Ludwig-Maximilians-Universität München (Hrsg.): Seminar Virtualisierte Systeme,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017

Windows Subsystem for Linux

Marius Rusu[1], Julia Sommer[2]

Abstract: The Windows Subsystem for Linux (WSL) is a new feature that makes it possible to run native Linux command-line tools directly on Windows 10. This paper shall investigate its architecture and implementation and compare it to other common Linux virtualizations, such as virtual machines and containers. We conclude that there are differences regarding the level of isolation from the host operating system, the number of instances and the performance. Our results show that Windows Subsystem for Linux might not yet be optimized enough to replace more common virtualizations of the Linux operating systems. We anticipate our project paper to provide an overview of how different virtualizations of operating systems work and stimulate for possible improvements of the Windows Subsystem for Linux.

Keywords: Windows Subsystem for Linux; virtualization; Linux virtualization; container; virtual machine

1 Microsoft's reasons for WSL

Many programmers that work with open source, Linux-based tools such as Pearl and Python are facing difficulties on Windows. Especially when it comes to server infrastructure, programmers work with applications native to Linux, since most of the servers are also powered by Linux. Those applications are usually not optimized for Windows and programmers depend on workarounds, such as containers and virtual machines. Alternatively, they switch to Linux or other Unix-based operating systems [Ha16c].
Based on their feedback, Microsoft has made investments that improve cmd, PowerShell and many other command-line tools and developer scenarios. According to Mike Harsh, they further decided to grow their command line family by adding real, native Bash and with it support for Linux command-line tools, which run directly on Windows in an environment that behaves like Linux. To accomplish this, Microsoft worked together with Cononical and published the Windows Subsystem for Linux in April 2016 [Ha16c].

[1] Ludwig-Maximilians-Universität München, Fakultät für Informatik, Oettingenstraße 67, 80538 München, Deutschland rusu.marius97@gmail.com

[2] Technische Universität München, Fakultät für Informatik, Boltzmanstraße 3, 85748 Garching, Deutschland sommerjulia99@gmail.com

2 Windows Subsystem for Linux

The following will provide information about the Windows Subsystem for Linux, based on Microsoft's publication.

2.1 General Concept

"Windows Subsystem for Linux is a collection of [user mode and kernel mode] components that enable native Linux ELF64 binaries [, which can be compared to Windows' .exe] to run on Windows [Ha16b]." User mode applications are low privileged and depend on system calls to operate on kernel mode. Only in kernel mode low level operations directly handled by the operating system can be executed. In WSL we have the bash.exe, similar to the Windows command prompt (cmd), running in user mode and initiating the Linux instance. Further, this instance submits, if necessary, native Linux system calls to be executed on a Linux kernel. However, by virtualizing a Linux kernel interface those system calls can be executed directly on the Windows kernel. This virtualization is done by the LXCore/LXSS (Linux Core/Linux Subsystem) running in kernel mode [Ha16b].

2.2 Basic Architecture

Windows Subsystem for Linux " is primarily comprised of:

1. LX Session Manager
2. LXCore/LXSS
3. Pico processes

[Ha16b]"

As depicted in the image below (1), the user initiates the Windows Subsystem for Linux by launching the bash.exe on Windows. According to Nick Judge, this application then calls the LXCore/LXSS (green arrow), which is a driver behaving like a Linux kernel and working in coordination with the Windows kernel. The driver would then spin up the native Linux process /bin/bash (purple arrow), which is the Linux standard-shell. All other Linux processes run under /bin/bash in the so-called Linux instance that can be considered as a container or a virtualized operating system environment [Ha16a].

"By wrapping unmodified Linux binaries into Pico processes, we enable Linux system calls to be directed into the Windows kernel [Ha16b]." A Pico process itself is an empty process, as far as the Windows kernel is concerned and therefore cannot be handled by the Windows kernel, but instead is redirected to the LXCore/LXSS (red arrow) [Ha16a]. Therefore "Pico processes and drivers [LXCore/LXSS] provide the foundation for the Windows Subsystem for Linux [Ha16b]."

