TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Triin Viitmaa 192557IVCM

Forensic analysis of Windows Subsystem for Linux on Windows 11

Master's thesis

Supervisors: Shaymaa Mamdouh Khalil
MSc
Sander Medri
MSc

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Triin Viitmaa 192557IVCM

Windows 11 liidese Windows Subsystem for Linux kriminalistiline analüüs

Magistritöö

Juhendajad: Shaymaa Mamdouh Khalil
Msc
Sander Medri
MSc

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Triin Viitmaa

16.05.2022

Abstract

Windows Subsystem for Linux (WSL) is a system designed by Microsoft for running Linux executables in Windows in a way that is more seamless compared to traditional virtualization methods. WSL is targeted at software developers and anyone else who needs to use Windows and Linux operating systems simultaneously.

WSL has evolved rapidly since its introduction in 2016. WSL version 2 (WSL2), which is currently the newest version, came with a completely updated architecture compared to the first version and includes support for graphical user interface applications in Windows 11.

Unfortunately, WSL exposes the host operating system to many new avenues of attack. Therefore, it is important that specialists such as forensic investigators and incident responders know where to find information and evidence regarding WSL usage on a system. The current research and documentation in this area are insufficient to support fast and thorough examinations of systems leveraging WSL.

This thesis aims to discover what forensic artifacts can be found when WSL is used on the Windows 11 operating system and where these artifacts appear. The experiments conducted for this purpose were designed around basic user behaviour such as carrying out file operations, downloading files from the Internet, and opening applications.

The conducted analysis demonstrates that WSL1 and WSL2 have certain forensic artifacts in common. However, because of the architectural differences between these systems, there are also a lot of differences from a forensics perspective. For example, with WSL1, files that are stored on the Linux file system can be found directly on the Windows file system. However, in the equivalent situation with WSL2 the files are stored in a Hyper-V virtual hard disk instead. From the network perspective, WSL2 stands out clearly whereas WSL1 does not.

This thesis is written in English and is 67 pages long, including 7 chapters, 27 figures and 6 tables.

Annotatsioon

Windows 11 liidese Windows Subsystem for Linux kriminalistiline analüüs

Windows Subsystem for Linux (WSL) on Microsofti arendatud Windowsi alamsüsteem, kus on võimalik jooksutada Linuxi programme. WSL loodi tarkvaraarendajatele ja kõigile, kes peavad kasutama Windowsi ja Linuxi operatsioonisüsteeme samaaegselt.

WSL on kiirelat arenenud alates aastast 2016, WSL-i teine versioon muutis täielikult alamsüsteemi arhitektuuri, alates Windows 11 on toetatud ka graafilised programmid.

WSL ei ole turvameede, vastupidi, see loob palju uusi võimalusi süsteemi ründamiseks, seega intsidentidele reageerijad peavad olema teadlikud, kust leida tõendeid alamsüsteemi kasutamise kohta. Praegu olemasolev uurimistöö ja dokumentatsioon on ebapisiavad, et toetada kiriet ja põhjalikku WSL süsteemi analüüsi.

Lõputöö eesmärk on leida millised digitaalsed tõendid asuvad Windows 11 operatsioonisüsteemis. Eksperiment baseerus põhilisyrl kasutaja tegevustel nagu operatsioonid failidega, Internetist alla laadimine ja programmide avamine.

Lõputöö demonstreerib, et WSL1 ja WSL2 jagavad mõningaid küberkriminalistilisi tõendeid, tulenevalt arhitektuurilistest erinevustest versioonide vahel, on paljud digitaalsed tõndid siiski erinevad. WSL1 failid, mis on salvestatud Linuxi failisüsteemis on leitavad ka Windowsi failisüsteemist, aga WSL2 failid asuvad Windowsis Hyper-V virtuaalselt kettal. Võrguliikluse vaatest on WSL2 liiklust võimalik kohalikus masina omast eristada, WSL1 puhul mitte.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 67 leheküljel, 7 peatükki, 27 joonist, 6 tabelit.

List of abbreviations and terms

API	Application Programming Interface
BAM	Background Activity Moderator
CLI	Command Line Interface
CPU	Central Processing Unit
ELF	Executable and Linkable Format
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HVCI	Hypervisor-Protected Code Integrity
ICMP	Internet Control Message Protocol
IP	Internet Protocol
KAPE	Kroll Artifact Parser and Extractor
MFT	Master File Table
NAT	Network Address Translation
NVMe	Non-Volatile Memory express
PC	Personal Computer
PCAP	Packet Capture
RAM	Random Access Memory
SID	Security Identifier
SRUM	System Resource Usage Monitor
SSD	Solid State Drive
TPM	Trusted Platform Module
UEFI	Unified Extensible Firmware Interface
USB	Universal Serial Bus
VHD	Hyper-V virtual disk

VM	Virtual Machine
WSL1	Windows Subsystem for Linux version 1
WSL2	Windows Subsystem for Linux version 2

Table of contents

List of figures

List of tables

1 Introduction

In 2016 Microsoft introduced the first version of Windows Subsystem for Linux (WSL). The second version came out three years later in 2019 and it is now the default option. Currently, users can install either WSL 1 or WSL 2 on Windows 10 and Windows 11. According to Microsoft, WSL was mostly designed for developers who need to use Linux along with Windows and for whom traditional solutions involving virtual machines are not practical due to compatibility and management issues.

A big problem with this easily accessible subsystem is that it creates a whole new environment for users and attackers to hide in. For incident responders and computer crime investigators, it is important to understand what evidence is left behind when WSL is used in a modern Windows operating system.

Therefore, the main research question is what artifacts and where WSL leaves on computers using a Windows 11 operating system. Similarly, it is important to understand the characteristics of WSL network traffic. For example, to understand if it is possible to distinguish the host's traffic from WSL's traffic.

The purpose of this work is to discover forensic artifacts created by WSL 1 and WSL 2 on the Windows 11 operating system and to help investigators find WSL-related evidence in Windows 11.

The design of the experiments was based on basic user behaviour such as creating and moving files, executing applications, and browsing the Internet. WSL-related network traffic is collected directly from the test machine and disk images are acquired after the computer has been shut down. Linux forensics is out of the scope of this thesis.

To the author's knowledge, this is the first forensic analysis paper examining WSL on Windows 11. Not a lot of research has been conducted on related topics. The presentations and research articles that exist focus on one specific theme in one version of WSL and exclusively on Windows 10. None of the previous research includes WSL graphical interface evidence which is Windows 11 specific but can be essential for investigators because humans are likely to prefer graphical applications over the command line. Previously published papers were pointing out that the field needs more adequate research even in the Windows 10 environment. This thesis not only covers various host-based forensic artifacts but also examines network artifacts.

Because this research was conducted on Windows 11 there were limitations with analysing volatile data. Namely, the most popular memory forensics framework Volatility did not support Windows 11 at the time of writing this thesis. Additionally, network traffic that was analysed was captured directly on the host. In reality, it would be almost impossible to have this kind of evidence without remote monitoring and collecting.

2 Background

Windows Subsystem for Linux was first announced in 2016 [1], and exited from beta one year later in 2017 [2]. The purpose of WSL is to provide support for native Linux applications on a Windows operating system environment. Windows Subsystem for Linux is not the first time Windows users can have functionality similar to Linux, Cygwin project was released 1995, it is a collection of GNU and Open-Source tools, but with Cygwin users cannot run native Linux applications on Windows, custom made apps are required [3].

2.1 WSL1

WSL1 allows Linux binary executables (ELF) run on Windows without recompilation by executing unmodified ELF64 binaries on top of the Windows Kernel [4]. Executing Linux binaries in native speed is possible due WSL1 interface, it translates Linux system calls into Windows system calls [4].

In 2017, when WSL was out of beta, Windows 10 users got a possibility to directly execute userland Linux applications, while each executed program was tied with a pico process [5]. “This allows users to execute ELF binaries without the need for a virtual machine, source code modification, or an intermediate application” [5]. Pico process and drivers (lxss.sys and lxcore.sys) are responsible for translating Linux syscalls, unmodified Linux binaries are placed to Pico process so the Linux calls will be directed into the Windows kernel [6]. “The lxss.sys and lxcore.sys drivers translate the Linux system calls into NT APIs and emulate the Linux kernel [6].”

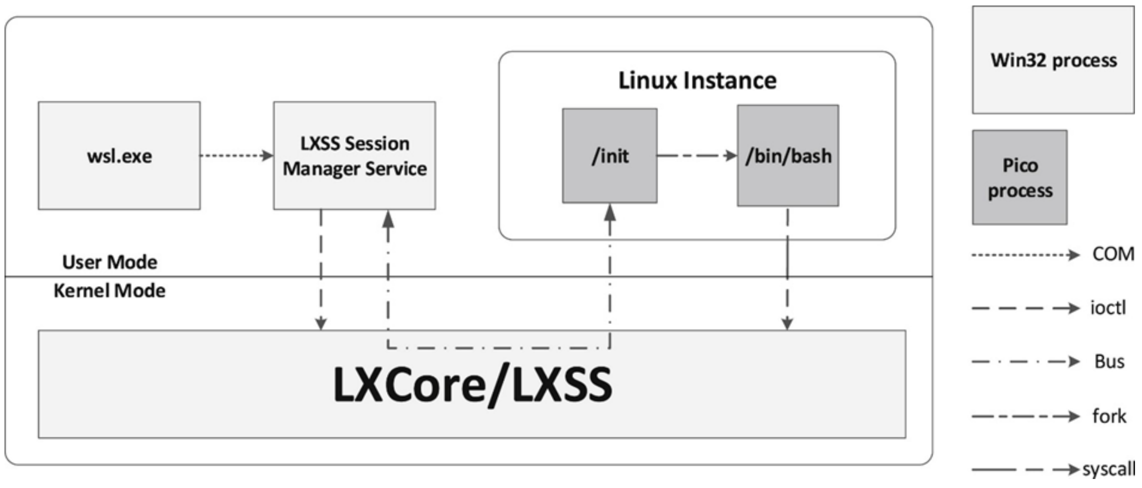


Figure 1. Communication between components of WSL [5]

The LXSS Session Manager Service is a broker to the Linux subsystem driver and is the way wsl.exe invokes Linux binaries, it is also responsible for allowing only one process at a time around install and uninstall [6]. A user can open the WSL by executing wsl.exe, bash.exe or <distro>.exe from command line [5]. Each Linux process launched by particular user will go into specific instance - a data structure responsible for tracking all LX processes, thread, and runtime state [6]. The first time NT process request launching a Linux binary a new instance is created, after terminating the NT client, the Linux instance and every process launched inside of it will be closed [6]. Communication between components of WSL1 are shown in Figure 1.

One of the motivations of developing Windows subsystem for Linux was to offer users a possibility to lose management of file sharing between host and a virtual machine and allowing users to work with their files as they would on native operating system [7]. To accomplish that WSL provides access to Windows files by emulating full Linux behaviour for the internal Linux file system [7]. In WSL1, Linux files are stored on the same drive as Windows [8]. Two main components responsible for that ability are VoIF and DrvFs. VoIF is the primary file system used by WSL, it stores the Linux system files, and content of Linux home directory [7]. There are two separate mounts to preserve home directory with personal files even after WSL is uninstalled [7]. DrvFs file system is used to facilitate integration with Windows drives and files [7].

2.2 WSL2

The second version of Windows Subsystem for Linux was introduced in 2019 and has different architecture to run ELF64 and ELF32 Linux binaries on Windows with full system call compatibility and increased file system performance [9].

A real Linux kernel is used inside of lightweight VM utility, virtualization technology is based on Hyper-V [10]. WSL2 architecture is shown in Figure 2.

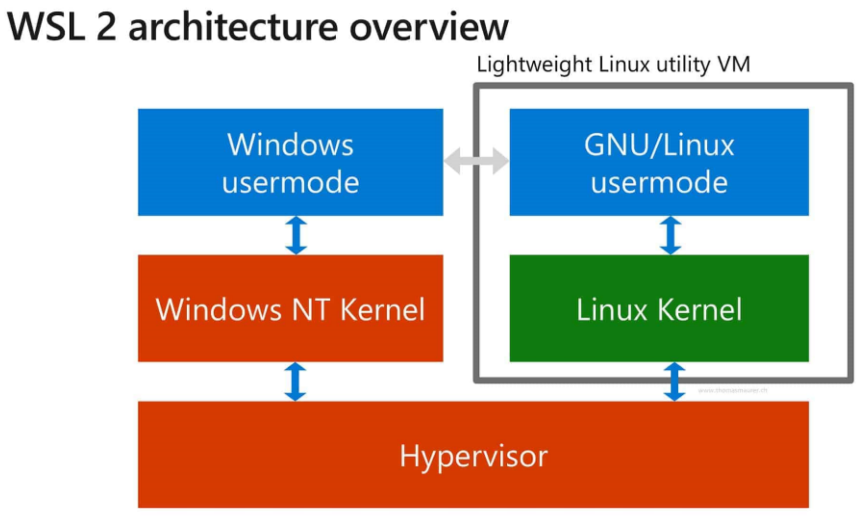


Figure 2. WSL2 architecture overview [8]

WSL2 is not traditional isolated Virtual Machine, VM can be slow to boot up, is resource and management intensive [11]. The Linux kernel in WSL2 is specially built with size and performance optimization based on latest stable branch available at kernel.org [11]. To reduce management kernel improvements security fixes are tied to Windows updates [11].

The speed increase in WSL2 has been achieved by two main reasons, first by storing Linux files in a virtual hard disk which uses the EXT4 file system [12]. Second reason is that WSL2 has real Linux kernel with full system call compatibility [11]. Latter is also the reason why users do not have to wait anymore for WSL team to implement updates, and Linux applications like Docker can run inside of WSL2 [11].

WSL2 uses a Network Address Translation (NAT) service for its virtual network [11], which means WSL2 and host machine are not in the same network, but this can be resolved by forwarding TCP ports of WSL2 services to the host [13].

Starting from Windows 11 build 22000 is possible to run Linux graphical user interface applications [14]. WSL2 is now the default version, but WSL1 can be run simultaneously. Ubuntu is the default distribution while installing WSL without specifying flag, other distributions can be downloaded from Microsoft Store or within installation process with flag “-d” and distribution name.

2.3 Comparison of WSL1 and WSL2

Main reasons Microsoft recommends upgrading WSL1 to WSL2 are increased file system performance and full system call compatibility [11]. Only exception is performance across OS file system, which is better in WSL1 [11]. WSL1 usage is recommended in case of project files must be stored in the Windows file system, a project requires cross-compilation, a project needs access to a serial port or USB, when memory usage is limited or strict, or bridged network adapter is required [11]. WSL1 do not support 32-bit but WSL2 does. WSL1 and WSL2 can be run side by side.

2.4 Windows 11

Windows 11 was released in October 2021, it is free to upgrade from Windows 10, but the host machine has to meet new system requirements. Host must have a compatible CPU [15], at least 4GB of RAM, at least 64GB of storage, UEFI, Secure Boot, and TPM 2.0 enabled [16]. The latter is the main reason many computers do not meet the requirements. “Microsoft claims that a combination of their latest security features: Windows Hello, Device Encryption, virtualization-based security, hypervisor-protected code integrity (HVCI) and Secure Boot have been shown to reduce malware by 60 percent” [17].

For forensic perspective some artifacts like prefetch, LNK files, Jumplists, Recycle Bin, Amcache, AppCompatCache, Registry and Event Logs did not change according to Eric Zimmermann’s Twitter post [18]. In addition to security upgrades, Windows 11 changed the user interface design to simplify the user experience and more [16], but these are not relevant in this context.

In this work we test Windows 11 unique ability to run Linux graphical user interface applications via WSL2.

2.5 Windows Forensics

Windows forensics is one part of digital forensic science which is focusing artifacts found in Windows operating system. This section is about various Windows artifacts relevant to the experiments and analysis.

