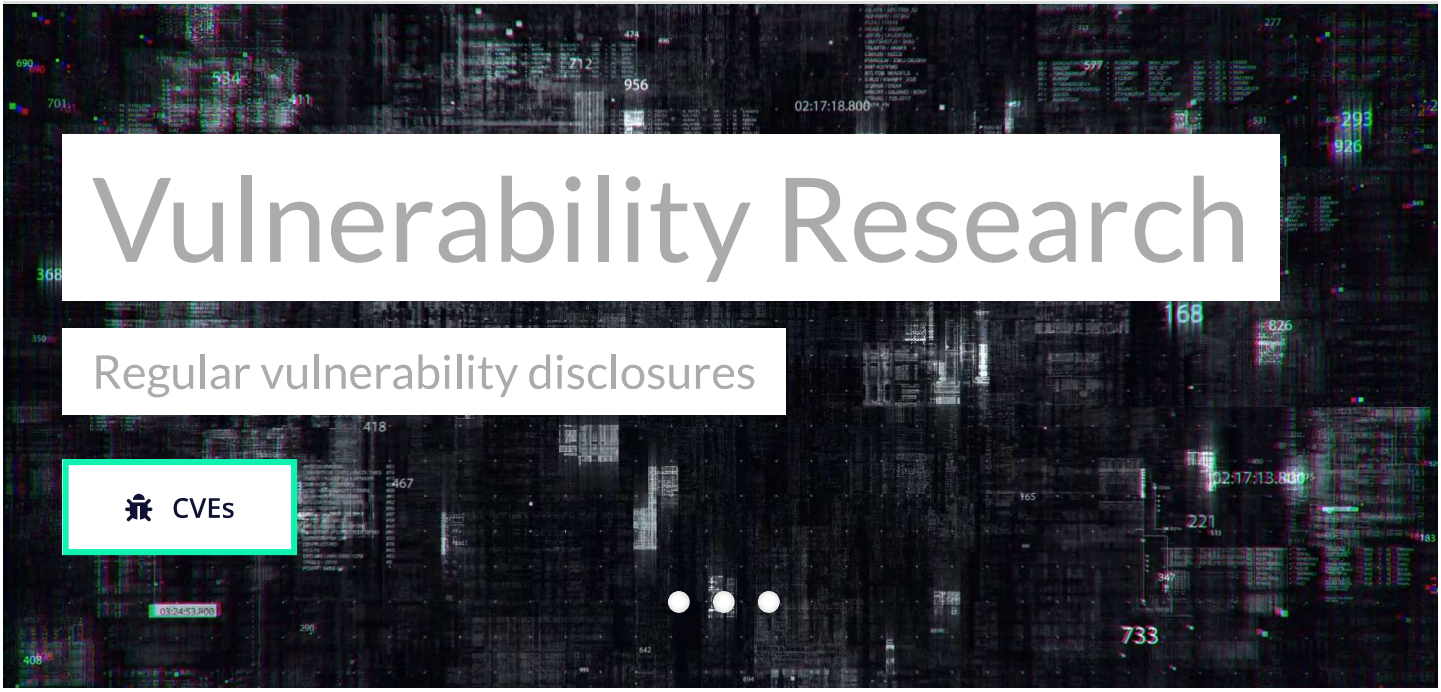


Escaping the Avast sandbox

By Kyriakos Economou | April 19, 2016

An Avast Sandbox escape, CVE-2016-4025, is possible due to a design flaw in the Avast DeepScreen feature. It is likely that this flaw will remain in supported Avast products for some time.

Breaking static AV detection signatures is quite trivial. The AV industry has started to understand that they cannot rely on this anymore nor on simple heuristics on known behavioural patterns, for example based on a certain logic of execution paths and function calls.
The next big thing in malware detection, from the AV point of view, is sandboxing an unknown sample and analysing it inside a fully controlled environment while monitoring its behaviour in a more

generic way.

In addition, providing extra sandboxing capabilities that allow the user to execute untrusted applications in a safer way, and/or mitigate in common scenarios the impact of an exploit against a trusted one, such as a web browser, is something that can be very valuable. There is still a lot of work to do in this area, but this is the future for preventing 0-day malware infections.