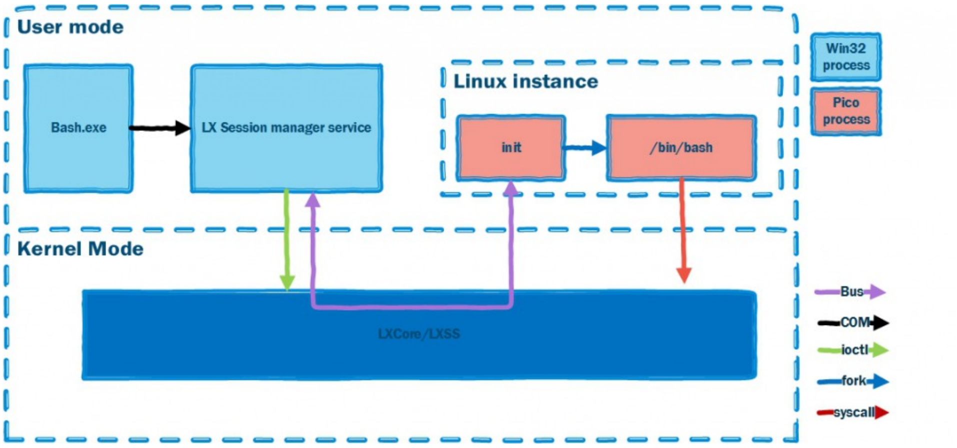


Fig. 1: Main components of WSL [Ha16b]

2.3 Implementation of its components

The Pico process concept was the result of the Drawbridge project in MSR (Microsoft Research). This project aimed to implement a lightweight method to run an application and its OS in an isolated environment, separated from the host OS [Ha16a]. This was achieved, not by a virtual machine, which would be too resource consuming, but by wrapping the application and guest OS into the address space of a single process of the host OS. The Drawbridge Pico process, which is a lightweight, secure isolation container, made this possible. It is built from an OS process address space, but with all traditional OS services removed. That is why the host OS cannot read its content and " all ABI [(Application binary interface)] calls are serviced by the security monitor, which plays a role similar to the hypervisor or VM monitor in traditional hardware VM designs [Mi11]."
In case of Windows Subsystem for Linux this is exactly the point where Pico processes are redirected to the LXCore/LXSS. According to Jack Hammons, those drivers represent a Linux-compatible kernel interface without containing any code from an actual Linux kernel. Their task is to translate the Linux system calls from the Linux instance to the equivalent Windows system call, so that the Windows kernel can meet the demands [Ha16b]. "Where there is no reasonable mapping the Windows kernel mode driver must service the request directly [Ha16b]." Since the Windows kernel was originally designed to support multiple operating systems, Microsoft had to reactivate old functionality and enhance it for performance and correctness, says Nick Judge [Ha16a]. Those changes in the Windows kernel enable it to execute even foreign operations like fork() that are implemented differently on Windows and Unix-based systems. Windows applications however do not have access to those specific Linux system calls and therefore no interference or security issues can occur.

Moving on to the file system, it was designed to meet the following goals:

1. support of Linux file systems
2. interoperability with Windows

[Ha16b]

"VolFs is a file system that provides full support for Linux file system features, including Linux permissions that can be modified through operations such as chmod and chroot [...] [Ha16b]" and other features. Due to the fact that VolFs file system is not able to interoperate with Windows, a second file system named DriveFs is implemented. With DriveFs, all Windows volumes and files can be accessed under /mnt. To make this possible, the DriveFs file system meets Windows requirements such as legal file names and Windows security, but loses some of the Linux features mentioned above [Ha16b].

3 Alternatives to Windows Subsystem for Linux

The following will discuss other possibilities of running Linux on Windows and compare them to the Windows Subsystem for Linux.

