Türsteher Light

A Path Based Application Whitelisting Filter Driver

Florian Rienhardt

e-mail: peanut@bitnuts.de

July 2014

Abstract

AppLocker's capatilities to whitelist and block executables, libraries and scripts with the comfort of group policies are great but it is pain if you need to use AppLocker as a helping hand to monitor, track and block potential malicious code in forensic scenarios. Having developed several minifilter drivers I was able to build up a light and easy to use filter driver acting like AppLocker helping you to monitor and block executables (exe, dll, sys, ocx...) that were not started from a trusted path. As known from AppLocker, in Tuersteher Light you simply specify a whitelist of trusted paths and fire up the driver. The driver then checks the corresponding path and filename against a list before allowing it to be read into memory for execution. Thus the driver is able to block malicious code started from external USB drives, e-mail attachments, your Internet browser's cache and many more.

Table of contents

If you want to license or buy Tuersteher Light, need consulting on Tuersteher or IT security issues in your commercial environment, contact me by e-mail for assistance and further information.

BEWARE! The driver is for educational, non commercial and test purposes only. Use the driver by and on your own risk. Only try it on a non production environment. I am not responsible nor liable for any damages caused by using the driver.

The target audience for Tuersteher Light are Windows enthusiasts (IT forensic guys and hackers). Meaning persons who know what they do. It is not a tool for the ordinary user!

Table of contents

Are you subject to targeted and special crafted (cyber) attacks?

Your Windows based information technology is attacked to drain your intellectual property?

An Anti-Virus and other protection systems are too generic forcing against these attacks?

Well, you have visited the right place.

During the last years we carefully developed technology against attacks on Windows based operating systems. Benefit from our experience, we are able to

- build up fully customized Microsoft Windows Kernel drivers helping you to identify, block, protect and mitigate against new and unknown threats.
- certify and audit our sources if needed. The main parts of our solutions are less than 1800 lines of code, so it is easy to peek through.

If you are interested in licensing our solutions just send an request via e-mail to

peanut@bitnuts.de

We can build up fully customized demos with respect to your needs, so do not hesitate. We look forward hearing from you.

1 Introduction

AppLocker's capatilities to whitelist and block executables, libraries and scripts with the comfort of group policies are great but it is pain if you need to use AppLocker as a helping hand to monitor, track and block potential malicious code in forensic scenarios. On the other hand AppLocker is only available in Enterprise versions of Microsoft Windows, thus not within reach to the majority of Windows users.

Having developed several minifilter drivers I was able to build up a light and easy to use filter driver acting like AppLocker helping you to monitor and block executables (exe, dll, sys, ocx...) that were not started from a trusted path. Components of this driver are part of our malware detection framework ExploitBuster and Tuersteher, but Tuersteher Light does not contain all the sticky icky features I have build into our heavy weight versions.

As known from AppLocker, in Tuersteher Light you also simply specify a whitelist of trusted paths and fire up the driver. The driver then checks the corresponding path and filename against a list before allowing it to be read into memory for execution. Thus the driver is able to block malicious code started from external USB drives, e-mail attachments, your Internet browser's cache and many more. It is no silver bullet against all attacks with regards to 0-days leading to privilege escalation, in-memory transient malware that only resists in the exploit's allocated memory portion but most exploits we were investigating in 2012/2013 initially stored their malicious executable modules somewhere into the user's folder space or Windows's system paths and hence could effectively being blocked by the driver's approach using a carefully defined white- and blacklist of paths (or files)[1]. There are limits, but as far as I know there does not exist any endpoint security solution out there, that faces all possible attacks[2].

In a typical forensics scenario where you run a test machine against potential toxic web contents I heavily encourage you to only whitelist the folders \Windows\ and \Program Files\. Then fire up DbgView, open a toxic web site (e. g. running an exploit kit) and watch out what my driver blocks and logs.

---

1 For more details on kernel based monitoring and a discussion regarding the limits of drivers to detect and mitigate against exploits and other malware check out http://bitnuts.de/KernelBasedMonitoring.pdf

2 For high security mitigation I suggest to combine Tuersteher Light with MZWriteScanner, an AntiVirus-Solution and EMET (http://www.microsoft.com/en-us/download/details.aspx?id=41138). Although this will not be the silver bullet it is really close to it, especially if you are subject to special crafted targeted attacks that focus on draining your intellectual property and know how.