The Windows **registry** is a collection of database files, called hives, that store configuration information for the system. Standard hives are [19]:

- **NTUSER.DAT** hive contains the configuration and environment settings for each user who has used the machine. This hive is located at User's folder C:\Users\<username>\NTUSER.DAT. The hive can be used to enumerate most recently opened files, which files was recently searched for hard drive, last URLs typed to browser window, last commands executed.

- **UsrClass.dat** contains additional program execution information and provides the ability to see which folders a user has opened or closed. The hive is located at C:\Users\<username>\AppData\Local\Microsft\Windows\UsrClass.dat

- **SAM** hive is used for user profiling, the hive contains user login information, group information, and user's RID (Relative Identifier). It is located at C:\Windows\System32\config\SAM

- **SOFTWARE** hive contains information about Microsoft Windows version, OS version, install dates, and more. It is located at C:\Windows\System32\config\SOFTWARE

- **SYSTEM** hive identifies system's configuration settings, control set, which is important to control system boot including driver and service information, also contains information about computer's name, time zone, network interfaces and types, autoruns, and more. The hive is located at C:\Windows\System32\config\SYSTEM

Amcache.hve hive is for Windows to run older executables found from older iterations in current version of Windows. For Forensic perspective it can be used to track program executions. In this hive Last Modification Time means First Run Time. The hive is located at C:\Windows\AppCompat\Programs\Amcache.hve

Another registry related artifact is **Application Compatibility: ShimCache** is used when a program opens, its purpose is to detect and remediate program compatibility challenges caused by that a program was built to work on older version of Windows. AppCompatCache tracks the executable file's last modification time, file path. Applications will be "shimmed" again if the file contents are updated or renamed, so it can be used for detecting that a program was moved, renamed, or timestamps altered. Located at SYSTEM hive:

SYSTEM\CurrentControlSet\Control\SessionManager\AppCompatCache\AppCompatCache

BAM, Background Activity Moderator record the path of the executable and last execution time. SYSTEM\CurrentControlSet\Services\bam\UserSettings\{SID}

One Windows artifact not related to registry forensics is the **Prefetch.** It is a process that makes Windows operating system faster by loading key pieces of data and code from disk into memory before it is needed. The forensic value of prefetch files is that each prefetch filename is a combination of the executable file name, followed by a dash and hexadecimal representation of a hash of the file's path. Prefetch files indicate program execution, embedded within total number (up to 8) of times a program has been executed, original path of execution, and the last time of execution. The execution times may differ from real execution time up to additional 10 seconds. Location of prefetch files is C:\Windows\Prefetch this folder holds up to 1024 entries.

The System Resource Usage Monitor (SRUM) is part of Windows diagnostics and tracks different system performance elements. SRUM records how the programs run and user ID responsible for launching the application [20]. SRUM analysis can be extra beneficial in cases of counter-forensic programs are used or a user transferring mass amount of data from the network to external places [20]. Performance data is initially collected in the SOFTWARE hive and written to the SRUDB.dat approximately every 60 minutes of system runtime or during proper system shutdown, the database which is located on C:\Windows\System32\SRU [20].

Windows artifacts could also be found in **Event Logs.** Windows Event Logs provide a standard, centralized way for the operating system and associated applications to record important software and hardware information [20]. Event logs provide historical information that can help illuminate system and security problems as well as tracking user actions and system resource usage [20]. What and how much is recorder to the logs is dependent on the applications involved and the system settings. Event logs are in .evtx format and default location is %systemroot%\System32\winevt\logs

Sysmon, short for System Monitor is for monitor and log system activity to the Windows event log [21]. In default configuration it provides information about process creation and termination, after configuration it is capable of logging various malicious or anomalous activities in system or network. Events are stored in Applications and Services Logs/Microsoft/Windows/Sysmon/Operational [21].

Journaling allows to peer back in time to find moment-by-moment changes to files and folders on the volume [22]. The two files that make up the journaling features of NTFS are the $LogFile and $UsnJrnl, the purpose of the first one is to provide low-level transactional data about the changes to the file system, which also provides resiliency to NTFS, the $UsnJrnl logs higher-level actions that can be used by applications to monitor for file and directory changes [22]. Each USN record tracks a change file’s or folder’s name, MFT number, its parent directory’s MFT number, a timestamp of the change, a reason code. The file size, and its attributes like is it hidden or read-only, etc. [22].

2.6 Network forensics

Network forensics consist of the monitoring, collecting, and analyzing the network traffic for the purpose of gathering legal evidence, and performing intrusion detection and response [23]. In contrast to host-based forensics, nearly all network artifacts considered to be volatile [23]. Network forensics provides overview of the events in network level, which helps to fill in the gaps that regular host based digital forensics may face.

In this work all packets were captured directly from the host with Wireshark to support the investigation of network-based operations.

2.7 Memory forensics

RAM is the bridge among the CPU, operating system, and getting thigs done [24]. Nearly everything that has happened on a modern computer has traversed RAM, so analyzing volatile data can give irreplaceable information.

In case of virtual machine memory acquisition it is recommended to suspend the virtual machine, and force a copy of a memory to be copied to the host system, or when this technique is not possible, the fallback plan to consider is to run a memory acquisition tool within the virtual guest [24], but WSL is not a traditional VM so the acquisition method should be tested as well. Unfortunately, at the time of writing this thesis Volatility did not support Windows 11 profile, which means the tool cannot be used in case of Windows 11. Analyzing memory manually is out of scope of this thesis. Investigating memory should be consider as future work.

3 Related work

The articles in this section are describing the architecture of WSL versions and pointing out the shortcomings in Windows Subsystem for Linux version 1 and 2. There is one white paper and one conference presentation that cover some aspects of endpoint forensic analysis related to WSL. All the research mentioned in this chapter are based on Windows 10.

Journal article "Memory forensics and the Windows Subsystem for Linux" points out the problems in memory forensics tools in case of Pico process and WSL in Windows 10 [5]. The problem about memory forensic frameworks was that these are designed to support one operating system type per analysis task [5]. The authors also pointed out that WSL subsystem internals are undocumented. The authors analyzed Volatility framework capabilities and developed a new plugin, for that they used reverse engineering technique of the WSL kernel and userland parts [5]. Previous research is based on WSL1 and focused on two versions (1703 and Fall Creators Update) of Windows.

As stated by P. Kochberger, A. Tauber, and S. Schrittwieser "the identification of the execution environment plays an important role in the areas of software protection and malware analysis." [25]. In this conference paper [25] the authors describe a list of vectors for an application to identify if it runs inside of Windows Subsystem for Linux, on native Ubuntu, or inside a VM, they built and implemented a prototype application for the detection. Depending on how they described WSL working principle, we can say that they wrote about WSL version 1. The paper contains the empirical list of deviations how an executable detects is it executed in WSL or in Ubuntu. The identification mechanisms were divided into three different categories, environmental artifacts, additional features, and translation discrepancies [25].

In 2021, a SANS Institute researcher Amanda Dreager published a white paper named "Looking for Linux: WSL Key Evidence", which focus on WSL1 in Windows 10 [26]. In this paper she pointed out that there is a lack of research and lack of documentation, and Windows versions have significant differences in WSL. Her research method includes two virtual machines, one with WSL installed, and one whiteout WSL to compare the behavior. The author did not use Windows default audit and logging configuration, in case of PowerShell logging, the module logging was set to log all modules, Sysmon logging, and object access auditing was enabled [26]. The paper focused more on Event log evidence not to different artifacts that can be found from the Windows operating system.

In 2017, Checkpoint researchers found a method that allows any known malware to bypass most common solutions in WSL1 and named it Bashware [27]. The method is divided into four parts, loading WSL components, enabling developer mode, installing Linux, and installing Wine [27]. The researchers said that the problem is not in WSL, but security vendors are falling behind how to monitor processes in hybrid systems.

F-secure researcher Connor Morley published a whitepaper how WSL2 can be used to bypass security mechanisms [28]. WSL2 can be easily deployed and configured with minimal input from a user, and when the instance is installed, an attacker can have full access to the WSL without the user or monitoring systems being aware of the actions [28]. The paper includes requirements to accomplish a payload weaponization, and an actual PowerShell script he used in his testing.

Another flaw in WSL2 allows network traffic to bypass the Windows firewall, the leak is possible even if the “Always require VPN” is enabled [29]. The flaw is in Hyper-V Virtual Ethernet Adapter, it is built to pass the traffic to and from guest without host’s firewall inspection, and this applies to all sandboxes that are using Hyper-V for networking [29]. WSL1 is not affected with this, because it is not using Hyper-V and network traffic is filtered by Windows Firewall [30]. Good news is that WSL2 supports Linux firewall implementations like iptables [30], but it expects that user configures it manually.

At OSDFCon 2020 conference, Asif Matadar from TANIUM cave a presentation “Investigating WSL Endpoints”, it focused on WSL2’s attack and its evidence [31]. He highlighted 11 techniques how to attack WSL2, including persistence, execution, lateral movement, command and control and execution attacks [31]. Every method included some ways to detect the attack, but only some possible forensic artifacts were covered.

4 Methodology

This section describes the setup of the experiment, the disk acquisition process, and the tools used in analysis. Well known and commonly used free tools were chosen for data acquisition and subsequent analysis. Some of these tools are recommended and used by SANS Institute instructors during forensic courses (FOR500 and FOR508). A commercial tool called Magnet AXIOM was also used for data analysis.

4.1 Tools

Caine or Computer Aided Investigative Environment virtual machine is customized Ubuntu 18.04 with graphical interface for computer forensics, with many forensic software tools installed [32]. An important aspect of Caine is that all the drives are mounted as read-only unless investigator change that. “The new write-blocking method assures all disks are really preserved from accidentally writing operations” [32].

FTK Imager is a tool for previewing and imaging electronic evidence [33]. In this work FTK Imager command line version was used to image the test computer’s disk.

Arsenal Image Mounter is an open-source forensically sound tool for mounting the contents of disk images as complete Windows disk [34]. Can be used for mounting RAW/DD, E01, and Virtual Machine Disk files [34]. The tool allows the reviewer to interact with files with their native or associated application, copy files out of the mounted filesystem.

Registry Explorer, written by Eric Zimmerman, is free open-source GUI-based tool used to the view contents of offline registry hives. It has ability to load multiple hives at once, search across all loaded hives using string or regular expressions.

KAPE, or Kroll Artifact Parser and Extractor, is a triage software written by Eric Zimmerman. KAPE is a multi-function program that allows to collect files and process collected files with one or more programs [35]. KAPE has two parts, one is Targets which is for collecting files, folders, and various types of items on a storage device, second is Modules, which is used for processing files and folders [35], KAPE’s working principle is shown in Figure 3. It is possible to run GUI application, which also displays command line arguments, that command can be saved and used for even faster collection for multiple devices.

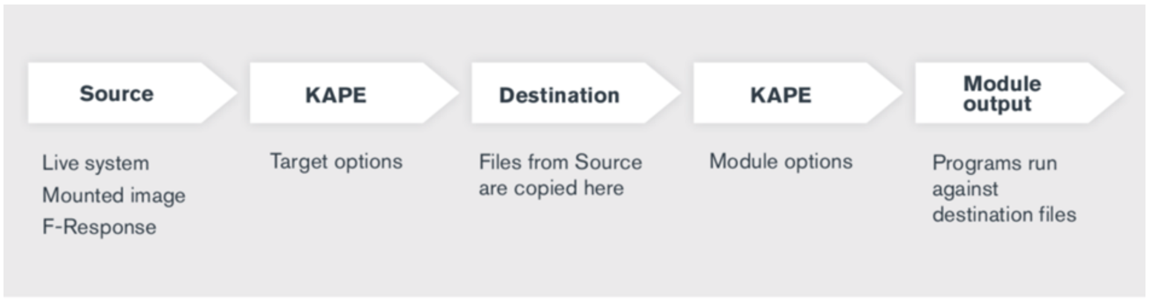


Figure 3. How KAPE works [14]

Timeline Explorer is a free tool replacing MS Excel. It's designed for digital forensic examination, written by Eric Zimmerman. It supports conditional coloring, filtering, and grouping, and allows to open multiple files simultaneously.

MFTCmd.exe is another tool by Eric Zimmerman, it is a fast command line parser for MFT.

LogFileParser.exe is a free tool from Joakim Schmicht to analyze $LogFile. The primary output is designed to give an overview of the data contained in the $LogFile [22].

PECmd.exe is prefetch command line parser, written by Eric Zimmerman. PECmd will output two files, one file will contain the embedded information such as run count, last run times, and files referenced for each prefetch file int folder. The second file is a timeline view of the output, it includes a row for each embedded timestamp.

AppCompatCacheParser.exe can be used to parse SYSTEM hive for Application Compatibility information to examine file executions, it is written by Eric Zimmerman [20].

Event Log Explorer is free for personal use event log management software package. It supports logs from every Windows NT operating system and can read both .evt and .evtx log formats. It is capable of working with corrupted log files, allows to open many log files simultaneously, and has robust filtering capability, including access to the text-based

Description field. Quick Filters allow options like showing only one events of a specific type or removing types of events from view [20].

Srum-dump.exe is written by Mark Baggett and it is a free srum-dump tool, for automatically decode all the files in the SRUM database [20].

NirSoft **NetworkUsageView** tool allows an investigator to analyze network usage information stored in the SRUDB.dat database. It is a graphical application that includes timestamps, App Name, App Description, User, User SID, bytes sent and received [36].

Magnet Axiom is powerful commercial tool for recover, process, and analyze digital evidence in one case [37].

Wireshark is free open-source multi-platform network packet analyzer. It has many features, including live capture and offline analysis, deep inspection of protocols, many export options, and more [38].

4.2 Experimental setup

Information about test PC can be found in Table 1. Sysmon was installed and running to provide some extra logging during the experiment, it had default configuration, which means only few events (process creation and termination) were logged.

Table 1. Test PC parameters

Computer	Lenovo T490
Processor	Intel® Core™ i7-8665U CPU @ 1.90GHz 2.11 GHz
Installed RAM	24 GB (23.7 GB usable)
System type	64-bit operating system, x64-based processor
Operating system	Windows 11 Pro
Version	21H2
OS built	22000.556

The experiments starts with Windows Subsystm for Linux version 2 because it is now the default and recommended version. The experiment was designed with forensic examiner’s work in mind. Investigators are looking for evidence of what users have done with files, what they did on network, and which applications they used.

The experiment can be divided into six parts. First, subsystem installation, the purpose of this part is to produce evidence of WSL installation. First WSL2 with default distribution, Ubuntu, was installed on Windows 11. To finish the installation process a reboot was required. After the reboot WSL2 was ready to use. The test computer was shut down and first disk image was collected, disk image acquisition process is described in chapter 4.3.

The second part was about file operations. Users are likely to engage with files so file operations like file creation, modification and deltation are essential part of understaning the subsystem specificity. Files created in WSL can be saved to Linux file system or directly to Windows’. This part of the experimention was focusing on producing the senario where files are created and modified in WSL and saved to Linux’ and Windows file system to analyse how the files differ and which tracks can be detected by using the computer forensic techniques. The experiment started with creating the file “test.txt”, it was created in WSL but saved to Windows file system. Next, the file’s writing permissions were removed using the command “chmod”. Second file “test2.txt” was created and removed instantly. First file “text.txt” was moved to Ubuntu’s file system. Then a third file “newtest.txt” was created and saved on Ubuntu’s file system, after it was moved to Windows Documents folder to test if there are differences between moving files from one file system to another and does the direction affect the outcome.

