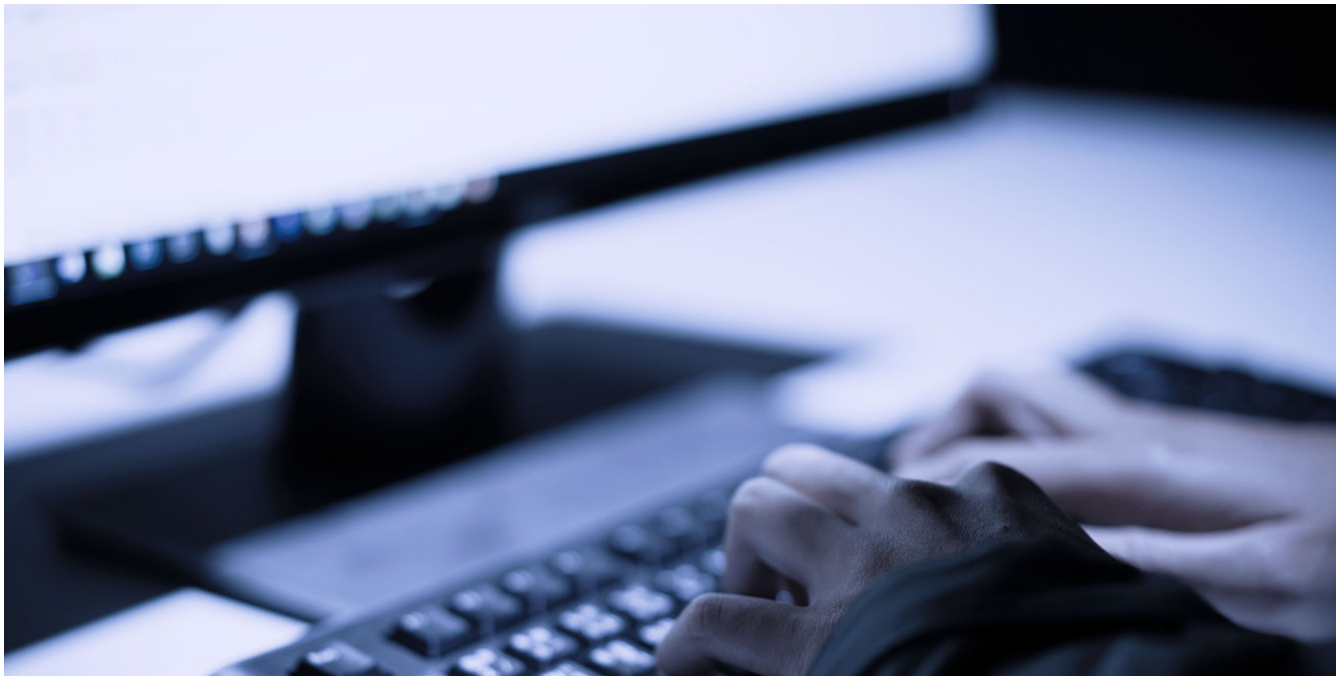
PACKING

Binary files can be packed with a packer (also known as "compressor", "crypter", "protector", or even "SFX" – selfextractors) that basically provide a generic layer on the original file, a "mask", so while running it, its stub will start the unpacking process that will be revealed and run later in the original code.

* For more details, please see note 3 in Bibliography

The unpacking can be achieved by several ways: starting from dropping the unpacked files to the disk, mapping and loading the entire original file in the memory, mapping only sections, or even doing that in more than a one shot by unpacking more and more areas during the execution of the binary. The original code/file can be either compressed, encoded or encrypted.

There are many types of packers, and in most of the cases they are used to protect the reading of the original source code statically; to compress the size of the binary, or to protect it from piracy. Those needs are usually required by legitimate software as well, and consequently, many legitimate software use packers too. UPX, for example, is one of the most common packers in the wild. If we examine the distribution of UPX-packed files, it seems that there is a ratio of 1:2.5 for benign:malicious files. Although there are a few security vendors that define packed files with specific packers as malicious, this kind of heuristic creates false positives on those legitimate packed files. Therefore, the ideal is to be able to determine whether the file is malicious or not without such robust heuristics.



Even though some packers might create a new variant each time the original file is packed, some of them will provide the same, new variant. Therefore, to achieve the purpose of generating a big number of mutations automatically, packing the original file won't necessarily help, as for each packing iteration the same packed file will be generated. Attackers usually pack the files not as an automatic mutations creation mechanism, but to provide another evasion layer.