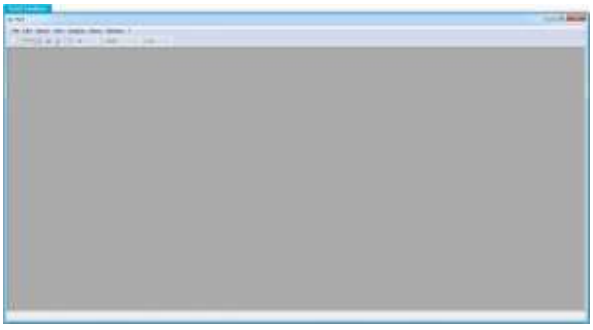
In this article we will focus on a design flaw in the AVAST Sandbox/DeepScreen features, and the impact that this can have over the extra security layers that these features attempt to provide.

As a side note, after doing some research regarding this flaw, I discovered a few videos online named as “AVAST Sandbox Bypass”, which are not related with escaping from a fully sandboxed process. These videos just demonstrate that AVAST products do not always trigger the DeepScreen scan feature, and not how to escape from the sandbox while the process is already running inside it.

The Avast Sandbox

Avast is one of the first AV vendors that incorporated their own Sandbox in an AV product. This is not to be confused with other sandboxing techniques implemented at a userland level, for example by web browsers such as IE. These are usually implemented by lowering the integrity level of a process and/or by removing certain resource access related privileges from it.

The Avast Sandbox implementation is based on a kernel mode file system driver called “Avast Virtualization Driver” (aswSnx.sys) which is responsible for isolating a specific process from the rest of the system. In other words, it blocks a sandboxed process from interacting (code/remote thread injection) with other processes that run outside of the sandbox, as well as dropping new files and/or modifying existing ones.



Popular Recent

Windows Inline Function Hooking
March 18, 2015

The Problem of Data Loss Intelligence
July 9, 2015


Explo Netw Secu Came Unde and Mitig the Risks
February 15, 2023

Figure 1: Sandboxed Application

Avast's Position

"The Avast Sandbox lets you run a questionable program without risking your computer."

"The Avast Sandbox is a special security feature which allows you to run potentially suspicious applications automatically in a completely isolated environment."

"…programs running within the sandbox have limited access to your files and system, so there is no risk to your computer or any of your other files."

"The advantage of running an application in the Sandbox is that it allows you to check suspicious applications while remaining completely protected against any malicious actions that an infected application might try to perform"

Avast 'DeepScreen' Scan

This is actually a very nice feature of Avast AV products which attempts to raise the bar in terms of behavioural analysis of unknown executables. It takes advantage of the built-in sandbox to monitor the behaviour of an executable for 15 seconds the first time it runs. During that time period the process runs inside the sandbox and if nothing suspicious is being detected by then end of it, the Avast will automatically restart the process which in the future will always run out of the sandbox.

That is of course, unless the user intentionally instructs the Avast software to run a specific executable in the sandbox next time it is executed or always.

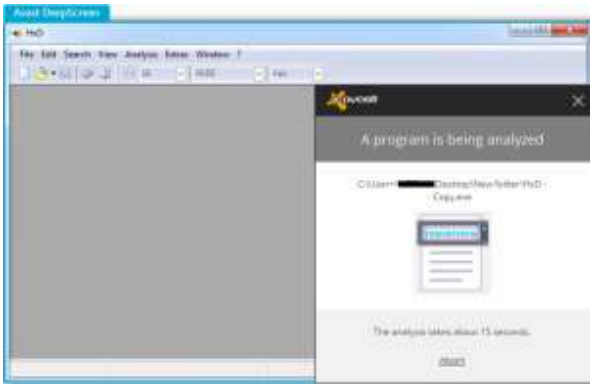


Figure 2: DeepScreen Scan

Avast's Position

"When a file is "DeepScreened," it is actually run in the Sandbox, which is mainly responsible for keeping things isolated while watching for various high-level events and behaviour of the program running."

Playing with Avast Sandbox…

We started examining the security of the sandbox by trying to find a way to drop new files or modify existing ones through a sandboxed process, as well as try to execute code in the address space of another. At first, we tried the most obvious actions which were only involving direct calls to Windows functions such as CreateFile and OpenProcess. These attempts were successfully blocked by the sandbox, so we had to keep looking.