Third step was about ICMP echo requests. The ping command was used to understand the connectivity between the host and the subsystem. Wireshark capture was started to collect network traffic for analysis. Network capture helps to understand how WSL looks from network perspective and is it possible to distinguish WSL connections from host. Command “ip a” in WSL2 shows the IP 172.21.56.123/20. On Windows “ipconfig” commands shows host IP 192.168.1.132 and Ethernet adapter vEthernet (WSL) IP 172.21.48.1, which can be handled as a gateway between host and WSL2. It was immediately seen that pinging host from WSL2 was unsuccessful, but echoing WSL2 from host got replies, third ping was from WSL2 to Google 8.8.8.8, and it was successful.

Fourth part was about various network connections a potential user may perform from a command line. Curl command was used to download a file from TalTech library to Windows file system and to display Tallinn's weather info in terminal. Second transfer utility that was used was wget, different utilities were used to see if there are any differences. Wget command was used to download Eicar test files to Windows and Ubuntu. It was interesting that on Windows the malware file disappeared but remained in place in Ubuntu. A telnet connection was established to see world map in terminal, the maps was interactive which allowed to zoom into different places.

A user usually needs to use some applications; therefore, next part was focusing on actions related to programs. The test started whit executing a program that resides on Windows, the first executed application was Sysmon, second one was Notepad which do not need elevated rights like Sysmon. Opened Notepad was used to save new file named "123.txt". The test user also upgraded the WSL to be able to use latest features and provide extra evidence for examination.

The last part of the experiment was about a graphical user interface which is new feature introduced in Windows 11. First GUI program installed with apt was Linux text editor Gedit. Second applications that was installed and opened was the Firefox browser. In 2020, September 30, a firewall bypass flaw was published as described in related work chapter 3, to test if it is still possible a new rule to block 80, 8080, and 443 was enabled in Windows firewall. The Firefox browser was opened in Ubuntu and Internet browsing was possible without any restrictions, meanwhile Edge browser opened in Windows couldn't access the same sites. Two browsers can be found in Appendix 2.

The following Figure 4 shows the steps of the WSL2 testing.

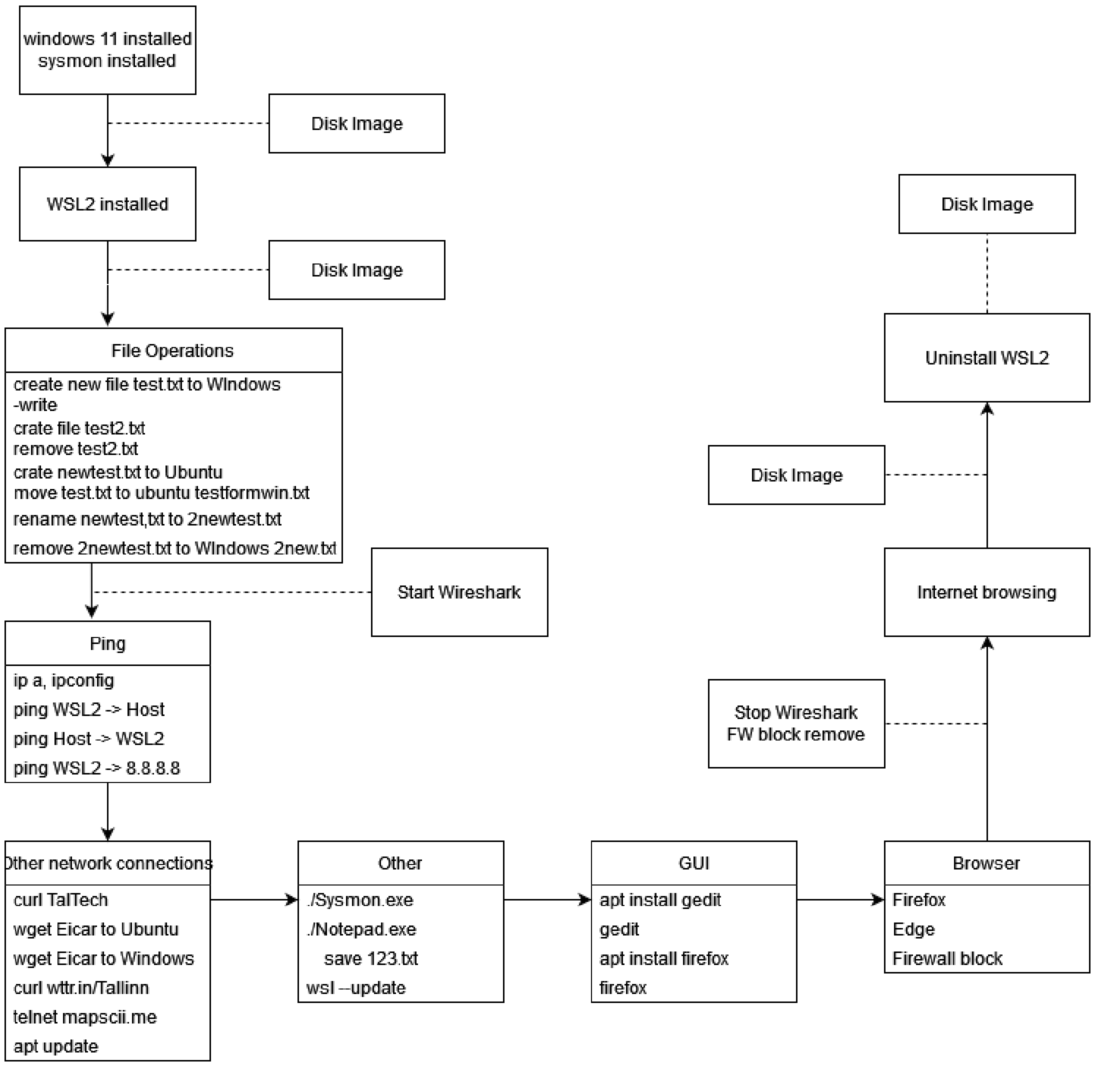


Figure 4. WSL2 experiment flow

The computer operating system was reinstalled after experiments on WSL2. The experiment to test WSL1 was similar to version 2 testing. The most significant difference comes from GUI part, because WSL1 do not support graphical interface. The WSL1 experiment flow is presented in Figure 5.

As described earlier WSL1 and WSL2 have different architecture and therefore the logic of storing the files is also different, for forensic perspective it is crucial to know where the files reside, and which evidence are left behind regarding the subsystem version. The experiment was based on same tests as WSL2 to be able to compare the results. The file operations part started with creating a new file named “testwsl1.txt”. After the creation the file has full rights, then the writing permission were removed with “chmod” command. Second file “test2wsl1.txt” was created and deleted immediately. Third file “newtestwsl1.txt” was created to Ubuntu’s file system and moved to Windows file system with a name “2newwsl1.txt”.

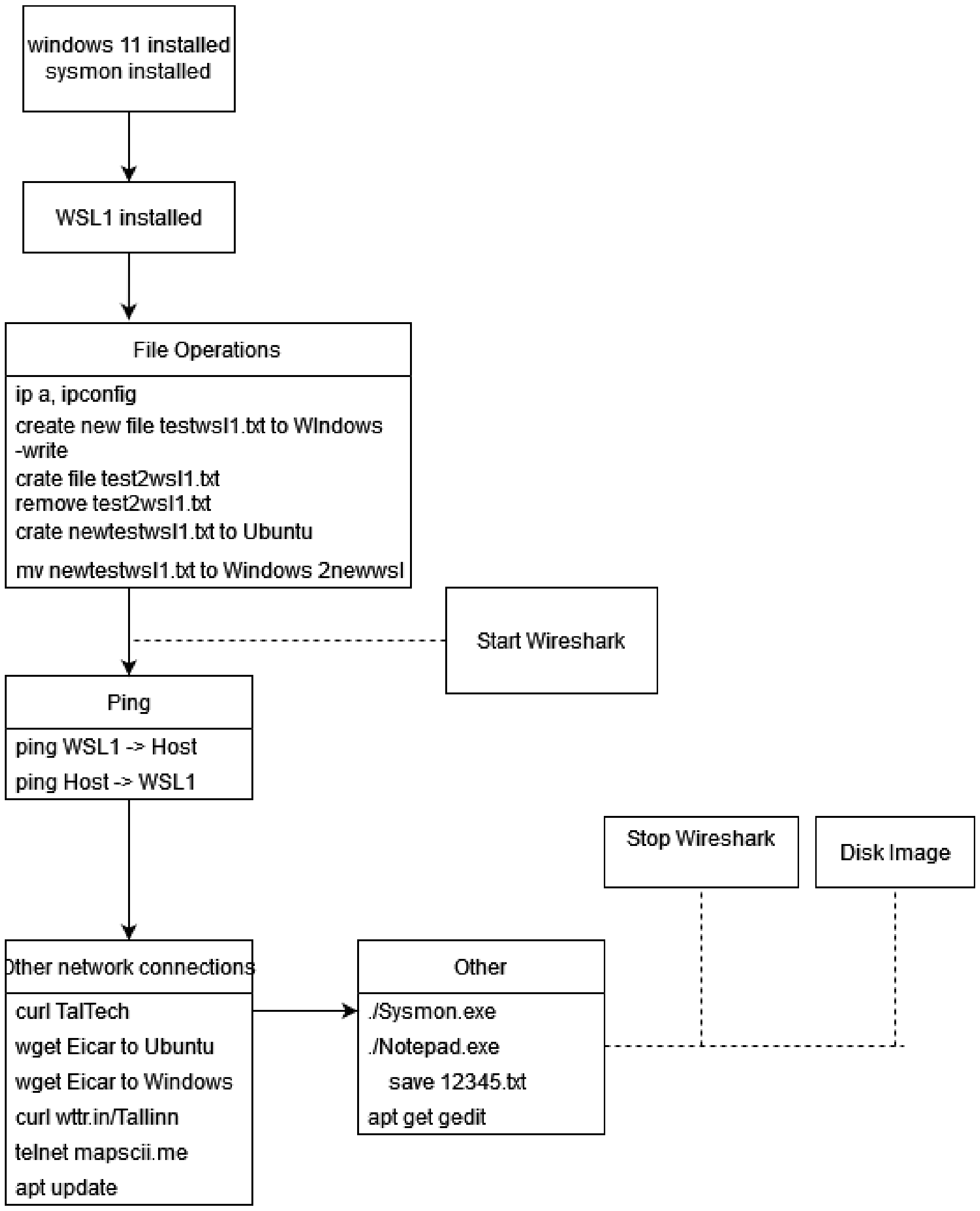


Figure 5. WSL1 experiment

Like previously the next section was about ICMP echo request to understand WSL1 from network perspective. Host IP was 192.168.1.132, WSL IP info was interesting because it showed three different ethernet interfaces and three wifi interfaces, one of those was the same as the host, WSL1 IP info can be found in Appendix 3. Ping request from Ubuntu to host was successful. Ping request from host to two WSL’s eth interfaces (169.254.75.17, 169.254.72.126) was unsuccessful.

Following test were same as WSL2, a paper was downloaded from TalTech library, Eicar file was downloaded to Windows and Ubuntu, weather info was requested with curl and telnet connection was established to see command line map. After the WSL was updated with apt. Sysmon and Notepad applications were executed from Ubuntu. Gedit and Firefox execution did not work because WSL1 do not support graphical interface applications and displayed error message “no DISPLAY environment specified”.

The Wireshark capture was stopped, and disk image was acquired according to the procedure described in next section.

4.3 Disk image acquisition process

After the testing the computer was shut down, and the NVMe SSD was removed from the computer. The disk was inserted to another computer. The analysis machine had USB with bootable Caine VM connected. After the computer booted from USB, the computer’s hard drive was mounted as read-only but changed to writable disk, the disk under investigation remained read-only mode. Command line version of FTK Imager was used to carve the disk image in E01 format, verification flag was set, to be sure that everything went right. Fragmentation was set to 2T so the disk will remain in one piece.

5 Results

In this section the acquired disk images were investigated. The purpose of the analysis is to detect as many artifacts locations as possible. It is important to have alternatives for evidence locations to look for in cases the data is deleted or wiped from system by a user or anti forensic measures were used.

5.1 Evidence of WSL2 installation and usage

This section presents the forensic evidence that WSL2 left after installation was completed. The analysis began with Windows registry forensics, followed by AppCompatCache and Event Logs.

5.1.1 Registry

Windows registry hives contain many values, but some of them are not human readable, are defective, or just do not add value to an investigation. For example, many date or time entries are not useful, some contain only zeros, others are overwritten every time the registry value is used. Information of WSL2 installation can be found on SOFTWARE, Amcache.hve and UsrClass.dat registry hives. Registry hives, locations, and values that can be beneficial to forensic examiners are presented in Table 2. The table can help the examiner to find information of interest precisely essential to the current case.

Table 2. Indications of WSL2 installation

Registry hive	**Values**
Amcache.hve: Root\InventoryApplicationFile\wsl.exe|88d5e8ee009f35b9	File Name, Path (..\System32\), Publisher, Version, Binary Type, Size
SOFTWARE: Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products\E752FE635D127F4448157029A3C864E5\Install Properties	Local Package (.msi), Display Version, Help Link, Install Source (..\Temp\), Publisher
SOFTWARE: Microsoft\Windows\CurrentVersion\Installer\Folder	Path (GUI location on C:\ ProgramData\Microsft\wsl)

Registry hive	**Values**
Amcache.hve: Root\InventoryApplication\00007c914b0dd2764fd645ed6e2c9620551900000904	GUI info: Program IDs, Name, Version, Publisher, Install Date
Amcache.hve: Root\InventoryApplicationFile\wslhost.exe|e4351ae5a193337	Program IDs, Path (..\System32\lxss\), Name, Publisher, Version, Binary Type, Size
UsrClass.dat: Local Settings\Software\Microsoft\Windows\Shell\MuiCache	Friendly name, Application Company

It is important to follow the IDs very precisely, because in different registry keys they may represent different thing, for example program IDs in Amcache.hve folders are not the same.

In some cases, distribution information can be important to a case, fortunately it can be found in different places. Information about installed distribution can be found in SYSTEM, SOFTWARE, NETUSER.DAT and UsrClass.dat hives. Distribution info is present after computer is rebooted. The Table 3 presents registry hives with interesting values in them.

Table 3. Distribution information

Registry hive	**Values**
SOFTWARE: Microsoft\Windows\CurrentVersion\AppModel\StagingInfo\CanonicalGroupLimited.UbuntuonWindows_2004.2020.424.0_x64__79rhkp1fndgsc	DownloadSize
SOFTWARE: Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\PackageRepository\Packages\CanonicalGroupLimited.UbuntuonWindows_2004.2020.424.0_x64__79rhkp1fndgsc	Path
UsrClass.dat: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppContainer\Mappings\S-1-15-2-202137915-1588483389-988967374-857029487-3114683470-1999124116-284053513	Display Name, Moniker

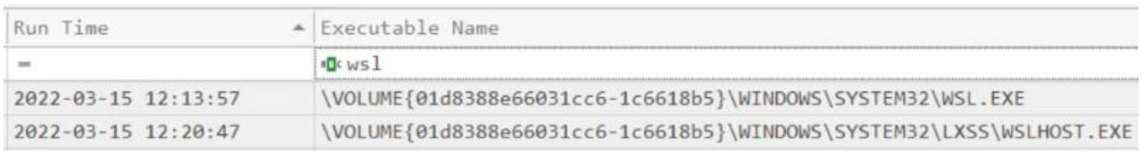
Registry hive	**Values**
UsrClass.dat: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Packages\CanonicalGroupLimited.UbuntuonWindows_2004.2020.424.0_x64__79rhkp1fndgsc	Package Root Folder, Display Name, Package ID
NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\App Paths\ubuntu.exe	Path
NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\Lxss\{17f1eb54-04fa-48ab-98d6-a654d1f70f47}	State, Distribution Name, Base Path, Package Family Name
SYSTEM: ControlSet001\Services\bam\State\UserSettings\S-1-5-21-2705457751-833678465-653486118-1001	Path, Data
SYSTEM: ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\RestrictedServices\AppIso\FirewallRules	Firewall rule

The location of WSL2 files is AppData folder, Linux file system files are on a Hyper-V virtual hard disk (vhdx):

The Path:
C:\Users\Thesis\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_79rhkp1fndgsc\LocalState\ext4.vhdx

5.1.2 Prefetch

To parse prefetch files a command line tool PECmd.exe was used and the relevant output is shown in Figure 6. From Prefetch we can see that WSL.exe was first run at 12:13:57 which is the same time like in Amcache.hve registry hive, and WSLHOST.EXE entry appeared first time after computer was rebooted and few seconds after Ubuntu.exe was executed.