3.1 Virtualization via virtual machine

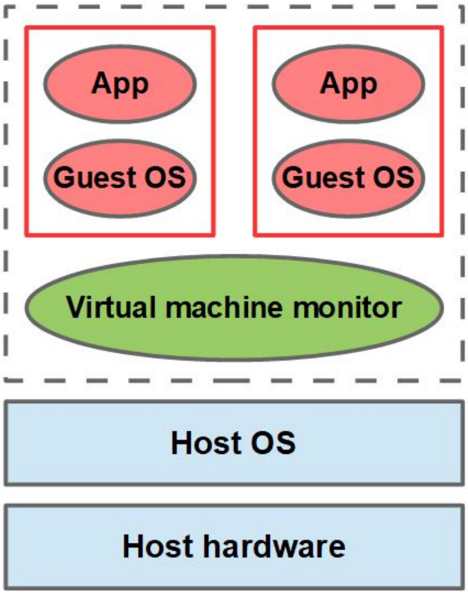
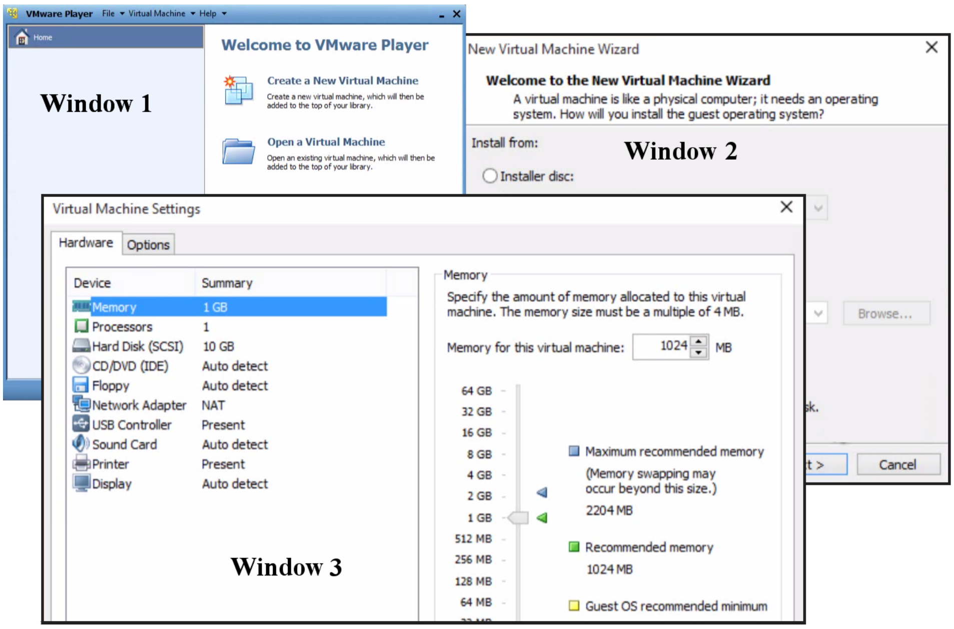


Fig. 2: virtual machine overview

"The virtual machine concept allows the same computer to be shared as if it were several. IBM [(International Business Machines Corporation)] defined the virtual machine as a fully protected and isolated copy of the underlying physical machine's hardware [Ro01, p. 2]." Therefore, it is possible to run different applications on different operating systems at the same time on the same hardware as depicted in the image above (2). The VMM (virtual machine monitor) is responsible for hosting the guest virtual machine [Ro01, p. 3]. Its task is firstly to coordinate the access of the guest operating systems on the host's hardware and secondly to handle traps adequately. Traps are created when a guest operating system is trying to execute a privileged operation. In that case, the virtual machine Monitor emulates its function in order to be executed on the host operating system.



Welcome to VMware Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

New Virtual Machine Wizard

Welcome to the New Virtual Machine Wizard
A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:
○ Installer disc:

Virtual Machine Settings

Hardware | Options

Device	Summary
Memory	1 GB
Processors	1
Hard Disk (SCSI)	10 GB
CD/DVD (IDE)	Auto detect
Floppy	Auto detect
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Memory
Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

Memory for this virtual machine: 1024 MB

Maximum recommended memory
(Memory swapping may occur beyond this size.)
2204 MB

Recommended memory
1024 MB

Guest OS recommended minimum

Fig. 3: VMware Player's graphic interface

A common tool for creating a virtual machine is VMware Player by VMware Inc. This application hides all the work of the virtual machine Monitor and comes with a graphic interface that can be seen above (Window 1) (3). For creating a new virtual machine an image file of the guest operating system is needed, which can be either on a disc or downloaded from the internet (Window 2) (3). The user can then decide on how many resources, e.g. memory space and CPU cores the new virtual machine can use (Window 3) (3). Finally, VMware Player launches the guest operating system in a new window as a fully functional and isolated operating system.