In addition, the output, packed file from many packers, is reversible. This means it is possible to unpack it easily without executing the packed file. Security vendors usually do this to scan the "clean", unpacked version of the file statically. Having said that, unique and "zero-day" packers exist. They are called "FUD" – fully undetectable – where the packing technique is still unknown and has not yet been reversed.

NEW VARIANTS – MODIFICATIONS OF THE MALWARE BINARY

New variants are usually created by modifications of the original malware binary itself. This is done on the features that security vendors might sign, starting from hardcoded strings, IP/domain names of C&C servers, registry keys, file paths, metadata or even mutexes, certificates, offsets, as well as file extensions that are correlated to the encrypted files by ransomware. It can also be on the code itself, with techniques such as polymorphism, in which the opcodes are changed into other ones while keeping the original functionality; or metamorphism, in which useless parts of code are added to confuseandchangetheorderofthestructures.

NEW MALWARE FAMILIES

Apart from the abovementioned methods that attackers might use to create such mutations and other variants of the same malware, new versions of existing malware or new malware families can be generated for the same purpose of evasion. A new version of an existing malware can be defined with new features that the malware provides, to make its business logic different. Another way can be by applying new attack vectors or evasion techniques to bypass the current signatures of the endpoint security solutions. Additionally, a new malware family can be written from scratch, or be based on a source code of another malware. For example, HiddenTear or EDA2 are opensource ransomware on which many new ransomware families are based.

WHAT IS THE RECOMMENDED METHOD TO EVALUATE ENDPOINT SECURITY SOLUTIONS?

At first glance, it is important to verify that the solution does not rely only on hash blacklisting, because it is easy to bypass in the real world. To change the hash, appending one byte is enough by using the following Linux command:

```
truncate -s +<amount of bytes> <file_to_mutate>
```

It is also important to verify that the solution can detect new malware. Even though such samples are already known, until recently they have not been discovered. As such, if it takes time for a solution to detect such a new family, it is not good.

For this purpose, there are plenty of public repositories with malware. However, bear in mind that those repositories usually contain a lot of non-malware files, whether benign or potentially unwanted applications (which might be considered with a lower priority in terms of the evaluation). Therefore, you cannot rely by default that their classification of files as malware is correct.

Alternatively, you can bundle such a repository by yourself, by looking for samples from publications over the Internet, or from threat intelligence feeds such as AlientVault OTX. You also need to keep in mind that it is important to make it as varied as possible – bundle as many samples from as many different families as possible, so that the results are representative.

In addition, the main goal is to test new, unseen malware. Creating mutations based on available source code of malware is a great option to evaluate endpoint security solutions.

DEEP INSTINCT AND DETECTION OF UNKNOWN MALWARE

Deep Instinct provides unmatched detection and prevention of any type of malware, using deep learning to leverage its detection capabilities. Since we do not use any type of signatures, Deep Instinct is immune to hash modifications. We also successfully classify packed files – whether using simple and known ones or even FUDs.

During our training phase, we add “noise”, which changes the raw data from the files we feed into the algorithm, in order to automatically generate slight “mutations”, which are fed in each training cycle during our training phase. This concept immunes Deep Instinct against the modifications that are applied to the different variants, such as strings or even polymorphism.

Regarding new malware, those are usually developed based on other malware source code, or at least based on some malicious piece of code, providing the ability to detect them as well.

Contact us for assistance with bundling the evaluation test case, with the methods described above, on both known and unseen, new malware using our propriety mutation tool.

ABOUT DEEP INSTINCT

Deep Instinct is an omni-cybersecurity platform that helps companies and organizations protect themselves against zero-day, APT and ransomware attacks with unmatched accuracy. By providing deep learning predictive capabilities, and a solution that is based on a proprietary deep learning framework Deep Instinct is revolutionizing cybersecurity. Deep Instinct's solution provides comprehensive defense designed to protect against known and unknown malware in real-time, across endpoints, servers, and mobile devices. Deep learning's capabilities of identifying malware from any data source results in comprehensive protection on any device and operating system.

To learn more about Deep Instinct capabilities, get a personal demo from one of our experts

GET A DEMO

deepinstinct
BEFORE YOU KNOW IT

www.deepinstinct.com