Run Time	Executable Name
–	wsl
2022-03-15 12:13:57	\VOLUME{01d8388e66031cc6-1c6618b5}\WINDOWS\SYSTEM32\WSL.EXE
2022-03-15 12:20:47	\VOLUME{01d8388e66031cc6-1c6618b5}\WINDOWS\SYSTEM32\LXSS\WSLHOST.EXE

Figure 6. Prefetch wsl.exe and wslhost.exe

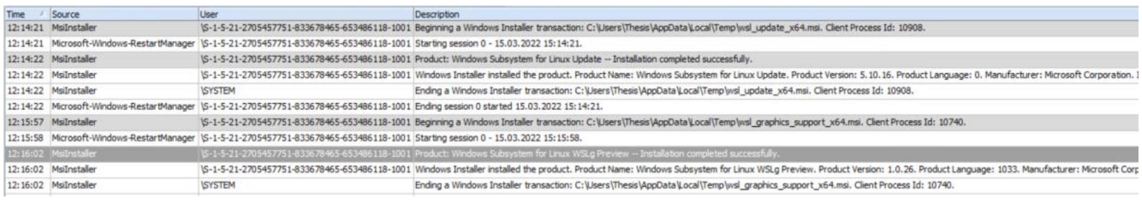
5.1.3 AppCompatCache

AppCompatCacheParser.exe output shows that first time wslhost.exe was “shimmed”, was right after WSL installation began. So AppCompatCache has earlier evidence than Prefetch that WSL exists in system.

5.1.4 Event logs

The test computer had Sysmon installed. Sysmon event 1 expresses Process creation, and event code 5 Process termination. The first entry of WSL process creation was few milliseconds earlier than in Amcache.hve, the event description included cmd.exe command line “wsl –install”, SHA256, ID, and folders, some events show command line “wsl –install -d ubuntu” although the distribution information was not typed manually, the Sysmon event is shown in Appendix 4. Same command execution is logged in *Microsoft-Windows-Shell-Core%4Operational* with event id-s 9707 (command execution started) and 9708 (command execution finished), but this log entry was written after the computer was rebooted. The *Microsoft-Windows-VHDMP-Operational* log file is a source for virtual disk information, first entry has a user SID info, and the description field contains “Handle for the file backing virtual disk 'C:\ProgramData\Microsoft\WSL\system.vhd' created successfully. “, the entry it is present after the computer was rebooted.

Application logs provide an overview of WSL installation, it includes related user SID, and informative descriptions. User SID is important information, it allows actions to be associated with the user. AS shown on Figure 7, graphical user interface is included in default WSL installation package. Information about distribution installation process, including time spent on installation and associated user SID, can be found in *AppXDeploymentServer%4Operational* log file.

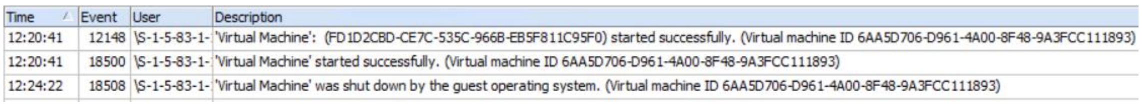


Time	Source	User	Description
12:14:21	MsiInstaller	\S-1-5-21-2705457751-833678465-653486118-1001	Beginning a Windows Installer transaction: C:\Users\Thesis\AppData\Local\Temp\wsl_update_x64.msi. Client Process Id: 10908.
12:14:21	Microsoft-Windows-RestartManager	\S-1-5-21-2705457751-833678465-653486118-1001	Starting session 0 - 15.03.2022 15:14:21.
12:14:22	MsiInstaller	\S-1-5-21-2705457751-833678465-653486118-1001	Product: Windows Subsystem for Linux Update -- Installation completed successfully.
12:14:22	MsiInstaller	\S-1-5-21-2705457751-833678465-653486118-1001	Windows Installer installed the product. Product Name: Windows Subsystem for Linux Update. Product Version: 5.10.16. Product Language: 0. Manufacturer: Microsoft Corporation. I
12:14:22	MsiInstaller	\SYSTEM	Ending a Windows Installer transaction: C:\Users\Thesis\AppData\Local\Temp\wsl_update_x64.msi. Client Process Id: 10908.
12:14:22	Microsoft-Windows-RestartManager	\S-1-5-21-2705457751-833678465-653486118-1001	Ending session 0 started 15.03.2022 15:14:21.
12:15:57	MsiInstaller	\S-1-5-21-2705457751-833678465-653486118-1001	Beginning a Windows Installer transaction: C:\Users\Thesis\AppData\Local\Temp\wsl_graphics_support_x64.msi. Client Process Id: 10740.
12:15:58	Microsoft-Windows-RestartManager	\S-1-5-21-2705457751-833678465-653486118-1001	Starting session 0 - 15.03.2022 15:15:58.
12:16:02	MsiInstaller	\S-1-5-21-2705457751-833678465-653486118-1001	Product: Windows Subsystem for Linux WSLg Preview -- Installation completed successfully.
12:16:02	MsiInstaller	\S-1-5-21-2705457751-833678465-653486118-1001	Windows Installer installed the product. Product Name: Windows Subsystem for Linux WSLg Preview. Product Version: 1.0.26. Product Language: 1033. Manufacturer: Microsoft Corp
12:16:02	MsiInstaller	\SYSTEM	Ending a Windows Installer transaction: C:\Users\Thesis\AppData\Local\Temp\wsl_graphics_support_x64.msi. Client Process Id: 10740.

Figure 7. Application logs

Evidence of WSL2 installation can also be found in *Microsoft-Windows-Hyper-V-VmSwitch-Operational* log file, it contains information how WSL is connected to Hyper-V VmSwitch, including ports and network adapters, user SID is also present, but this is different than previous one.

Microsoft-Windows-Hyper-V-Worker-Admin.evtx log is shown in Figure 8, the log file is a source for following virtual machine session, every session gets unique ID and user SID, and every virtual machine associated got another unique ID. It is possible to track one VM session by ID. Events 12148 and 18500 are indicating that Virtual Machine started successfully and event id 18508 was written to logs when the VM was shut down by the guest operating system. Between start and stop there were logs about PCI device operations.



Time	Event	User	Description
12:20:41	12148	\S-1-5-83-1-	'Virtual Machine': (FD1D2CBD-CE7C-535C-966B-EB5F811C95F0) started successfully. (Virtual machine ID 6AA5D706-D961-4A00-8F48-9A3FCC111893)
12:20:41	18500	\S-1-5-83-1-	'Virtual Machine' started successfully. (Virtual machine ID 6AA5D706-D961-4A00-8F48-9A3FCC111893)
12:24:22	18508	\S-1-5-83-1-	'Virtual Machine' was shut down by the guest operating system. (Virtual machine ID 6AA5D706-D961-4A00-8F48-9A3FCC111893)

Figure 8. Microsoft-Windows-Hyper-V-Worker-Admin.evtx

Virtual machine IDs in *Hyper-V-Worker-Admin* logs are the same as in *Hyper-V-Compute-Operational logs*.

Amanda Draeger claimed in her white paper that Windows Security logs can be used for detecting process creation and object access (event id 4688 and 4656), but in this work security logs had information about audit policy change (event id 4907). This may be caused by that she enabled extra logging and this work used default settings or because Windows versions are different.

5.2 Evidence of WSL1 installation and usage

This section presents the forensic evidence that WSL1 left after installation was completed. The analysis began with Windows registry forensics, followed by AppCompatCache and Event Logs.

5.2.1 Registry

WSL1 Windows registry entries that can be useful for forensic investigation are presented in Table 4.

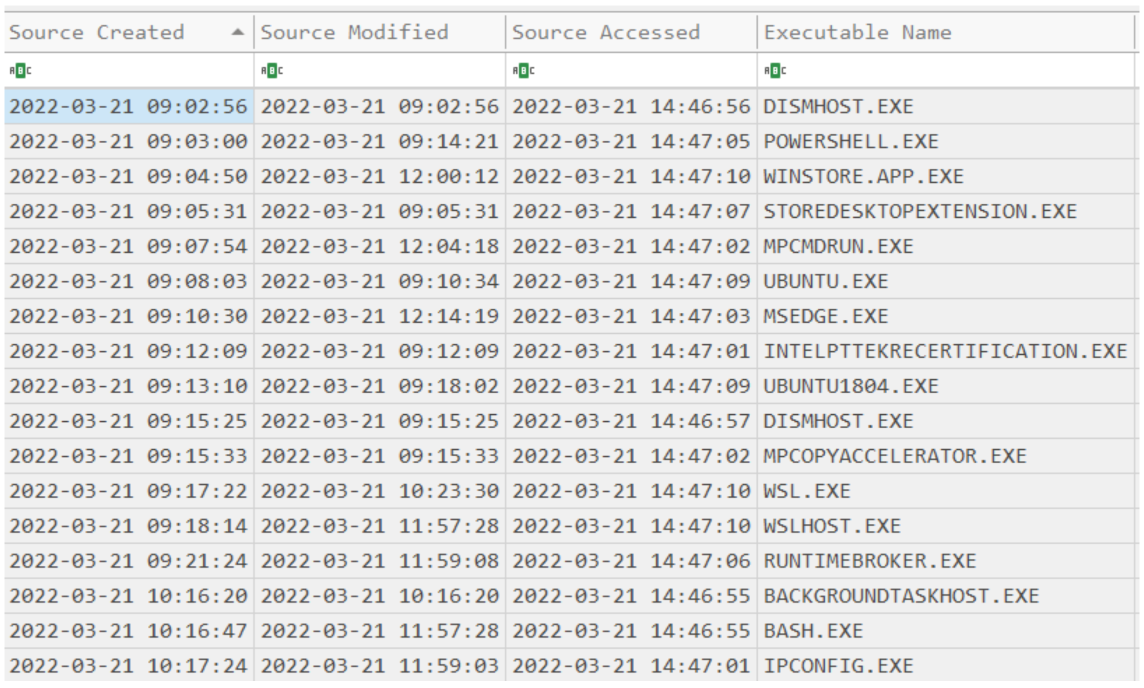
Table 4. WSL1 registry evidence

Registry hive	**Values**
SOFTWARE: Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\PackageRepository\Packages\CanonicalGroupLimited.Ubuntu_2004.4.2.0_neutral_~_79rhkp1fndgsc	Path
UsrClass.dat: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppContainer\Mappings\S-1-15-2-3731972660-2291479297-3906359699-410282953-842910066-2550138301-2406496836	DisplayName, Moniker,
UsrClass.dat: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Packages\CanonicalGroupLimited.Ubuntu_2004.4.2.0_x64__79rhkp1fndgsc	DisplayName, PackageID, PackageRootFolder, PackageSid
NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\App Paths\ubuntu.exe	Path
\SYSTEM: ControlSet001\Services\bam\State\UserSettings\S-1-5-21-1154849975-3556662000-2890839169-1001\CanonicalGroupLimited.Ubuntu_79rhkp1fndgsc	
SOFTWARE: Classes\Directory\(background)\shell\WSL	
SOFTWARE: Microsoft\Windows\CurrentVersion\Explorer\IdListAliasTranslations\WSL	source - \\wsl.localhost
SOFTWARE: Classes\CLSID\{615a13be-241d-48b1-89b0-8e1d40ffd287}	Data: WslClient C:\Windows\System32\lxss\wslclient.dll

Registry hive	**Values**
SOFTWARE: Classes\CLSID\{B2B4A4D1-2754-4140-A2EB-9A76D9D7CDC6}\Instance\InitPropertyBag	Provider, ResName,
SOFTWARE: Classes\Directory [or \Drive\]\shell\WSL\command	Data : wsl.exe --cd "%V"
SYSTEM: ControlSet001\Control\Session Manager\AppCompatCache	Wslhost.exe location
SYSTEM: ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\RestrictedServices\AppIso\FirewallRules	Firewall rule
NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\Lxss\{4c42a344-8010-42b7-8aa7-d2595c169358}	BasePath, DistributionName, PackageFamily,Name
Amcache.hve: Root\InventoryApplicationFile\ubuntu.exe|264b49043e869df6	AppxPackageFullName, BinaryType, Path, Name, ProgramId, Size, Usn
Amcache.hve: Root\InventoryApplicationFile\wsl.exe|88d5e8ee009f35b9 and wslhost.exe|e4351ae5a193337	BinaryType, BinFileVersion, Path, Name, ProductName, ProductVersion, ProgramID, Publisher, Size, Usn, Version
Amcache.hve: Root\InventoryApplication\0000b4e521368d49ea531c16c262f06c607800000908	ManifestPath, Name, ProgramID, ProgramInterfaceId, Publisher, RootDirPath, Source, StoreAppType, UesrSID, Version

5.2.2 Prefetch

Prefetch analysis shows the programs executed during the experiment nearly in correct order is shown in Figure 9. Dismhost.exe (dism.exe) was ran with powershell.exe, the reason powershell.exe creation time is later is that prefetch timestamps have up to 10 seconds delay since the actual start time, so applications executed closely can be in shift.



Source Created	Source Modified	Source Accessed	Executable Name
2022-03-21 09:02:56	2022-03-21 09:02:56	2022-03-21 14:46:56	DISMHOST.EXE
2022-03-21 09:03:00	2022-03-21 09:14:21	2022-03-21 14:47:05	POWERSHELL.EXE
2022-03-21 09:04:50	2022-03-21 12:00:12	2022-03-21 14:47:10	WINSTORE.APP.EXE
2022-03-21 09:05:31	2022-03-21 09:05:31	2022-03-21 14:47:07	STOREDESKTOPEXTENSION.EXE
2022-03-21 09:07:54	2022-03-21 12:04:18	2022-03-21 14:47:02	MPCMDRUN.EXE
2022-03-21 09:08:03	2022-03-21 09:10:34	2022-03-21 14:47:09	UBUNTU.EXE
2022-03-21 09:10:30	2022-03-21 12:14:19	2022-03-21 14:47:03	MSEDGE.EXE
2022-03-21 09:12:09	2022-03-21 09:12:09	2022-03-21 14:47:01	INTELPTTEKRECERTIFICATION.EXE
2022-03-21 09:13:10	2022-03-21 09:18:02	2022-03-21 14:47:09	UBUNTU1804.EXE
2022-03-21 09:15:25	2022-03-21 09:15:25	2022-03-21 14:46:57	DISMHOST.EXE
2022-03-21 09:15:33	2022-03-21 09:15:33	2022-03-21 14:47:02	MPCOPYACCELERATOR.EXE
2022-03-21 09:17:22	2022-03-21 10:23:30	2022-03-21 14:47:10	WSL.EXE
2022-03-21 09:18:14	2022-03-21 11:57:28	2022-03-21 14:47:10	WSLHOST.EXE
2022-03-21 09:21:24	2022-03-21 11:59:08	2022-03-21 14:47:06	RUNTIMEBROKER.EXE
2022-03-21 10:16:20	2022-03-21 10:16:20	2022-03-21 14:46:55	BACKGROUNDTASKHOST.EXE
2022-03-21 10:16:47	2022-03-21 11:57:28	2022-03-21 14:46:55	BASH.EXE
2022-03-21 10:17:24	2022-03-21 11:59:03	2022-03-21 14:47:01	IPCONFIG.EXE

Figure 9. WSL1 Prefetch

In the second PECmd.exe output file (PECmd_Output_Timeline.csv), powershell.exe source created timestamp is earlier than dism.exe's.