3.2 Virtualization via Container

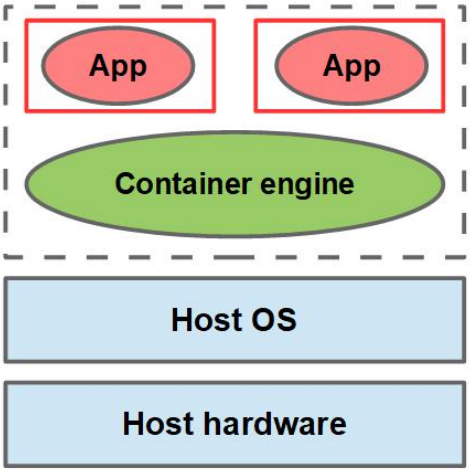


Fig. 4: Container overview

Rather than virtualizing the hardware, containers use the host operating system and share its kernel. According to V. Badola containers are " stripped down virtual machines running just enough software to deploy an application [Ba15]." Instead of a virtual machine monitor there is a container engine running on top on the host operating system (4). The container engine is responsible for deploying containerized applications. It allocates cores and memory to containers, ensures spatial isolation from the host and security [Do17].

One of the most common container engines is the open source Docker, which developed a method to give containers better portability. According to Margaret Rouse "with Docker containers, there are no guest OS environment variables or library dependencies to manage [Ma14]." In order to run a container in Docker, a so called Dockerfile is needed, which provides the Docker engine with information about the desired container. Dockerfiles contain firstly the image of the system environment of the container, secondly the application to run and thirdly the port for communication between host and container. Docker Hub provides users with a collection of images e.g. Ubuntu for download. All those steps are done in the console and a text editor for the Dockerfile, since there is no graphic interface.

3.3 Comparison to Windows Subsystem for Linux

In view of all the facts, Windows Subsystem for Linux is neither a virtual machine nor a container, but rather something in between. It is not a fully virtualized Linux operating system, which would be the result of a virtual machine, but it is also more than just a container environment. There are three major differences that result both, in advantages and disadvantages.

First of all, Windows Subsystem for Linux is not isolated from the host operating system but rather works hand in hand. This can be seen by the fact that processes are directly handled by the host kernel instead of a virtual machine monitor or container engine. Further, the file systems VolFs and DriveFs are both accessible from Windows and Linux, while containers and virtual machines are usually invisible to each other. This can be a disadvantage, especially when running possibly dangerous or unstable programs. Even though there are security measures, there have been fears that malware can reach Windows via the Linux Subsystem. [Tu17]. It can also be an advantage for common users, who want to access and edit Windows files with open source, Linux-native tools or want to run their programs on both operating systems for debugging, without having to launch a virtual machine or container.

