Project proposals

Memoryscanner

Description:

Research a method to reliably identify suspicious memory regions of processes running on the Windows OS to perform a scan on them. Two possible scenarios should be considered:

Scan on-access: Memory allocations shall monitored and if that memory gets executed we want to scan it before it gets executed (guard feature). The easiest way would be scanning memory regions which have a thread running code from that region.

Scan on-demand: this means identifying interesting regions and scan them (repair feature). Since there is no indication where to start with the scan an option would be, to traverse through the PTE entries or the VAD-tree and extract the information in there (which information?) to determine if this region is worth scanning. This could be for example the memory rights of the region - we just want to scan regions marked as executable or regions which don't have a file object or unknown/suspicious file object associated. Also of interest could be if we can identify if a certain address is mapped to different processes which would make it also very suspicious. Those identified memory regions then should be scanned with a classic signature engine or binsider.

Scope:

The goal is to create a POC which can identify potential suspicious memory regions of processes which then can be passed to a scanning engine (not scope of the project).

Simulated internet for offline sandbox systems

Description:

Sandbox systems play a large role in today's security industry to automatically analyse the ever increasing quantities of samples reaching the users every day. A Sandbox can have different purposes, for example providing insight into the complex malicious behaviour or letting the malware remove it's own cryptographic layers by executing it, to reduce the work the analyst has, as well as providing new methods of detection. A lot of samples actually require an active internet connection to function correctly, for example to download the actual payload, report back, execute some attack or check for a sandboxed environment. Because of different security and privacy concerns, most Sandbox systems don't have an active internet connection.

Scope:

The goal is to investigate the possibility and feasibility of a driver (and probably some kind of user mode component) that allows such a Sandbox to emulate basic network functionality without actually being connected to the internet and create a POC.

Generic approach to deobfuscate obfuscated .NET binaries

Description:

The .NET Framework is an important part of today's malware because of the pre-installed availability in modern Windows systems as well as wide spread publicly available malicious sourcecode examples. One big draw back of the .NET Framework is the possibility to decompile the binaries easily and obtain some sort of "plaintext" sourcecode that makes the analysis of samples very easy. To counter idea theft in commercial applications a large variety of Protectors/Packers has been created, which obfuscate the original binary, making it impossible to draw conclusions from decompilation. Of course now malware authors use the same tools to obfuscate their malicious code, making analysis much harder.

Scope:

The goal is to investigate the different Protectors/Packers and their methods with the aim to create a generic tool that is able to simplify the sourcecode and make it understandable again.

Note: requires a freakish skills

Utilizing AMSI

Description:

File formats other than PE are playing a big role these days when it comes to infecting users. Scripting languages like VBS, VBA, PowerShell, JavaScript, etc. are abused to do the work. To avoid detection of those formats the malware authors encrypt the scripts to make them basically unreadable. Microsoft offers the so called AMSI interface (https://docs.microsoft.com/en-us/windows/desktop/amsi/antimalware-scan-interface-portal) which offers a possibility to scan potentially decrypted artifacts of the obfuscated script during the execution of the script. An Anti-malware scanning engine has then the chance to scan the potentially decrypted script and stop the execution of it in case of a threat is detected.

Scope:

The goal is to create a detection module for the Avira Protection Cloud utilizing the AMSI interface to achieve better detection rates on obfuscated script malware.

Botchecker - infiltrating botnets

Description:

There is project called "*BotChecker"* which infiltrates botnets to harvest the freshest malware samples, download URLs and C&C servers. This is mainly achieved by two steps:

First: Extract the information off the sample which C&C server it is going to connect to - each sample could have different addresses.

Second: The second part is to mimic the botnet protocol for this the researcher needs reverse engineer to get a complete understanding of the custom protocol. Sometimes the protocols contain traps to exactly prohibit such an infiltration with a rebuild protocol and it just doesn't work. Figuring something out like this is quite time consuming.

E.g.: A better approach could be to use a VM, run the sample and create protocol buffers. Then monitor only the main communication loop and the send/receive buffers in the sample. The buffer then needs to be analyzed to extract the data of interest.

Scope:

Overall, this project requires high maintenance. The goal is to find a more pragmatic approach achieving similar results.

Threat Intelligence Platform

Description:
In order to enhance the capabilities of the Avira Threat Intelligence Platform, the threat data needs to be continuously analysed, aggregated and correlated by employing statistical methods, machine learning models and visualization techniques that facilitate pattern and information discovery. Extensive reading and research are needed so that the platform not only keeps up to date with the competition but is at least one step ahead (feed standard formats, emerging threats..). Additionally, most of the analysis is performed automatically, therefore services and applications need to be developed so that the large volume of data and the ingestion speed never becomes an issue. Complex and hidden relationships residing in the data can be revealed and discovered faster by taking advantage of visualization techniques; these can also increase the value of intelligence reports as a product.

Scope:

The goal is to enhance the capabilities of the Avira Threat Intelligence Platform by using combined knowledge of computer science and applications, modeling, statistics, analytics and maths to solve problems

Memory Protection

Description:

A lot of malware is using methods like reflective loading and process hollowing to execute x86/x64 payload in-memory in the scope of harmless/clean processes. These methods are used by so called "fileless" malware or living off the land attacks (using standard tools from Microsoft Windows to achieve the infection).

An older internal project called Exploit Shield was transformed into a module that can block reflective loading and process hollowing, initially only if this is performed by Powershell.
The module is not yet added to the product and released as this needs further testing and research.

Scope:

The goal is to setup test machines and test the memory protection, finding potential false positives, research for other sources than Powershell also if other special targets can be monitored without triggering new false positives. Research needs to be done on the most relevant/widespread malware families to see if they use these injection methods and if the memory protection module can support them.

WMI monitoring

Description:

Windows malware abuses all tools available on the operating system to retrieve information without revealing its presence. Also WMI is abused by malware by sending WQL (WMI query language) queries to get the information of interest.

Scope:

The goal is find out if there is a way to monitor the WQL queries and their corresponding origin (e.g. PowerShell, ...) and block the execution in case of it is identified as malicious.

Restoring the broken execution chain

Description:

Windows malware abuses WMI to break the real execution chain. Whenever WMI is used to spawn a child process the real parent process is no longer obtainable. WMI utilizes the Win32_Process class and the child process is then being created by wmiprvse.exe. In such a case the process tree shows wmiprvse.exe as parent instead of, e.g. PowerShell.

Scope:

Since it is valuable information which process spawns what, the goal is to find a way how it is possible to identify the real parent process.