At this stage we started looking at the flexibility related capabilities of the sandbox, or in other words what happens if a user wants to save a file generated by a sandboxed process.

We started a sandboxed instance of notepad.exe, typed a few letters and then used the 'Save As' dialogue box to permanently save the text file on the hard drive. This attempt was successful, but then we were also going through a quite legitimate route. So, just to ensure that there isn't some sort of white-listing we tried to directly save a file from a sandboxed instance of notepad. We manually injected some code that was invoking a direct call to *CreateFile* in the address space of that process which was successfully blocked as well.

At this stage, there were a couple of questions that we had to answer.

1. How does the 'Save as' functionality succeed in saving a file out of the sandbox?
2. How secure is this?

Finding our way out…

From the previous observations, we made a fair assumption that there must be an interaction with the kernel mode driver and from the title of this article you know where this is going.

Indeed, by using 'API Monitor' in order to look for calls to DeviceIoControl and CreateFile functions we discovered something very interesting happening when the 'Save As' dialogue was used.

Figure 3: API Monitor – Calling DeviceIoControl

As you can see in the figure above, there is a combination of calls to *DeviceIoControl* and *CreateFile* functions. The interesting call that involves the interaction with the virtualization driver occurs through a proprietary module of Avast (*snxhk.dll*) which is loaded by the sandboxed notepad process.
Let's have a more detailed look at this:

- *DeviceIoControl*
  - *Calling Module: snxhk.dll*
  - *Device: .aswsnx*
  - *Driver: aswSnx.sys [avast! Virtualization Driver]*
  - *IOCTL: 0x82AC0168*
  - InputBuffer_Size:0x80C
  - OutputBuffer: Not Required

Figure 4: Calling DeviceIoControl

At this point, it is important to mention that Figures 3 and 4 were created while analysing version 10.x of Avast Windows home user products. Later, in versions 11.x we noticed that there is an attempt to hide the existence of the *snxhk.dll* module which doesn't appear any more in the list of loaded modules. This causes *'API Monitor'* to miss those calls and gives some food for thought for other things as well. This, however, is not really a problem since it doesn't change the fact that the call to *DeviceIoControl* needs to be performed which then makes it trivial to find the module in the address space of the

examined process as shown in following figure.

Figure 5: Locating snxhk.dll module in versions 11.x

That's all we need to know in order to escape from a fully sandboxed process either this has been temporarily or permanently sandboxed by the user or during the 15 seconds time interval that the *'DeepScreen'* scan offers.

Indeed, there is no further validation by the virtualization driver nor is the user going to be asked for his own permission to release a file out of the sandbox. Furthermore, using the aforementioned IOCTL we can also freely modify or let's say infect or encrypt, in the case of a ransomware attack, any existing file that the user has read/write access permissions to this. All of that from the comfort and security confidence that the Avast sandbox claims to provide.

Keep in mind that for different versions of these products small modifications might be required in the parameters passed to the *DeviceIoControl* function, but these are trivial to figure out.

Security vs Usability: 0 – 1

We reported this issue to Avast at the beginning of November 2015 as soon as we discovered that it basically renders their most important security feature of all their Windows products basically useless.

This is clearly a very serious flaw, but according to them *"it's quite hard to find the balance between security and usability"* and for that reason it didn't qualify for a reward under their bug bounty scheme.

This vulnerability is still present as more than 4 months later their attempts to remediate this were insufficient . Recently, Avast claimed that they mitigated that issue during the *'DeepScreen'* scan. However, Avast's solution was not to allow the user to click on the *'Save As'* dialogue of an application during those 15 seconds, but this doesn't stop the application from using the *DeviceIoControl* function at will, just as we demonstrated. In other words, the attack surface remains the

same.