5.2.3 AppCompatCache

AppCompatCache information is the same in WSL2 and WSL1, AppCompatCacheParser.exe output shows that first time wslhost.exe was "shimmed", was right after WSL installation began. So AppCompatCache has earlier evidence than Prefetch that WSL exists in system.

5.2.4 Event logs

Sysmon Operational logs provided process creation information and SHA256 hash for dism.exe, dishost.exe, ubuntu.exe, and wsl.exe. Sysmon also logged the command "wsl.exe –set-default-version 1" which was ran to set default version from WSL2 to WSL1. From Sysmon entry we can see which distribution was executed and where are the files stored in Windows file system. The Sysmon log entry is shown in Appendix 4.

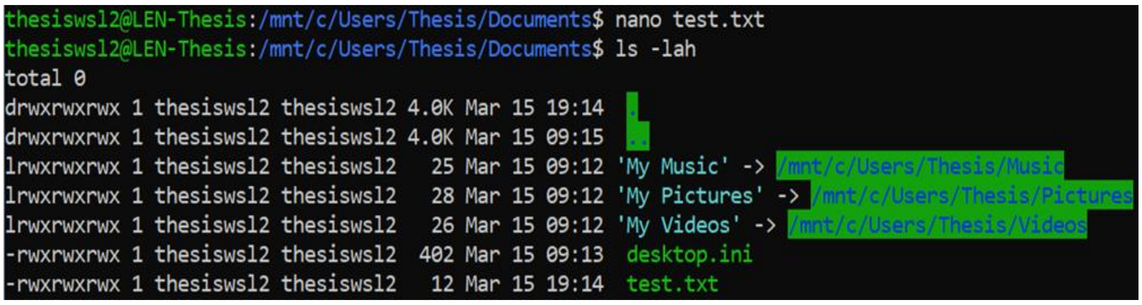
In *Microsoft-Windows-AppXDeployment-Server* log file we can find information about WSL deployment, including paths, user SID, time, and packages. First deployment had user SID S-1-5-18, which is referring to System, later the process got unique user SID with domain identifier. The *Microsoft-Windows-AppModel-Runtime* has user SID, and AppContainer information. The *System* event log contains information about Ubuntu's update start and finish. Microsoft *Windows Store%4Operational* log file has detailed information about actions related to Microsoft Store and Install Agent.

5.3 File operations

This section covers the analysis of the file operations described in experimental setup section.

5.3.1 File operations in WSL2

The Figure 10 shows the creation of the first file named "test.txt", the file got full permissions (777) in Windows file system. The file was moved to Linux file system and it inherited permission set from Windows.

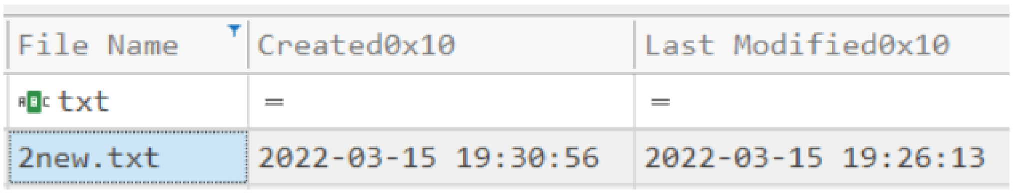


```
thesiswsl2@LEN-Thesis:/mnt/c/Users/Thesis/Documents$ nano test.txt
thesiswsl2@LEN-Thesis:/mnt/c/Users/Thesis/Documents$ ls -lah
total 0
drwxrwxrwx 1 thesiswsl2 thesiswsl2 4.0K Mar 15 19:14  .
drwxrwxrwx 1 thesiswsl2 thesiswsl2 4.0K Mar 15 09:15  ..
lrwxrwxrwx 1 thesiswsl2 thesiswsl2   25 Mar 15 09:12 'My Music' -> /mnt/c/Users/Thesis/Music
lrwxrwxrwx 1 thesiswsl2 thesiswsl2   28 Mar 15 09:12 'My Pictures' -> /mnt/c/Users/Thesis/Pictures
lrwxrwxrwx 1 thesiswsl2 thesiswsl2   26 Mar 15 09:12 'My Videos' -> /mnt/c/Users/Thesis/Videos
-rwxrwxrwx 1 thesiswsl2 thesiswsl2  402 Mar 15 09:13  desktop.ini
-rwxrwxrwx 1 thesiswsl2 thesiswsl2   12 Mar 15 19:14  test.txt
```

Figure 10. New file "test.txt"

The file named "newtest.txt" was saved to Ubuntu file system and by default got 644 permissions (owner can read and write, group and others can only read), but after moving the file to Windows Documents folder with new name "2new.txt", the file did not inherit permissions from Ubuntu, instead it got full permissions.

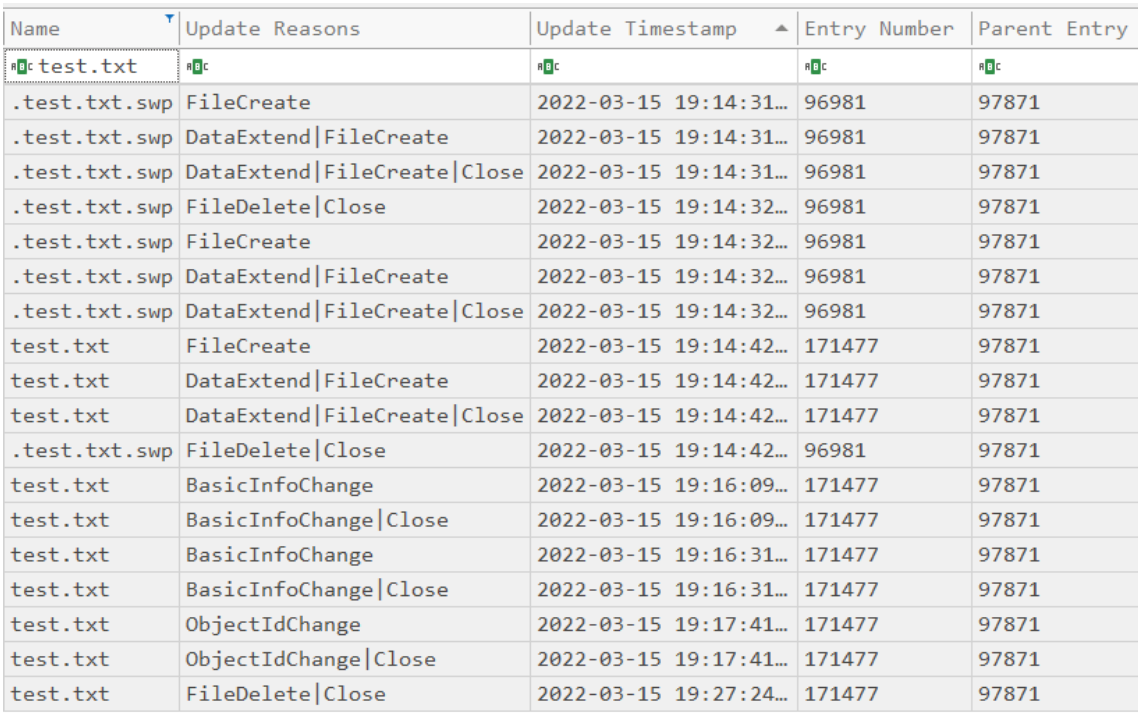
To analyze MFT a KAPE triage image with targets $MFT, $J and $LogFile was created and MFTCmd.exe was used to parse the KAPE VHDX image. The output file shows no evidence of files "test.txt" and "test2.txt", first file was removed to Ubuntu and second was just removed. File "2new.txt" created in Ubuntu and moved to Windows is present in $MFT. As shown in Figure 11 the file's creation time is later than modification time, that indicates that the file is created somewhere else and creation time shows the time created in current file system.



File Name	Created0x10	Last Modified0x10
txt	=	=
2new.txt	2022-03-15 19:30:56	2022-03-15 19:26:13

Figure 11. $MFT

MFTECmd.exe was used to parse Journal file “$J”. UsnJrnl file has detailed information about the file “test.txt” shown on Figure 12, it was created at 19:14 and file permissions were changed at 19:16, so journaling technique can be used to track modifications made from WSL to a Windows file. The $MFT output files shows that file was deleted at 19:25 but the files moved to Ubuntu, for Windows it equals to deletion. Swp file extension is an indicator that file is created in Linux OS, “SWP files are created immediately when a Vi text editing session is started. They are saved to the same directory as their original file. If a Vi session terminates due to a program kill or crash, the SWP file remains. [39]”.. Searching for swp files we can detect screen-oriented text editors for Unix operating system. Another interesting column in journal output is parent entry number. Filtering this entry number, we can see all file actions related to WSL.



Name	Update Reasons	Update Timestamp	Entry Number	Parent Entry
test.txt				
.test.txt.swp	FileCreate	2022-03-15 19:14:31…	96981	97871
.test.txt.swp	DataExtend|FileCreate	2022-03-15 19:14:31…	96981	97871
.test.txt.swp	DataExtend|FileCreate|Close	2022-03-15 19:14:31…	96981	97871
.test.txt.swp	FileDelete|Close	2022-03-15 19:14:32…	96981	97871
.test.txt.swp	FileCreate	2022-03-15 19:14:32…	96981	97871
.test.txt.swp	DataExtend|FileCreate	2022-03-15 19:14:32…	96981	97871
.test.txt.swp	DataExtend|FileCreate|Close	2022-03-15 19:14:32…	96981	97871
test.txt	FileCreate	2022-03-15 19:14:42…	171477	97871
test.txt	DataExtend|FileCreate	2022-03-15 19:14:42…	171477	97871
test.txt	DataExtend|FileCreate|Close	2022-03-15 19:14:42…	171477	97871
.test.txt.swp	FileDelete|Close	2022-03-15 19:14:42…	96981	97871
test.txt	BasicInfoChange	2022-03-15 19:16:09…	171477	97871
test.txt	BasicInfoChange|Close	2022-03-15 19:16:09…	171477	97871
test.txt	BasicInfoChange	2022-03-15 19:16:31…	171477	97871
test.txt	BasicInfoChange|Close	2022-03-15 19:16:31…	171477	97871
test.txt	ObjectIdChange	2022-03-15 19:17:41…	171477	97871
test.txt	ObjectIdChange|Close	2022-03-15 19:17:41…	171477	97871
test.txt	FileDelete|Close	2022-03-15 19:27:24…	171477	97871

Figure 12. UsnJrnl

File “test2.txt” has similar information, only BasicInfoChange rows are missing, and this file was really deleted, the UsnJrnl output is presented in Appendix 5.

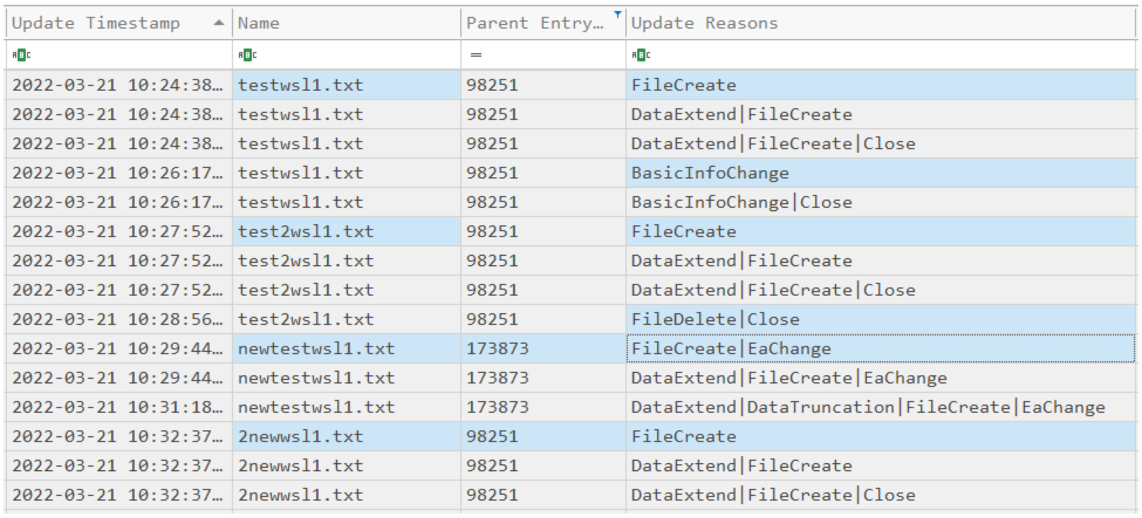
Documents that are opened at least once can be found in NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.txt. This registry hives stores up to 50 recent documents if the default value not changed.

5.3.2 File operations in WSL1

In $MFT output we can see only two files “testwsl1.txt” and “2newwsl1.txt” saved to Windows file system, deleted, or saved to Ubuntu are not present.

The ffirst file “testwsl1.txt” was saved to Windows and got full set of permissions by default, but, writing permissions were removed with “chmod” command. The file “newtestwsl1.txt” was saved to Ubuntu’s file system and showed only reading rights but the test showed that the file was also writable. The last file was moved to Windows and by that it got full permissions (777) in Windows file system, which is unexpected, because moving a file from Windows to Ubuntu did not change permissions, this could present a threat to the system.

UsnJrnl analysis shows all file creations, modifications, and deletions. Unlike WSL2, the file saved to Ubuntu is also present in journal output. Parent entry value changes when file saved to Windows or Ubuntu. File “newtestwsl1.txt” was saved to Ubuntu and in journal output it has parent entry value 173873, same file was moved and renamed to Windows and the new file “2newwsl1.txt” has parent file entry value 98251, which is the same as file created earlier. In Figure 13 are all four text files created, .swp fails also exist like in WSL2.



Update Timestamp	Name	Parent Entry…	Update Reasons
2022-03-21 10:24:38…	testwsl1.txt	98251	FileCreate
2022-03-21 10:24:38…	testwsl1.txt	98251	DataExtend|FileCreate
2022-03-21 10:24:38…	testwsl1.txt	98251	DataExtend|FileCreate|Close
2022-03-21 10:26:17…	testwsl1.txt	98251	BasicInfoChange
2022-03-21 10:26:17…	testwsl1.txt	98251	BasicInfoChange|Close
2022-03-21 10:27:52…	test2wsl1.txt	98251	FileCreate
2022-03-21 10:27:52…	test2wsl1.txt	98251	DataExtend|FileCreate
2022-03-21 10:27:52…	test2wsl1.txt	98251	DataExtend|FileCreate|Close
2022-03-21 10:28:56…	test2wsl1.txt	98251	FileDelete|Close
2022-03-21 10:29:44…	newtestwsl1.txt	173873	FileCreate|EaChange
2022-03-21 10:29:44…	newtestwsl1.txt	173873	DataExtend|FileCreate|EaChange
2022-03-21 10:31:18…	newtestwsl1.txt	173873	DataExtend|DataTruncation|FileCreate|EaChange
2022-03-21 10:32:37…	2newwsl1.txt	98251	FileCreate
2022-03-21 10:32:37…	2newwsl1.txt	98251	DataExtend|FileCreate
2022-03-21 10:32:37…	2newwsl1.txt	98251	DataExtend|FileCreate|Close

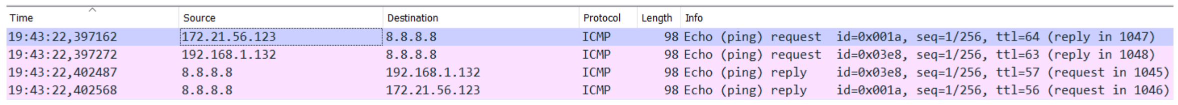
Figure 13. WSL1 journal

5.4 Network operations

This section analyzes network traffic to see if WSL is distinguishable from network traffic. The experiment included ping command, downloading from Internet, and establishing a telnet connection.