2 Configurate and start up the driver

First at all define the list of paths or files you want to white- or blacklist, save this list into an unicode file named tuersteher_light.ini and copy it to your Windows folder (in most cases it is c:\Windows\). The file must be an unicode text file to prevent deadlocking your machine. Tuersteher Light supports unicode making it possible to use the driver all over the world, from the USA to Europe, Russia, Africa and also Asia and the Middle East. Thus you are able to define filenames and paths like:

☻♫♪♥♠слу́шаю!דער ... א ָדער אNǐhǎo!مَرْحَبًا

The list of names is case sensitive, beware of that.

You are not allowed to specify paths and files directly by their DOS filename, e.g. c:\Windows\ etc. Instead you must use the path's device and volume descriptor. To make things a bit more clear I included an example tuersteher_light.ini file into the driver's package, so check out this file for more details and on how to specify the names there. I also included a tool that prints out your mount points in the correct format (see MountPointFunctions.exe).

As a first step I recommend to specify all paths or filenames by their path and filename you want to whitelist into the file tuersteher_light.ini. You start the whitelist with the section identifier **whitelist***. After each line of a path or filename in the section of the whitelist you must set an asterisk (*). If you forget the asterisk you might crash or deadlock your system after starting up the driver.

Files for the blacklist are defined in the section **blacklist|**. Files or paths you want to blacklist must be marked with a vertical dash (|). After each line of a blacklisted file or path you must set the dash. If you forget the dash you might crash or deadlock your system after starting up the driver.

If you enable to block the execution of binary modules by defining the **[LETHAL]** tag, Tuersteher Light blocks such modules like in AppLocker. You can also enable a logfile by defining **[FORENSICS_PATH]**, telling Tuersteher Light to log the filenames and/or paths into the file

```
tuersteher_light.log
```

located in your Windows installation path (in most cases c:\Windows). Make sure that you copy an empty Unicode Textfile named tuersteher_light.log into your Windows installation path (in most cases c:\Windows).

Example of a tuersteher_light.ini file:

```
[LETHAL]
[FORENSICS_PATH]
whitelist*
\Device\HarddiskVolume2\Windows\*
\Device\HarddiskVolume2\Program Files\Common Files\*
\Device\HarddiskVolume2\Users\Magnum\AppData\Local\Google\Chrome\User
Data\PepperFlash\*
\Device\HarddiskVolume2\Users\Magnum\AppData\Local\Google\Chrome\User
Data\SwiftShader\*
\Device\HarddiskVolume2\Program Files\MSBuild\*
\Device\HarddiskVolume2\Program Files\Reference Assemblies\*
\Device\HarddiskVolume2\Program Files\Windows Defender\*
\Device\HarddiskVolume2\Program Files\Windows Journal\*
\Device\HarddiskVolume2\Program Files\Windows Media Player\*
\Device\HarddiskVolume2\Program Files\Windows NT\*
\Device\HarddiskVolume2\Program Files\Windows Photo Viewer\*
\Device\HarddiskVolume2\Program Files\Windows Portable Devices\*
\Device\HarddiskVolume2\Program Files\Internet Explorer\*
\Device\HarddiskVolume2\Program Files (x86)\Internet Explorer\*
\Device\HarddiskVolume2\ProgramData\Microsoft\Windows Defender\*
\Device\HarddiskVolume2\Program Files (x86)\Google\*
\Device\HarddiskVolume2\Program Files (x86)\Microsoft Silverlight\*
\Device\HarddiskVolume2\Program Files\7-Zip\*
\Device\HarddiskVolume2\Program Files (x86)\Common Files\*
\Device\HarddiskVolume2\ProgramData\Microsoft\IdentityCRL\*
\Device\HarddiskVolume2\Users\Magnum\Desktop\Dbgview.exe*
\Device\HarddiskVolume2\Users\Magnum\Desktop\SoftMaker Office 2010\*
\Device\HarddiskVolume2\Users\Magnum\Desktop\LibreOfficePortable\*
\Device\HarddiskVolume2\Users\Magnum\Desktop\☻♫♪♥♠слу́шаю! … אַ דער
אַ דערNǐhǎo!مَرْحَبَاً\*
\Device\HarddiskVolume3\downloads\TrueCrypt\*
blacklist|
\Device\HarddiskVolume2\Windows\notepad.exe|
\Device\HarddiskVolume2\Program Files\Internet Explorer\iexplore.exe|
\Device\HarddiskVolume2\Program Files (x86)\Internet Explorer\iexplore.exe|
```

Make sure that you divide the file into the two parts whitelist* and blacklist|. It is important not to mix filenames and paths from one section into another section. Meaning, **please do not** something like:

```
whitelist*
\Device\HarddiskVolume2\Windows\*
\Device\HarddiskVolume2\Windows\notepad.exe|
blacklist|
\Device\HarddiskVolume2\Program Files\Internet Explorer\iexplore.exe|
\Device\HarddiskVolume2\Program Files\Common Files\*
\Device\HarddiskVolume2\Users\Magnum\AppData\Local\Google\Chrome\User
Data\PepperFlash\*
```

The (bad) example above will not work properly and might end up in a halting or crashing system. So take care while defining the tuersteher_light.ini file!

