TRUST IN GERMAN SICHERHEIT

BEAST

Overcoming limits of traditional malware detection

G DATA CyberDefense AG | September 2019

Detecting sophisticated real-world malware has become a challenging task, when only relying on traditional signature-based detections.

Limit n° 1: retroactive detection

First and foremost, signature-based detection is by definition a reactive method. Regardless of the level of automation involved: Only after a file is categorized as malicious, a signature can be written for this file or a group of related files (typically a malware family).

Today, malware authors break this reactive approach by vastly increasing the frequency in which they release malware that is perceived as new.

Limit n° 2: uniqueness of malware

Cyber-crime has become a billion-dollar industry. Just as traditional businesses, cyber criminals boost their efficiency by using automation tools. Some malware authors use automation services to actively monitor if anti-malware solutions detect their malware. Once a detection is registered, they immediately remove the identified pattern from their samples or alter their programs automatically to avoid detection.

They heavily apply encryption and packers to their samples to obfuscate their malicious core. Some malware authors roll out a new malware sample every few minutes. Others even generate an individual sample for every single victim (“server-side polymorphy”).

Obviously, when malicious files are so unique, it is difficult to find malicious patterns that can be used to generate a traditional signature, which is able to identify a whole malware family. Even if there are patterns that can be used to build a signature, this effort can be practically worthless.

Limit n° 3: evasion from backend analysis

Another problem is that the identification of malicious files often relies on the analysis backend of the security vendors. Every potentially malicious sample the vendors get their hands on are thoroughly analyzed, for example in sandbox systems that execute the samples and try to identify any malicious actions. Only if the samples are identified as malicious, a process starts to find malicious patterns. Unfortunately, often malware samples are designed to be aware of their environment, and will detect if they are being analyzed. They might only expose their malicious

behavior at a certain time, on a certain location, after running a certain amount of time, or after recognizing user interaction that is normally not present in a sandbox system.

Because of these problems, besides traditional detection methods we also need to be able to detect malicious behavior where it happens: on the affected system.

The solution: behavior analysis

Security vendors, therefore, have implemented detection technologies that analyze the behavior of processes on a system to determine if it is malicious or benign. In order to be resourceful with the hardware such an analysis focuses on especially suspicious system parts, such as the file system, the registry or the autostart-folder. This enables security vendors to detect entirely unknown malware families.

Most of those solutions try to translate threatening behavior into values to determine a degree of “badness”. Mathematically, it is not possible to avoid a loss in accuracy when many of those values are being aggregated into a total score. Even with machine learning there is still a level of haziness involved in this method that leads to a certain level of misjudgment when categorizing a process as either malicious or benign.

Most consumers will not be affected by this. Businesses however, often use highly specialized software tools and use processes in an intended, unharmful way that is unusual in most other environments. If the threshold of a behavioral analysis tool is set too high it will block those processes, if it is too low malware might not be detected. In the real world, security vendors tend to avoid detection errors by asking users to permit processes to happen. When this occurs too often, users will either turn of the technology or ignore warnings. Either way, the risk of infections will increase.