5.4.1 Network operations in WSL2

Ping.exe execution can be found only when run directly on Windows. Sysmon logged ping.exe when host echoed WSL2's IP, same goes for Prefetch. When Ubuntu is used, it is possible to detect ping only from network logs. We can see from Wireshark pcap that pingig 8.8.8.8 has also host IP but this doesn't affect ping.exe execution information on Windows system.

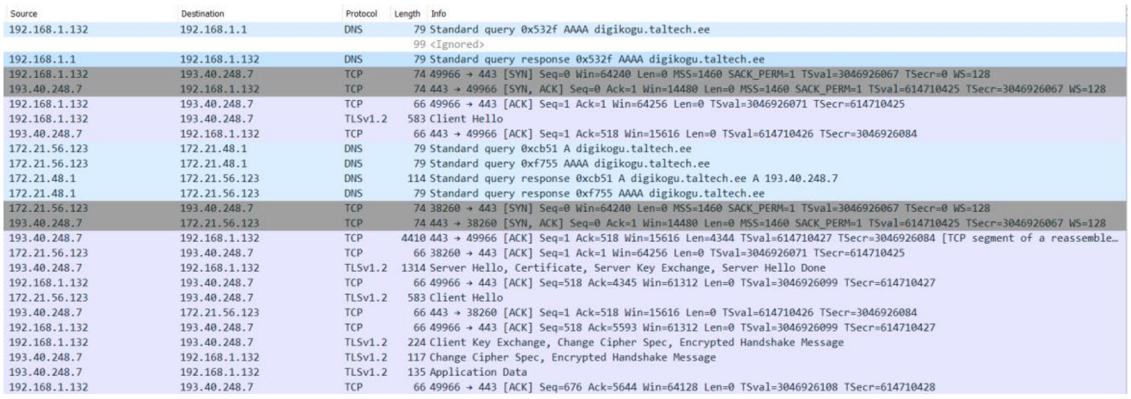


Time	Source	Destination	Protocol	Length	Info
19:43:22,397162	172.21.56.123	8.8.8.8	ICMP	98	Echo (ping) request id=0x001a, seq=1/256, ttl=64 (reply in 1047)
19:43:22,397272	192.168.1.132	8.8.8.8	ICMP	98	Echo (ping) request id=0x03e8, seq=1/256, ttl=63 (reply in 1048)
19:43:22,402487	8.8.8.8	192.168.1.132	ICMP	98	Echo (ping) reply id=0x03e8, seq=1/256, ttl=57 (request in 1045)
19:43:22,402568	8.8.8.8	172.21.56.123	ICMP	98	Echo (ping) reply id=0x001a, seq=1/256, ttl=56 (request in 1046)

Figure 14. Ping WSL2 -> 8.8.8.8

In Figure 14 is ping request from WSL2 to Google, we can see there are two requests from different IPs, one is WSL's and second is host's IP. The reason there are two IPs is that WSL2 uses NAT. It is possible to detect request made by WSL by User-Agent, which is Debian APT-HTTP/1.3 (2.02)

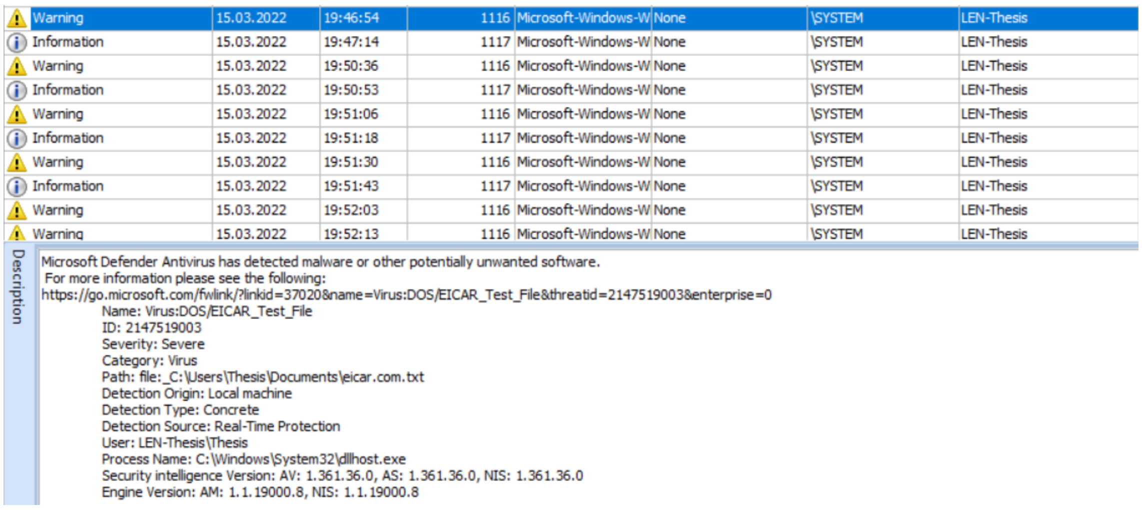
Curl command with two source IPs is in Figure 15.



```
Source           Destination      Protocol  Length Info
192.168.1.132    192.168.1.1      DNS         79 Standard query 0x532f AAAA digikogu.taltech.ee
                                              99 <Ignored>
192.168.1.1      192.168.1.132    DNS         79 Standard query response 0x532f AAAA digikogu.taltech.ee
192.168.1.132    193.40.248.7     TCP         74 49966 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3046926067 TSecr=0 WS=128
193.40.248.7     192.168.1.132    TCP         74 443 → 49966 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=614710425 TSecr=3046926067 WS=128
192.168.1.132    193.40.248.7     TCP         66 49966 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3046926071 TSecr=614710425
192.168.1.132    193.40.248.7     TLSv1.2    583 Client Hello
193.40.248.7     192.168.1.132    TCP         66 443 → 49966 [ACK] Seq=1 Ack=518 Win=15616 Len=0 TSval=614710426 TSecr=3046926084
172.21.56.123    172.21.48.1      DNS         79 Standard query 0xcb51 A digikogu.taltech.ee
172.21.56.123    172.21.48.1      DNS         79 Standard query 0xf755 AAAA digikogu.taltech.ee
172.21.48.1      172.21.56.123    DNS        114 Standard query response 0xcb51 A digikogu.taltech.ee A 193.40.248.7
172.21.48.1      172.21.56.123    DNS         79 Standard query response 0xf755 AAAA digikogu.taltech.ee
172.21.56.123    193.40.248.7     TCP         74 38260 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3046926067 TSecr=0 WS=128
193.40.248.7     172.21.56.123    TCP         74 443 → 38260 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=614710425 TSecr=3046926067 WS=128
193.40.248.7     192.168.1.132    TCP       4410 443 → 49966 [ACK] Seq=1 Ack=518 Win=15616 Len=4344 TSval=614710427 TSecr=3046926084 [TCP segment of a reassemble…
172.21.56.123    193.40.248.7     TCP         66 38260 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3046926071 TSecr=614710425
193.40.248.7     192.168.1.132    TLSv1.2   1314 Server Hello, Certificate, Server Key Exchange, Server Hello Done
192.168.1.132    193.40.248.7     TCP         66 49966 → 443 [ACK] Seq=518 Ack=4345 Win=61312 Len=0 TSval=3046926099 TSecr=614710427
172.21.56.123    193.40.248.7     TLSv1.2    583 Client Hello
193.40.248.7     172.21.56.123    TCP         66 443 → 38260 [ACK] Seq=1 Ack=518 Win=15616 Len=0 TSval=614710426 TSecr=3046926084
192.168.1.132    193.40.248.7     TCP         66 49966 → 443 [ACK] Seq=518 Ack=5593 Win=61312 Len=0 TSval=3046926099 TSecr=614710427
192.168.1.132    193.40.248.7     TLSv1.2    224 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
193.40.248.7     192.168.1.132    TLSv1.2    117 Change Cipher Spec, Encrypted Handshake Message
193.40.248.7     192.168.1.132    TLSv1.2    135 Application Data
192.168.1.132    193.40.248.7     TCP         66 49966 → 443 [ACK] Seq=676 Ack=5644 Win=64128 Len=0 TSval=3046926108 TSecr=614710428
```

Figure 15. curl TalTech Library

Eicar malware test file was downloaded to Windows and Ubuntu. On Windows the file disappeared, and information why can be found in *Microsoft-Windows-Windows Defender%4Operational* log file. The Figure 16 shows that Real-Time Protection detected a virus but based on this log we cannot say this file is related to WSL.

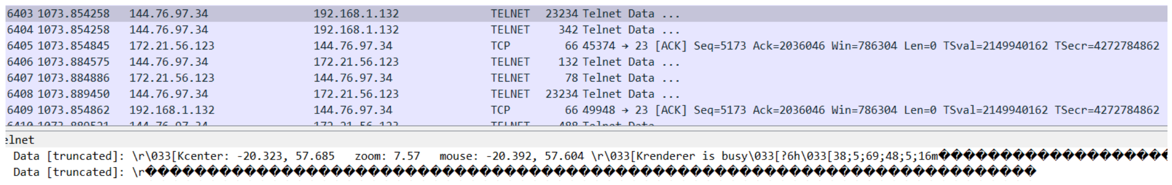


```
Warning      15.03.2022  19:46:54  1116  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Information  15.03.2022  19:47:14  1117  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Warning      15.03.2022  19:50:36  1116  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Information  15.03.2022  19:50:53  1117  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Warning      15.03.2022  19:51:06  1116  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Information  15.03.2022  19:51:18  1117  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Warning      15.03.2022  19:51:30  1116  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Information  15.03.2022  19:51:43  1117  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Warning      15.03.2022  19:52:03  1116  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
Warning      15.03.2022  19:52:13  1116  Microsoft-Windows-W  None  \SYSTEM  LEN-Thesis
```

Description
```
Microsoft Defender Antivirus has detected malware or other potentially unwanted software.
 For more information please see the following:
https://go.microsoft.com/fwlink/?linkid=37020&name=Virus:DOS/EICAR_Test_File&threatid=2147519003&enterprise=0
        Name: Virus:DOS/EICAR_Test_File
        ID: 2147519003
        Severity: Severe
        Category: Virus
        Path: file:_C:\Users\Thesis\Documents\eicar.com.txt
        Detection Origin: Local machine
        Detection Type: Concrete
        Detection Source: Real-Time Protection
        User: LEN-Thesis\Thesis
        Process Name: C:\Windows\System32\dllhost.exe
        Security intelligence Version: AV: 1.361.36.0, AS: 1.361.36.0, NIS: 1.361.36.0
        Engine Version: AM: 1.1.19000.8, NIS: 1.1.19000.8
```

Figure 16. Microsoft-Windows-Windows Defender%4Operational.evtx

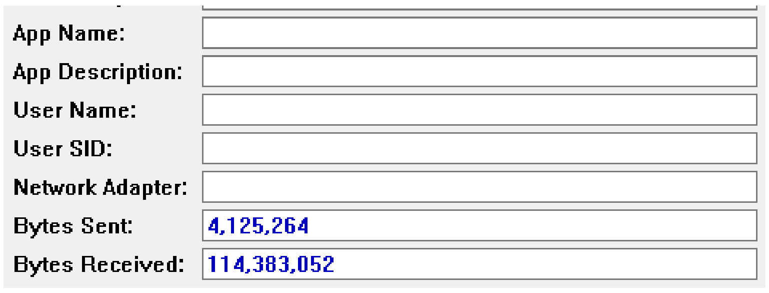
Telnet connection to command line map mapscii.me is shown in Figure 17. Because Telnet is not encrypted the data can be read. In this case we can see which location were viewed on the map. In this case coordinates -20,323 57,685 refer to Indian ocean island Mauritius.



```
6403 1073.854258    144.76.97.34           192.168.1.132          TELNET   23234 Telnet Data ...
6404 1073.854258    144.76.97.34           192.168.1.132          TELNET     342 Telnet Data ...
6405 1073.854845    172.21.56.123          144.76.97.34           TCP         66 45374 → 23 [ACK] Seq=5173 Ack=2036046 Win=786304 Len=0 TSval=2149940162 TSecr=4272784862
6406 1073.884575    144.76.97.34           172.21.56.123          TELNET     132 Telnet Data ...
6407 1073.884886    172.21.56.123          144.76.97.34           TELNET      78 Telnet Data ...
6408 1073.889450    144.76.97.34           172.21.56.123          TELNET   23234 Telnet Data ...
6409 1073.854862    192.168.1.132          144.76.97.34           TCP         66 49948 → 23 [ACK] Seq=5173 Ack=2036046 Win=786304 Len=0 TSval=2149940162 TSecr=4272784862
elnet
  Data [truncated]: \r\033[Kcenter: -20.323, 57.685   zoom: 7.57   mouse: -20.392, 57.604 \r\033[Krenderer is busy\033[?6h\033[38;5;69;48;5;16m�������������������
  Data [truncated]: \r�������������������������������������������������������������������������
```

Figure 17. Telnet mapscii.me

In case network monitoring is not set up, SRUM analysis shows that there was network traffic but information what app caused it is missing. The Figure 18 shows that most of the important information is missing.

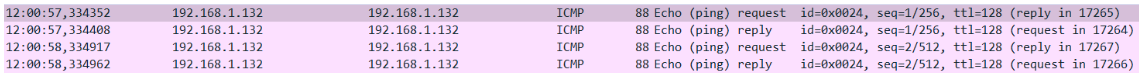


App Name:	
App Description:	
User Name:	
User SID:	
Network Adapter:	
Bytes Sent:	4,125,264
Bytes Received:	114,383,052

Figure 18. NetworkUsageView SRUDB.dat

5.4.2 Network operations in WSL1

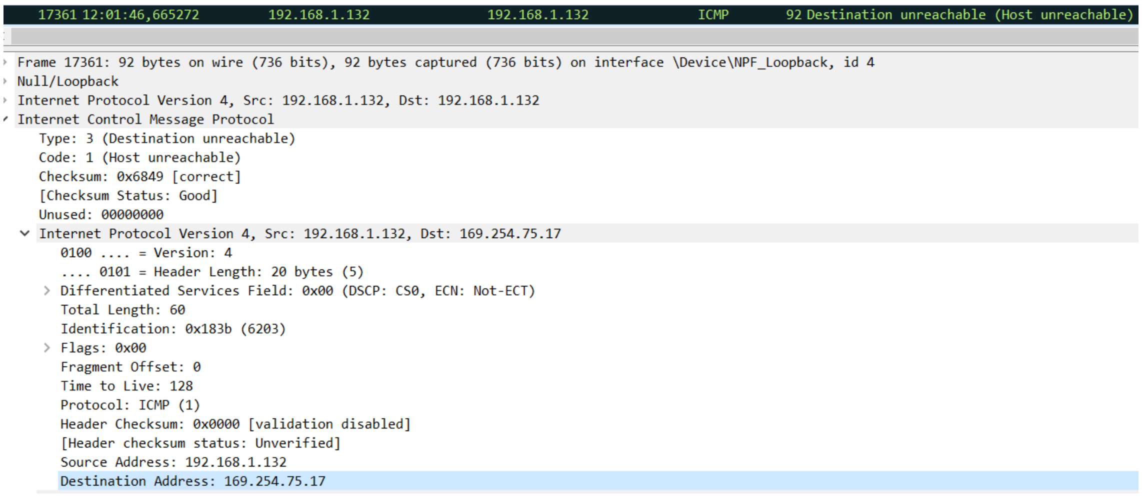
Ping request from WSL1 to host was successful and seen in Figure 19. Wireshark capture shows that the ping source and destination IPs were same. As mentioned in experimental setup section, WSL1 had six IPs in total, and one was same as host IP.



```
12:00:57,334352      192.168.1.132        192.168.1.132        ICMP     88 Echo (ping) request  id=0x0024, seq=1/256, ttl=128 (reply in 17265)
12:00:57,334408      192.168.1.132        192.168.1.132        ICMP     88 Echo (ping) reply    id=0x0024, seq=1/256, ttl=128 (request in 17264)
12:00:58,334917      192.168.1.132        192.168.1.132        ICMP     88 Echo (ping) request  id=0x0024, seq=2/512, ttl=128 (reply in 17267)
12:00:58,334962      192.168.1.132        192.168.1.132        ICMP     88 Echo (ping) reply    id=0x0024, seq=2/512, ttl=128 (request in 17266)
```