Products Affected

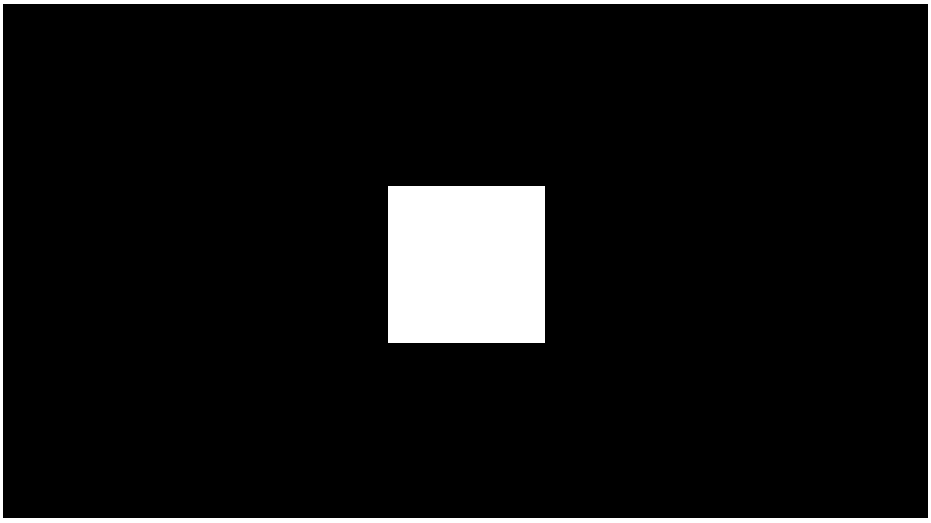
- Home User Products
  - Avast Internet Security v11.x.x
  - Avast Pro Antivirus v11.x.x
  - Avast Premier v11.x.x
  - Avast Free Antivirus v11.x.x
- Avast For Business
  - Avast Business Security v11.x.x
  - Avast Endpoint Protection v8.x.x
  - Avast Endpoint Protection Plus v8.x.x
  - Avast Endpoint Protection Suite v8.x.x
  - Avast Endpoint Protection Suite Plus v8.x.x
  - Avast File Server Security v8.x.x
  - Avast Email Server Security v8.x.x

Earlier and the latest versions of these products are currently affected.

Conclusion

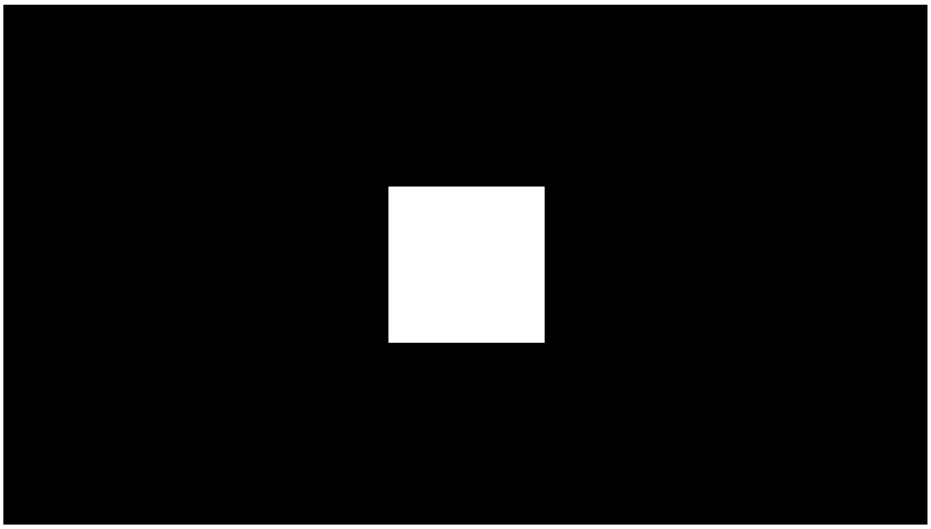
After exposing this issue, we hope Avast will work seriously on it and mitigate the impact that this might have. There are also other interesting IOCTLs that we encourage you to play with, but we will leave those as an exercise to the reader.

Avast Sandbox – Escape during DeepScreen scan



00:00 01:35

Avast Sandbox – Escape during permanent sandboxing



00:00 01:56

References

AVAST Sandbox – https://blog.avast.com/2015/09/09/what-does-the-avast-sandbox-do/

AVAST DeepScreen – https://blog.avast.com/topic/DeepScreen