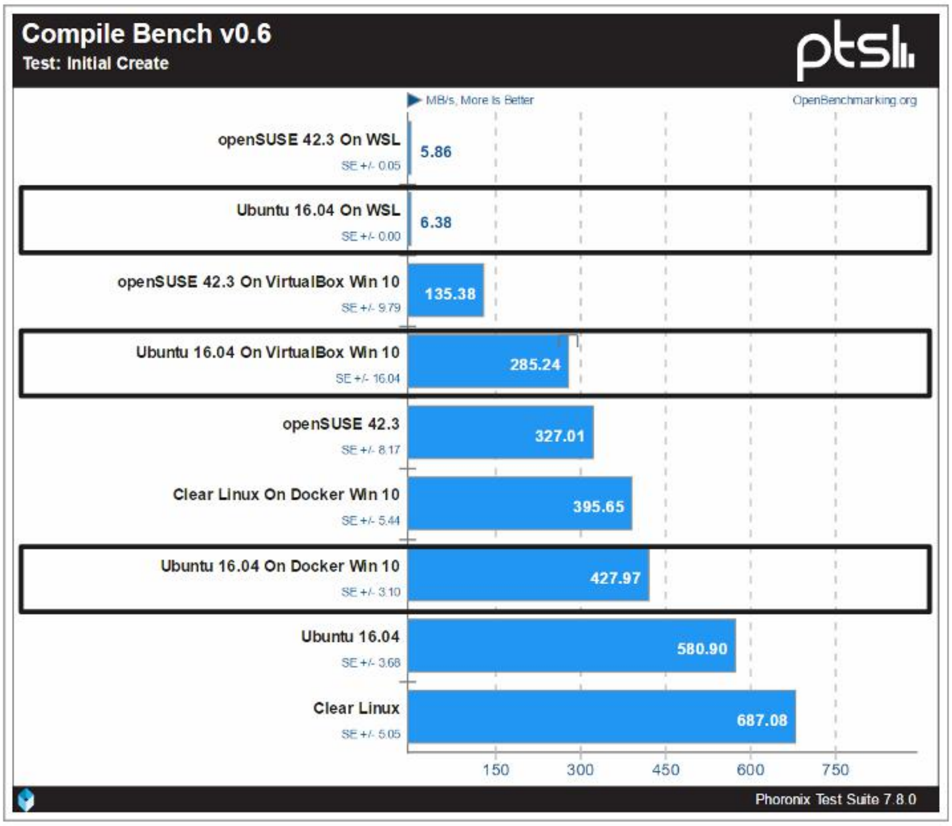
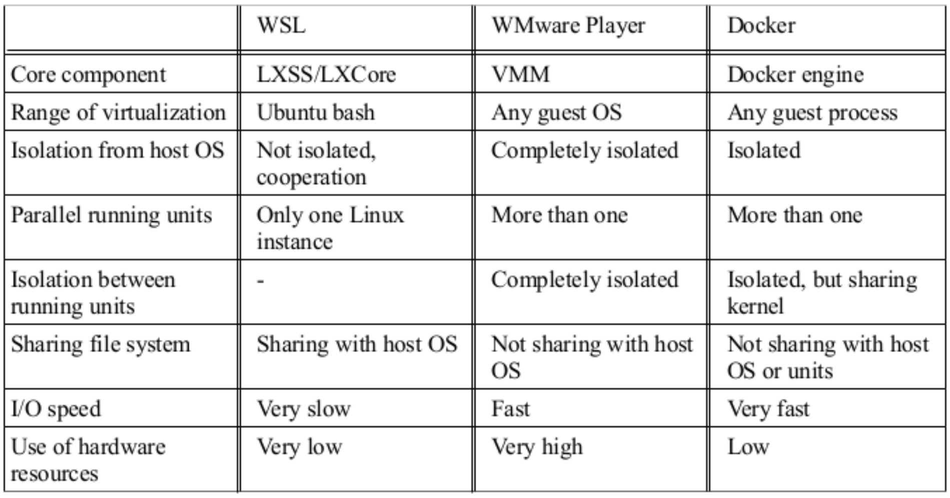


Fig. 5: Compile benchmark
[La18]

Secondly, there exists only one Linux instance for each user. All programs run together under /bin/bash and cannot be separated from each other. Even launching Ubuntu more than once will always result in the same Linux instance [Ha16b]. This is a disadvantage when an isolated environment is needed. This case is similar to the security concerns between Windows and the Linux Subsystem mentioned above, but this time it affects programs inside the Linux instance. In both cases a fully isolated virtual machine or container is the better choice.

Thirdly, there are differences regarding the performance of those three attempts of virtualization. According to tests run by Phoronix, the I/O performance of the Windows Subsystem for Linux rather lags behind as can be seen in the image above (5). The Windows Subsystem for Linux is only able to compile 6.38 MB/s, while the virtual machine VirtualBox compiles 285.24 MB/s and Docker even 427.97 MB/s, which is about 70 times faster. This issue has become even worse since the Spectre/Meltdown updates, which were supposed to mitigate those security vulnerabilities. " If exploited, these vulnerabilities can give hackers unprecedented access to compromised systems and widespread liberty to steal a broad variety of confidential, sensitive data [Pe18]." The poor I/O performance is certainly the biggest disadvantage of the Windows Subsystem for Linux and the reason it cannot fully replace virtual machines or containers yet. On the other hand, Windows Subsystem for Linux performs at least as good as virtual machines and containers when it comes to MP3/FFMPEG encoding and many other benchmarks e.g. AOBench [La18]. Further, Windows Subsystem for Linux is far less resource consuming than both VMware Player and Docker and therefore does have very little impact on the host OS performance. A summary of those aspects can be seen below (6).