Figure 19. Ping from WSL to Host

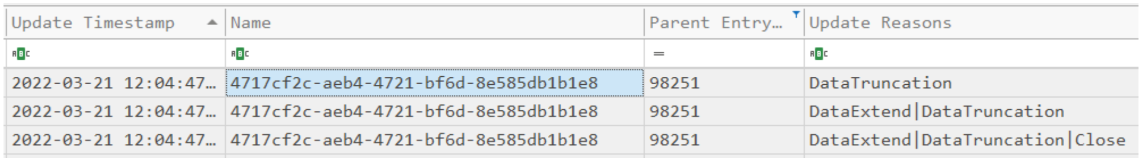
Echoing WSL ethernet interfaces was unsuccessful, the packet list pane in Wireshark shows again destination IP same as source IP, but typed IP was different from that, it was 169.254.75.17, and it is displayed in the packet details pane in Figure 20.



```
17361 12:01:46,665272    192.168.1.132    192.168.1.132    ICMP    92 Destination unreachable (Host unreachable)

› Frame 17361: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface \Device\NPF_Loopback, id 4
› Null/Loopback
› Internet Protocol Version 4, Src: 192.168.1.132, Dst: 192.168.1.132
⌄ Internet Control Message Protocol
    Type: 3 (Destination unreachable)
    Code: 1 (Host unreachable)
    Checksum: 0x6849 [correct]
    [Checksum Status: Good]
    Unused: 00000000
  ⌄ Internet Protocol Version 4, Src: 192.168.1.132, Dst: 169.254.75.17
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    › Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 60
      Identification: 0x183b (6203)
    › Flags: 0x00
      Fragment Offset: 0
      Time to Live: 128
      Protocol: ICMP (1)
      Header Checksum: 0x0000 [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 192.168.1.132
      Destination Address: 169.254.75.17
```

Figure 20. Ping from Host to WSL1

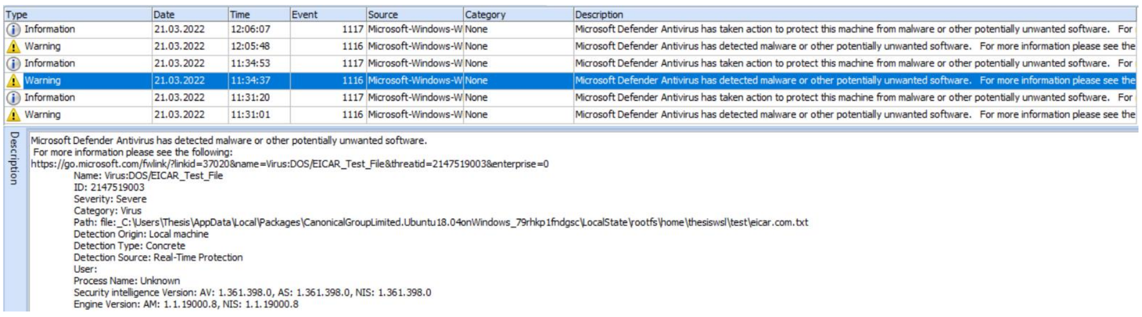
Downloading a file and asking for weather info with curl command also shows the same IP as host, the same goes for telnet connection. The reason for that is that WSL is in the same network as host, not like WSL2 which uses NAT. The file downloaded from TalTech library can be found in UsnJrnl and shown in Figure 21.



Update Timestamp	Name	Parent Entry…	Update Reasons
2022-03-21 12:04:47…	4717cf2c-aeb4-4721-bf6d-8e585db1b1e8	98251	DataTruncation
2022-03-21 12:04:47…	4717cf2c-aeb4-4721-bf6d-8e585db1b1e8	98251	DataExtend|DataTruncation
2022-03-21 12:04:47…	4717cf2c-aeb4-4721-bf6d-8e585db1b1e8	98251	DataExtend|DataTruncation|Close

Figure 21. File downloaded from TalTech in UsnJrnl

The Eicar malware test file was downloaded to both file systems, this time Windows Defender detects it in both systems, the log entry can be found in Figure 22 where *Microsoft-Windows-Windows Defender%4Operational.evtx*. output is displayed.



Type	Date	Time	Event	Source	Category	Description
Information	21.03.2022	12:06:07	1117	Microsoft-Windows-W	None	Microsoft Defender Antivirus has taken action to protect this machine from malware or other potentially unwanted software. For
Warning	21.03.2022	12:05:48	1116	Microsoft-Windows-W	None	Microsoft Defender Antivirus has detected malware or other potentially unwanted software. For more information please see the
Information	21.03.2022	11:34:53	1117	Microsoft-Windows-W	None	Microsoft Defender Antivirus has taken action to protect this machine from malware or other potentially unwanted software. For
Warning	21.03.2022	11:34:37	1116	Microsoft-Windows-W	None	Microsoft Defender Antivirus has detected malware or other potentially unwanted software. For more information please see the
Information	21.03.2022	11:31:20	1117	Microsoft-Windows-W	None	Microsoft Defender Antivirus has taken action to protect this machine from malware or other potentially unwanted software. For
Warning	21.03.2022	11:31:01	1116	Microsoft-Windows-W	None	Microsoft Defender Antivirus has detected malware or other potentially unwanted software. For more information please see the

```
Microsoft Defender Antivirus has detected malware or other potentially unwanted software.
 For more information please see the following:
https://go.microsoft.com/fwlink/?linkid=37020&name=Virus:DOS/EICAR_Test_File&threatid=2147519003&enterprise=0
        Name: Virus:DOS/EICAR_Test_File
        ID: 2147519003
        Severity: Severe
        Category: Virus
        Path: file:_C:\Users\Thesis\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc\LocalState\rootfs\home\thesiswsl\test\eicar.com.txt
        Detection Origin: Local machine
        Detection Type: Concrete
        Detection Source: Real-Time Protection
        User:
        Process Name: Unknown
        Security intelligence Version: AV: 1.361.398.0, AS: 1.361.398.0, NIS: 1.361.398.0
        Engine Version: AM: 1.1.19000.8, NIS: 1.1.19000.8
```

Figure 22. Microsoft-Windows-Windows Defender%4Operational.evtx

The log description shows where WSL files are stored in Windows system:

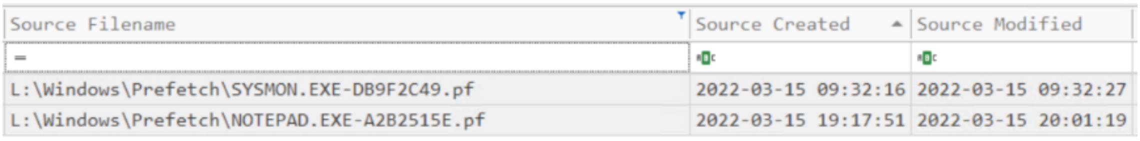
C:\Users\Thesis\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc\LocalState\rootfs\home\thesiswsl\test\eicar.com.txt Browsing to this rootfs location we can see a regular Ubuntu's root direction, shown in Appendix 6.

The Eicar file is present in journal with one parent entry number 98251 even the file was saved to both file systems.

SRUM network usage analysis in WSL1 unlike WSL2, shows bytes sent, bytes received, user SID, and app name which is the file system location […]\rootfs\usr\bin\curl, the output is shown in Appendix 7.

5.5 CLI and GUI applications in WSL2

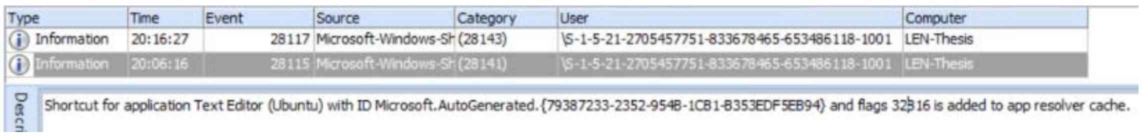
Executing Sysmon from bash displays program information and that administrator privileges are required. Notepad was also executed and a new file "123.txt" was saved to Windows Documents folder. Both file executions exist in Prefetch and are shown on Figure 23.



Source Filename	Source Created	Source Modified
=		
L:\Windows\Prefetch\SYSMON.EXE-DB9F2C49.pf	2022-03-15 09:32:16	2022-03-15 09:32:27
L:\Windows\Prefetch\NOTEPAD.EXE-A2B2515E.pf	2022-03-15 19:17:51	2022-03-15 20:01:19

Figure 23. Sysmon in prefetch

The first graphical application installed with apt command was a text editor Gedit, second was Firefox browser. Appendix 8 shows how Gedit download look from Network perspective. From Wireshark pcap we can see the User-Agent Debian APT-HTTP/1.3 (2.02). In Figure 25 in Microsoft-Windows-Shell-Core%4Operational event log we can see a log entry that describes that a shortcut for application Text Editor (Ubuntu) and Firefox Web Browser (Ubuntu) are added to app resolver cache.



Type	Time	Event	Source	Category	User	Computer
Information	20:16:27	28117	Microsoft-Windows-Sh	(28143)	\S-1-5-21-2705457751-833678465-653486118-1001	LEN-Thesis
Information	20:06:16	28115	Microsoft-Windows-Sh	(28141)	\S-1-5-21-2705457751-833678465-653486118-1001	LEN-Thesis

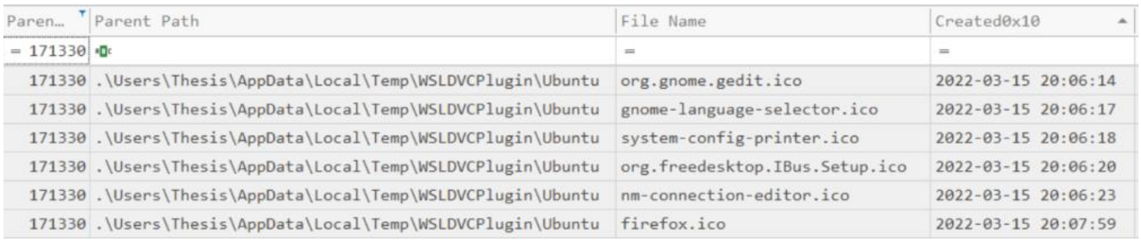
Descr

Shortcut for application Text Editor (Ubuntu) with ID Microsoft.AutoGenerated.{79387233-2352-954B-1CB1-B353EDF5EB94} and flags 32816 is added to app resolver cache.

Figure 24. Microsoft-Windows-Shell-Core%4Operational.evtx Text Editor entry

NTUSER.DAT registry hive Software\Microsoft\Windows\CurrentVersion\UFH\SHC contains data about graphical applications, not just these that were installed by a user, but also additional components like language support, ibus-setup, advanced network configuration (nm-connection-editor), printers (system.config-printer), these components were installed automatically. The output is presented in Appendix 9.

Both installed GUI applications are represented in MFT. Figure 26 displays the search of the parent entry number (171330), the output shows all previously mentioned additional components.



Paren...	Parent Path	File Name	Created0x10
= 171330		=	=
171330	.\Users\Thesis\AppData\Local\Temp\WSLDVCPlugin\Ubuntu	org.gnome.gedit.ico	2022-03-15 20:06:14
171330	.\Users\Thesis\AppData\Local\Temp\WSLDVCPlugin\Ubuntu	gnome-language-selector.ico	2022-03-15 20:06:17
171330	.\Users\Thesis\AppData\Local\Temp\WSLDVCPlugin\Ubuntu	system-config-printer.ico	2022-03-15 20:06:18
171330	.\Users\Thesis\AppData\Local\Temp\WSLDVCPlugin\Ubuntu	org.freedesktop.IBus.Setup.ico	2022-03-15 20:06:20
171330	.\Users\Thesis\AppData\Local\Temp\WSLDVCPlugin\Ubuntu	nm-connection-editor.ico	2022-03-15 20:06:23
171330	.\Users\Thesis\AppData\Local\Temp\WSLDVCPlugin\Ubuntu	firefox.ico	2022-03-15 20:07:59

Figure 25. GUI additional components in $MFT

Searching for the same parent entry number in UsnJrnl we can see both installed GUI programs. Gedit and Firefox are not present in Prefetch.

5.6 CLI operations in WSL1

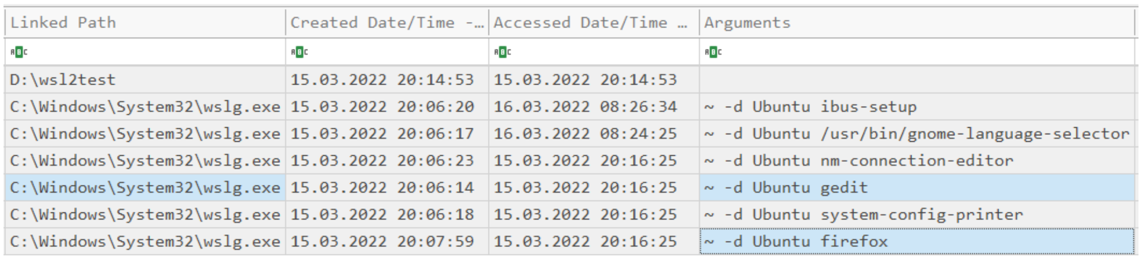
Prefetch has evidence of Sysmon and Notepad executions, but it does not have information about Gedit nor Firefox installation attempts. Prefetch files information is also present in $MFT output.

The Notepad.exe execution in journal in expressed with update reason "FileCreate" because it is a prefetch file, what was written to disk. The file "12345.txt" created in Notepad, is present in journal with the same parent entry number 98251 as other files saved to Windows, which is an indicator that WSL was used for creating this file not just Windows application. Sysmon execution is also present in journal output, both Sysmon.exe and prefetch file entries. Gedit and Firefox installation attempts are also represented in journal, but it is impossible to say if the installation was a success or not.

5.7 Magnet Axiom

In this section a commercia tool Magnet Axiom was used to find additional evidence that was not found before. Axiom was used to search some keywords and the results were exported.

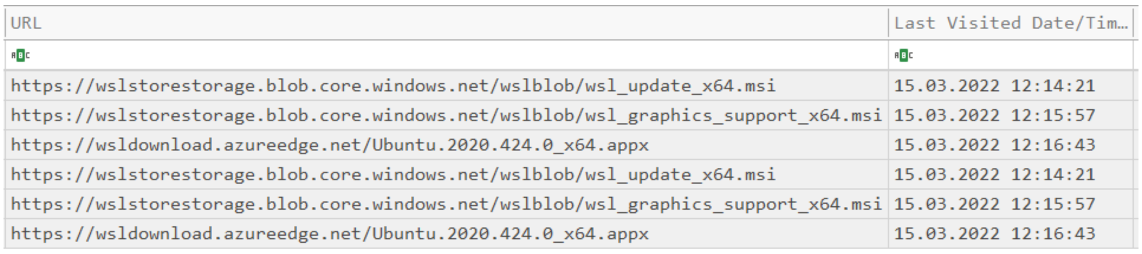
Windows Timeline Activity showed the time spent on WSL and Ubuntu, when was the application opened and closed, the output is shown in Appendix 10. Magnet Axiom parsed automatically all the LNK files from Windows operating system, the output is presented in Figure 26, and it displays the WSL2 graphical interface wslg.exe timestamps and command typed by user or system.