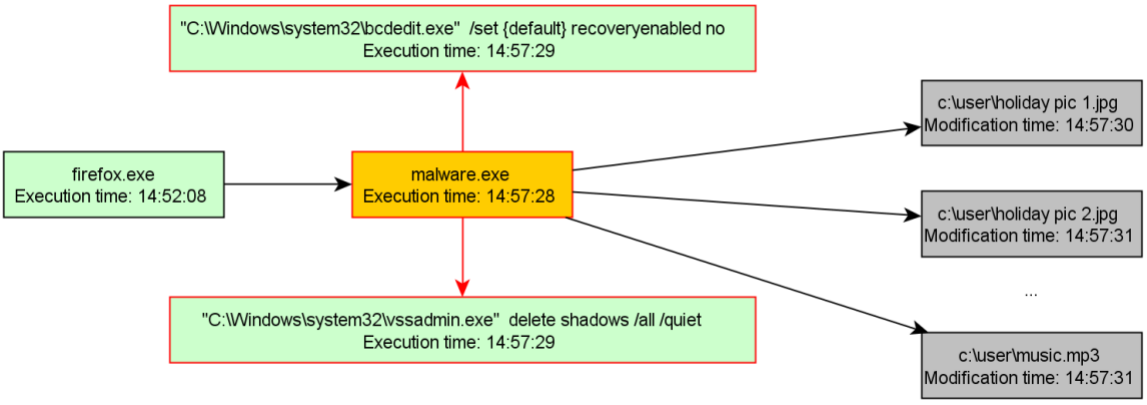
The real solution: BEAST

BEAST is a behavior-based detection technology, developed by G DATA that monitors behavior and stores every observed action to a local light-weight graph data base. BEAST does not rely on the identification of the malware itself but on the observation of generic malicious behavior. This is especially helpful against rare malware and malware families.

How BEAST works: Graph-based rule matching

On a protected system, BEAST monitors the behavior and stores every action. Actions include access to the file system, to the registry, network connections and inter-process communication. Whenever something is added to the graph database, the graph is searched for malicious behavior patterns.

The following graph can be used to illustrate this kind of rule-based matching:



This exemplary user likely has been tricked by a website to download and execute the malicious file “malware.exe” from the internet in the Firefox browser. Actually, in this case it is an infection with ransomware. Ransomware encrypts the user’s files and afterwards asks the user to pay a certain amount of money for the decryption of the files.

The malicious process here immediately starts an instance of the “bcdedit” system tool to disable Windows’ startup repair function. Afterwards it starts an instance of the “vssadmin” system tool to delete so-called shadow copies, which can be used to restore accidently overwritten file. Then it starts to encrypt several files in the “C:\user” directory.

As the start of the two aforementioned system tools is a typical preparation of ransomware before encrypting files, with the idea of preventing the user to restore his system, the behavior (highlighted in red) can be considered as clearly malicious. Therefore, the process “malware.exe” would be stopped and the binary file would be moved to quarantine. As the binaries “vssadmin.exe” and “bcdedit.exe” are benign system tools only abused by the ransomware, they would be kept on the system.

New Opportunities: Retrospective Removal

G DATA, with automated backend systems or manual analysis processes, identifies lots of Indicators of Compromise (IOC) every day. An IOC could be a Command&Control-Server (C&C) used to operate a botnet, or a particular file that has been identified to be malicious.

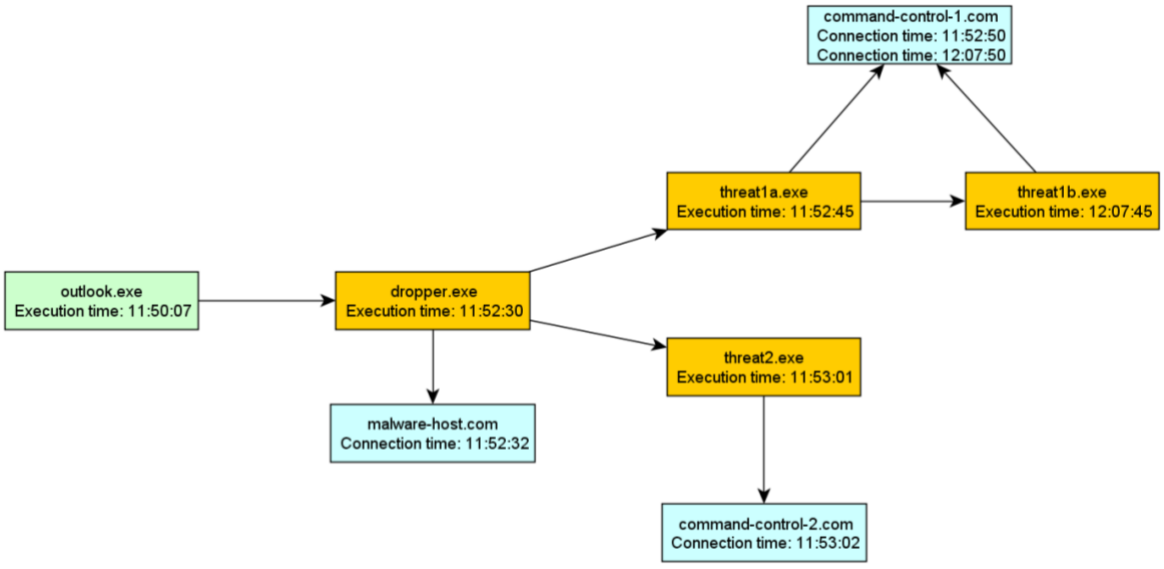
In conventional endpoint security software, actions are only compared to IOC lists at the very time the action takes place. For example, before a file is executed, it is compared to a list of known malicious files. Or if a process connects to a host, the host is checked against a list of known C&Cs. If the host is identified as malicious, the whole process is detected as malicious.