	WSL	WMware Player	Docker
Core component	LXSS/LXCore	VMM	Docker engine
Range of virtualization	Ubuntu bash	Any guest OS	Any guest process
Isolation from host OS	Not isolated, cooperation	Completely isolated	Isolated
Parallel running units	Only one Linux instance	More than one	More than one
Isolation between running units	-	Completely isolated	Isolated, but sharing kernel
Sharing file system	Sharing with host OS	Not sharing with host OS	Not sharing with host OS or units
I/O speed	Very slow	Fast	Very fast
Use of hardware resources	Very low	Very high	Low

Fig. 6: Comparison of WSL, VMware Player and Docker

4 Conclusion

The Windows Subsytem for Linux offers accustomed Linux users a simple way to access common Linux features on Windows 10. It uses the Windows kernel to execute Pico processes via the LXCore/LXSS, that maps Linux system calls to equivalent Windows system calls. Those Pico processes are, as far as the Windows kernel is concerned, empty processes and are therefore redirected to the LXCore/LXSS for handling. This procedure is new and distinguishes itself from virtual machines and containers. In addition, it is more user-friendly and resource saving, but does not quite reach the performance of the classic approaches. Therefore, Windows Subsystem for Linux is a good alternative for multi-platform programmers and those who want to use native Linux software, especially when the tasks are not too expensive. However, when it comes to commercial use, such as server hosting, Windows Subsystem for Linux does not provide the requested performance and speed yet. Altogether WSL represents an interesting approach on virtualization and with Microsoft working on optimizations it has the potential to prove itself as a practical alternative to virtual machines and containers.

References

[Ba15] Badola, V.: Container Virtualization: what makes it work so well?, https://cloudacademy.com/blog/container-virtualization, Online, accessed 30-April-2018; 19.55 Uhr, Oct. 2015.

[Do17] Donald, F.: Virtualization via Container, https://insights.sei.cmu.edu/sei_blog/2017/09/virtualization-via-containers.html, Online, accessed 30-April-2018; 19.58 Uhr, Sept. 2017.

[Ha16a] Hammons, J.: Pico Process Overview, https://blogs.msdn.microsoft.com/wsl/2016/05/23/pico-process-overview/, Online, accessed 30-April-2018; 19.30 Uhr, May 2016.

[Ha16b] Hammons, J.: Windows Subsystem for Linux Overview, https://blogs.msdn.microsoft.com/wsl/2016/04/22/windows-subsystem-for-linux-overview/, Online, accessed 18-April-2018; 19.32 Uhr, Apr. 2016.

[Ha16c] Harsh, M.: Run Bash on Ubuntu on Windows, https://blogs.windows.com/buildingapps/2016/03/30/run-bash-on-ubuntu-on-windows/#xKYy1Osl93c7kTAv.97, Online, accessed 30-April-2018; 19.21 Uhr, Mar. 2016.

[La18] Larabel, M.: Windows 10 WSL vs. Linux Performance For Early 2018, https://www.phoronix.com/scan.php?page=article&item=wsl-february-2018&num=2, Online, accessed 30-April-2018; 20.14 Uhr, Feb. 2018.

[Ma14] Margaret, R.: containerization (container-based virtualization), https://searchservervirtualization.techtarget.com/definition/container-based-virtualization-operating-system-level-virtualization, Online, accessed 30-April-2018; 20.01 Uhr, 2014.

[Mi11] Microsoft: Drawbridge, https://www.microsoft.com/en-us/research/project/drawbridge/?from=http%3A%2F%2Fresearch.microsoft.com%2Fen-us%2Fprojects%2Fdrawbridge%2F, Online, accessed 30-April-2018; 19.38 Uhr, Sept. 2011.

[Pe18] Perez, J. C.: Meltdown / Spectre Mitigation Is a Work in Progress, https://blog.qualys.com/news/2018/01/16/meltdown-spectre-mitigation-is-a-work-in-progress, Online, accessed 30-April-2018; 20.09 Uhr, Jan. 2018.

[Ro01] Rose, R.: Survey of System Virtualization Techniques, Oregon State University, Mar. 2001.

[Tu17] Tung, L.: Windows 10's Subsystem for Linux: Here's how hackers could use it to hide malware, https://www.zdnet.com/article/windows-10s-subsystem-for-linux-heres-how-hackers-could-use-it-to-hide-malware/, Online, accessed 30-April-2018; 20.04 Uhr, Sept. 2017.