The driver was compiled for Microsoft Windows Vista, 7, 8 and 8.1 (32/x86 and 64/x64 bit versions). To start it up go into the driver binary's path regarding your version of Windows and execute the corresponding *.inf file in order to install the driver.

If you use a 32bit Version of Windows, driver signing is not required and you should be able to run Tuersteher Light just out of the box. In Windows Vista, 7, 8 and 8.1 x64 you need to digitally sign any driver. This is Microsoft policy for all kernel drivers in recent versions of Windows, for more details see Driver Signing Requirements for Windows.

As a temporary work around you can also disable the signature check in Window's boot options. An alternative way is to digitally sign the driver by yourself using a test certificate and booting up Windows into the TESTSIGNING mode:

Download, install the Windows Driver Kit, then open a WDK Build Environment console as Administrator.

Run the MakeCert.exe tool to create a test certificate, e.g. with:

```
MakeCert -r -pe -ss TestCertStoreName -n "CN=TestCertName" CertFileName.cer
Install the test certificate with CertMgr.exe, e.g. with
CertMgr /add CertFileName.cer /s /r localMachine root
Sign Tuersteher_Light.sys with the test certificate, e.g. with
SignTool sign /v /s TestCertStoreName /n TestCertName Tuersteher_Light.sys
Enable Windows TESTSIGNING mode, to do this, run the command
Bcdedit.exe -set TESTSIGNING ON
Restart Windows.
```

After these steps you should be able to run the driver without disabling driver signature check every time. If you have any questions, suggestions or need assistance feel free and contact me by e-mail.

3 Some words regarding Tuersteher and what is it all about

Tuersteher Light is part of my private projects to build up endpoint security solutions that I can trust. There are a lot of players out there building heavy weight endpoint security solutions to defend against zero days and other new threads not handled by the ordinary anti virus or desktop firewall.

These tools are great but I really do not trust them at the end. I am developing such tools and drivers since 2006 and saw lots of companies claiming that they got the ultimate solution against new threats and attacks. In the end all of these companies suffered at some point although their shiny sales brochures praised their stuff to highest heaven.

Most solutions on the market right now come with hundreds of executables and dynamic libraries, they install several services, drivers, slow down system performance and at the end I really do not know what they really do on my machine. Most of them also send back forensics data and information to their "mothership" or update their engines via the internet, although they claim not to be like a classic anti-virus needing updates all the time.

We as users do not really know what information an data they transmit. For the ordinary user such solutions are like a a closed book and in most cases are not applicable to a privat user's environment because they are build to scale on big infrastructures people like you and me do not run at home.

That's why I started writing my own tiny, fast and reliable protection drivers that do not need any executable, dll or service besides their one and only driver executable file. My drivers do not talk back to me or my servers - they are just simple and only do what they were intended to do. With Tuersteher, Tuersteher Light, MZWriteScanner it is possible to protect and mitigate against many attacks out there without spending a lot of bucks and yearly fees (most of my solutions are totally free of charge). My drivers are intended to enable you to look behind the scenes and letting you to come of age regarding your Windows based systems.

If you want to know more about kernel based monitoring I highly recommend to read my whitepaper on http://www.bitnuts.de/KernelBasedMonitoring.pdf.

If you need additional information or want to test Tuersteher (Light) in a big scaled infrastructure/environment I am open minded and interested in new challenges using my tools and drivers, so do not hesitate and contact me. I believe in my drivers and I can go into a battle against the heavy weight endpoint protection champs in the scene without loosing my face, because all of them put their pants on one leg at a time.

3.1 Further reading

- Application Whitelisting: Approaches And Challenges at
  http://airccse.org/journal/ijcseit/papers/2512ijcseit02.pdf
- See Microsoft's Driver Signing Requirements for Windows at
  http://msdn.microsoft.com/en-us/windows/hardware/gg487317.aspx
- For more information on TESTSIGNING check out
  http://msdn.microsoft.com/en-us/library/windows/hardware/ff553484(v=vs.85).aspx
- Whitepaper on Kernel Based Monitoring
  http://www.bitnuts.de/KernelBasedMonitoring.pdf