The core problem is though that the security vendor's IOC identification processes are again reactive – they begin after the vendors start analyzing the threat, which obviously can only happen sometime after a threat actually emerges. Even, if this is automated and executed in a very short amount of time, the time gap is still significant in the context of malware detection. To put it simple: Security vendors are often too late and often only can be too late if they rely on traditional methods.

In BEAST the actions (behavior) are stored in a local graph database. Therefore, everything in this database can be compared to IOCs identified by G DATA even after they take place. And as the graph database also contains all actions related to the IOC, all these actions can be rolled back, effectively allowing for a retrospective malware removal.

This is particularly important if a system has been compromised, but no malicious actions have been triggered yet that could have triggered the generic behavior detection.

To illustrate this, imagine the following simplified behavior graph:



Initially, in the mail program Outlook an infected attachment is opened. Therefore, a file called “dropper.exe” is created and executed by the process “outlook.exe”. The new process then connects to “malware-host.com” to download and execute further malicious binaries (“threat1a.exe”, “threat2.exe”). Both connect to their respective C&C servers (“command-control-1.com”, “command-control-2.com”). After around 15 minutes, “threat1a.exe” receives a command to update the binary to “threat1b.exe”, which subsequently also establishes a connection to the same C&C server.

If, for instance, G DATA identifies the server “command-control-2.com” or the binary “dropper.exe” as an IOC, BEAST can – even hours after the infection – just walk through the graph, find and remove every single binary related to the infection.
Two side notes: First, also any change to the Windows registry and therefore the system configuration is recorded and can be rolled back (omitted here to keep the graph comprehensible). Second, outlook.exe as a known benign executable wouldn’t be removed.

What’s the difference to the existing Behavior Blocker?

The existing Behavior Blocker basically receives a stream of every action, conducted by a process. It assigns a particular numerical “badness” value to every single action. It then summarizes all the badness values, and when a certain threshold of badness is exceeded, a process is considered to be malicious.
Principally, the Behavior Blocker therefore has a process centric view, while BEAST has an overview of the whole system. Also, because the Behavior Blocker only aggregates numerical badness of actions, it’s not possible to detect particular combinations of actions as malicious. This makes it difficult to specifically detect malicious behavior patterns. Whenever an action is assigned a new or higher badness value in the Behavior Blocker, this might provoke false positive detections. This made it troublesome in the past to quickly react on new threats. And even if carefully considered, whenever an action was assigned a new or higher badness value in the past to detect a new threat, there was a huge risk of provoking false positive detections. For BEAST on the other hand, the detections are based on very specific combinations of malicious actions. Therefore, it’s easier to add new rules, while being generally way less prone to false positives.

Also, the possibility of a retrospective comparison of IOCs and a retrospective removal is only possible with BEAST.

Why do we need BEAST when we have DeepRay?

DeepRay's strength is that it's able to look through packers, enabling us to identify malicious malware cores. The initial identification of malware cores is a manual process though. For the most prevalent crimeware families, this is a manageable task. Still, even for these, BEAST will help to fill the time gap until a DeepRay detection, in the case the malware is changed at its core. But besides the most prevalent crimeware families, there's also a big heap of malware families that are not very prevalent as a single family, but summarized these families also make up for a huge amount of infections. And if the reason for being not so prevalent is that these families are used in targeted attacks, they are even particularly dangerous. As BEAST doesn't rely on the identification of a particular malware core, but on the generic observation of malicious behavior, BEAST will help mitigating these threats and therefore add another layer of security.