Linked Path	Created Date/Time -…	Accessed Date/Time …	Arguments
D:\wsl2test	15.03.2022 20:14:53	15.03.2022 20:14:53	
C:\Windows\System32\wslg.exe	15.03.2022 20:06:20	16.03.2022 08:26:34	~ -d Ubuntu ibus-setup
C:\Windows\System32\wslg.exe	15.03.2022 20:06:17	16.03.2022 08:24:25	~ -d Ubuntu /usr/bin/gnome-language-selector
C:\Windows\System32\wslg.exe	15.03.2022 20:06:23	15.03.2022 20:16:25	~ -d Ubuntu nm-connection-editor
C:\Windows\System32\wslg.exe	15.03.2022 20:06:14	15.03.2022 20:16:25	~ -d Ubuntu gedit
C:\Windows\System32\wslg.exe	15.03.2022 20:06:18	15.03.2022 20:16:25	~ -d Ubuntu system-config-printer
C:\Windows\System32\wslg.exe	15.03.2022 20:07:59	15.03.2022 20:16:25	~ -d Ubuntu firefox

Figure 26. Axiom LNK Files output

One evidence of WSL2 download can be found by examining Edge-Internet Explorer content, the content is displayed in Figure 27 and shows the websites connected in installation process. During the installation the user did not open the Edge browser.



URL	Last Visited Date/Tim…
https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi	15.03.2022 12:14:21
https://wslstorestorage.blob.core.windows.net/wslblob/wsl_graphics_support_x64.msi	15.03.2022 12:15:57
https://wsldownload.azureedge.net/Ubuntu.2020.424.0_x64.appx	15.03.2022 12:16:43
https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi	15.03.2022 12:14:21
https://wslstorestorage.blob.core.windows.net/wslblob/wsl_graphics_support_x64.msi	15.03.2022 12:15:57
https://wsldownload.azureedge.net/Ubuntu.2020.424.0_x64.appx	15.03.2022 12:16:43

Figure 27. Edge-Internet Explorer 10-11 Content

In Windows Firewall event log entries, we can find a description of new rule that has been added to exception list at the time of Ubuntu installation.

Exported Windows Defender Log output referred to a file “MPLog-20220315-090500.log”. Investigation of that file shows that is a text file which contains more detailed logs from Windows Defender. The MPLog contains historical evidence of process execution, threats detected, scan results and actions taken, signature update versions, and file existence [40] but the latter do not apply for all files, all the text files created in this experiment are not represented in this log file. This alternative location (ProgramData\Microsoft\Windows Defender\Support\) is important because one of the easiest anti forensic methods is to delete event logs folder when it is in default location (C:\Windows\System32\winevt\Logs).

6 Analysis summary

Previous chapters provided detailed overview and understanding about Windows Subsystem for Linux both versions and the forensics on Windows 11. This section is for pointing out the most important findings.

Registry hives are providing great evidence for WSL investigation. SOFTWARE, SYSTEM, NTUSER.DAT, UsrClass.dat, and Amcache.hve hives provided some information about WSL usage, starting from installation to some file operations. The comparison of WSL1 and WSL2 registry locations are presented in Appendix 10, the table shows which registry entry can be found in either version. The table can help the investigator to find information fast and appropriate to the case.

In some cases, it is important to know the first time the activity occurred. The Application Combability or AppCompatCache is a source to look for the earliest timestamps for WSL installation. Prefetch is a place to look for program executions, programs that were present and executed from Windows file system are listed in Prefetch folder, the real execution time may be up to 10 seconds earlier.

Windows event logs are providing various entries about WSL's actions. To have even better overview of the system, the test computer had Sysmon installed. Sysmon log entry had the full installation command even if it was not typed in by user. It is also a source for installed distribution, SHA256 hash, IDs, and location where the files are stored in Windows files system. WSL2 uses Hyper-V, different Hyper-V log files are providing information about network adapters, ports, virtual machine IDs, VM start and shutdown events, IDs and SIDs are the same throughout the Hyper-V logs, but different from other event logs. Like Sysmon the *Microsoft-Windows-Shell-Core%4Operational* log includes full command line, and command execution and finish events. The *Microsoft-Windows-VHDMP-Operational* is a place for virtual disk information. The *AppXDeploymentServer%4Operational* and *Application* logs have installation process overview, times, and SIDs. In Table 5 are event logs that can be used in either WSL1 or WSL2 investigation. Windows Defender log in WSL2 is considered as optional, because it cannot tie to WSL2, it detected malware from Windows file system without a sign of Ubuntu. Windows Defender can detect at least some malware from WSL1, but not from

WSL2. The Eicar test file was removed from Windows and WSL1 but remained in place in WSL2.

Table 5. Event Logs

Event Log	**WSL2**	**WSL1**
Sysmon	x	x
Application	x	
Security	x	x
Microsoft-Windows-VHDMP-Operational	x	
Microsoft-Windows-Shell-Core%4Operational	x	
AppXDeploymentServer%4Operational	x	x
Microsoft-Windows-Hyper-V-VmSwitch-Operational	x	
Microsoft-Windows-Hyper-V-Worker-Admin	x	
Hyper-V-Compute-Operational	x	
Microsoft-Windows-Store%4Operational	x	x
Microsoft-Windows-AppModel-Runtime	x	x
Microsoft-Windows-Windows Defender%4Operational	(x)	x

In Windows file system, the WSL1 files are in AppData folder in C:\Users\<user>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc\LocalState\rootfs\, where rootfs corresponds to root folder in native Linux. In case of WSL2 the LocalState folder has a virtual hard disk file ext4.vhdx file.

Unfortunately, not all commands typed by user are logged to Windows by default, only some commands can be found from Sysmon and *Microsoft-Windows-Shell-Core%4Operational* event logs. It is possible that enabling some extra logging may improve this shortcoming. Conducting Linux forensic may also give the command history but this was not in scope.

A File created in WSL2 and saved to Windows file system will have full permission. The File created in WSL2 and saved to Ubuntu file system will have only read permission. It is notable that when the file created to Linux file system is moved to Windows, it will have full permissions, it needs further investigation because it may present a threat to the system. The other way the file will inherit the permission set from Windows. When the file is first saved to Windows then moved to Ubuntu and again back to Windows, it will have the same set of permissions it had first time, not full permissions if it did not have them before so Windows do not see the file as a new file. MFT analysis shows only the files that are present in Windows, no deleted files, nor Ubuntu’s files. When the file was moved from WSL2 to Windows its creation time in MFT was later than modification time, which is an indication that the file was created somewhere else. In case of WSL1 the creation time is the same as modification time.

Journal analysis provide very detailed overview what happened in a system. It logs also the swp files, which can be indication that the file was created in Linux screen oriented VI application which can be used as a evidence that a file is created in WSL.

WSL2 in Windows 11 allows users to use GUI applications, in Windows 10 or in WSL1 it is not supported. GUI applications leave evidence to Windows Event logs, MFT, UsnJrnl, and NTUSER.DAT registry hive, but not to Windows Prefetch.

From network perspective WSL1 and WSL2 are different. WSL1 uses the same subnet as the host, but WSL2 uses NAT. In case network monitoring is possible, WSL1 looks like regular traffic from host machine, WSL2 has second IP and HTTP user agent is distribution specific, in this case it was Debian APT-HTTP/1.3 (2.02). In conclusion WSL2 traffic can be distinguished from host traffic but WSL1 traffic is indistinguishable. SRUM analysis can be used to detect WSL1 network operations. WSL2 GUI browser can be used to bypass Windows Firewall network rules. In this work we showed that blocking port 80, and 443 affect host’s web browsers but WSL2’s browser was able to continue Internet browsing.

Commercial tools are fast and easy to use, the Magnet Axiom provided all evidence at once in one place. In this work Axiom was used to catch some extra evidence overlooked by manual analysis. It helped to detect Edge content, which included web pages that installation process used. A not very common or known forensic artifact MPLog came out investigating Axiom’s Windows Defender log output. In some cases, it could contain historical evidence of process execution, threats detected, scan results and actions taken, signature update versions, and file existence [40]. It can be very useful in cases of some anti forensic techniques are in use because the artifact is not well known and is not located in the same folder as other Windows event logs and therefore may remain in place.

7 Conclusions and Future work

Windows Subsystem for Linux has evolved rapidly from 2016 and continues to evolve. WSL2 introduced a introduced new architecture and on Windows 11 the support for GUI applications.

WSL is meant for developers and other specialist who need to use Windows and Linux operating systems simultaneously without the traditional virtual machine environment. WSL is definitely not a security measure, quite the opposite, it brings many new ways to attack the system, and incident responders need to know how to find evidence of WSL use.

In this thesis, we investigated WSL 1 and WSL 2 on Windows 11 version 21H2. The goal was to be comprehensive and find as much forensic evidence as possible since sometimes some artifacts can be overlooked by investigators, or anti-forensic measures can be used to hide evidence.

The experiment was based on a basic user behaviour, like file operations, downloading from Internet, and opening applications, advanced actions were not included.

There are many forensic artifacts left of the WSL existence, but not all the actions can be easily bind to WSL. Forensic examiner must look for different artifacts to understand where the files were created or what caused the network traffic. WSL1 files are visible from Windows file explorer, in the equivalent situation with WSL2 the files are stored in a Hyper-V virtual hard disk instead. From the network perspective, WSL2 stands out whereas WSL1 does not, unexpectedly analysing the SRUM database's network information it is possible to get more information about WSL1 than WSL2.

Regarding the future work, memory forensics was not covered in this thesis however it is something that could potentially provide valuable evidence. Another aspect for future research involves logging. Windows does not log everything by default, therefore enabling features that enhance logging could also be a way to continue the research.

References

[1] K. Öteyo, „Running Linux GUI applications on Windows Subsystem for Linux (WSL),“ Computing for Geeks, 28 02 2022. [Võrgumaterjal]. Available: https://computingforgeeks.com/running-linux-gui-applications-on-windows-subsystem-for-linux/. [Kasutatud 25 03 2022].

[2] R. Turner, „Windows Subsystem for Linux out of Beta!,“ Microsoft, 28 07 2017. [Võrgumaterjal]. Available: https://devblogs.microsoft.com/commandline/windows-subsystem-for-linux-out-of-beta/. [Kasutatud 25 03 2022].

[3] „This is the home of the Cygwin project,“ Cygwin , 09 02 2022. [Võrgumaterjal]. Available: https://cygwin.com/index.html. [Kasutatud 25 03 2022].

[4] „WSL,“ Ubuntu, 10 02 2021. [Võrgumaterjal]. Available: https://wiki.ubuntu.com/WSL. [Kasutatud 25 03 2022].

[5] N. Lewis, A. Case, A. Ali-Gombe ja G. G. Richard, „Memory forensics and the Windows Subsystem for Linux,“ *Digital Investigation,* kd. 26, nr DFRWS 2018 USA, 2018.

[6] D. Thomas, „Windows Subsystem for Linux Overview,“ Microsoft, 22 04 2016. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/archive/blogs/wsl/windows-subsystem-for-linux-overview. [Kasutatud 25 03 2022].

[7] „WSL File System Support,“ Microsoft, 15 06 2016. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/archive/blogs/wsl/wsl-file-system-support. [Kasutatud 02 04 2022].

[8] C. Loewen, „A Deep Dive Into How WSL Allows Windows to Access Linux Files,“ Micrososft, 30 05 2019. [Võrgumaterjal]. Available: https://devblogs.microsoft.com/commandline/a-deep-dive-into-how-wsl-allows-windows-to-access-linux-files/. [Kasutatud 02 04 2022].

[9] „What is the Windows Subsystem for Linux?,“ Microsoft, 29 12 2021. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/wsl/about. [Kasutatud 26 03 2022].

[10] T. Maurer, „Install WSL 2 on Windows 10,“ Thomas Maurer, 13 06 2019. [Võrgumaterjal]. Available: https://www.thomasmaurer.ch/2019/06/install-wsl-2-on-windows-10/. [Kasutatud 26 03 2022].

[11] „Comparing WSL 1 and WSL 2,“ Microsoft, 29 12 2021. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/wsl/compare-versions. [Kasutatud 26 03 2022].

[12] C. Loewen, „A Deep Dive Into How WSL Allows Windows to Access Linux Files,“ Microsoft, 30 05 2019. [Võrgumaterjal]. Available: https://devblogs.microsoft.com/commandline/a-deep-dive-into-how-wsl-allows-windows-to-access-linux-files/. [Kasutatud 26 03 2022].

[13] J. Kasten ja E. Chiwona, „[WSL 2] NIC Bridge mode 🖧 (Has TCP Workaround🔨)," Microsoft, 16 06 2019. [Võrgumaterjal]. Available: https://github.com/microsoft/WSL/issues/4150. [Kasutatud 02 04 2022].

[14] „Run Linux GUI apps on the Windows Subsystem for Linux (preview)," Microsoft, 12 02 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/wsl/tutorials/gui-apps. [Kasutatud 02 04 2022].

[15] „Windows Processor Requirements," Microsoft, 22 03 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-gb/windows-hardware/design/minimum/windows-processor-requirements. [Kasutatud 02 04 2022].

[16] Z. Bowden, „Windows 11 review: The start of a new era," Windows Central, 22 02 2022. [Võrgumaterjal]. Available: https://www.windowscentral.com/windows-11. [Kasutatud 02 04 2022].

[17] B. Gale, „Understanding Windows 11 TPM Support Requirements," Schnider Downs, 15 07 2021. [Võrgumaterjal]. Available: https://www.schneiderdowns.com/our-thoughts-on/windows-11-tpm-requirements. [Kasutatud 02 04 2022].

[18] E. Zimmerman, „Windows 11 testing. Did any artifacts change?," Twitter, 15 06 2021. [Võrgumaterjal]. Available: https://twitter.com/ericrzimmerman/status/1404859472275779584. [Kasutatud 02 04 2022].

[19] R. Lee, „FOR500 Windows Forensics," %1 *Windows FOrensics Analysis II: Windows Registry Forensics and Analysis*, SANS Institute, 2020, p. 186.

[20] R. Lee, „FOR500 WIndows Forensics," %1 *FOR 508.4: Windows Forensics Analysis IV: Email, Key Additional Artifacts, and Event Logs*, SANS Institute, 2020, p. 202.

[21] M. Russinovich ja T. Garnier, „Sysmon v13.33," Microsoft, 16 02 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon. [Kasutatud 05 04 2022].

[22] R. Lee, C. Tilbury ja M. Pilkington, „FOR508 Advanced incident response, threat hunting, and digital forensics," %1 *Advanced Adversary and Anti-Forensics Detection*, SANS Insitute, 2021, p. 117.

[23] *Digital Forensics*, U.S.Army, 2021.

[24] R. Lee, C. Tilbury ja M. Pilkington, „FOR508: Advanced Incident Response, Threat Hunting, and Digital Forensics," %1 *FOR508.3: Memory FOrensics in Incident Response and Threat Hunting*, SANS Institute, 2021, p. 205.

[25] P. Kochberger, S. Schrittwieser ja A. Tauber, „Assessment of the Transparency of theWindows Subsystem for Linux (WSL)," %1 *2019 International Conference on Software Security and Assurance (ICSSA)*, St. Pölten, 2019.

[26] A. Draeger, „Looking for Linux: WSL Key Evidence," 2021. [Võrgumaterjal]. Available: https://sansorg.egnyte.com/dl/qWyxoYdVIs. [Kasutatud 23 04 2022].

[27] G. Elbaz ja D. Atias, „Beware of the Bashware: A New Method for Any Malware to Bypass Security Solutions," Checkpoint, 11 09 2017. [Võrgumaterjal]. Available: https://research.checkpoint.com/2017/beware-bashware-new-method-malware-bypass-security-solutions/. [Kasutatud 06 04 2022].

[28] C. Morley, „Docplayer," 2020. [Võrgumaterjal]. Available: https://docplayer.net/190109889-Wsl-2-research-into-badness-f-secure-whitepaper-by-connor-morley.html. [Kasutatud 10 04 2022].

[29] „Linux under WSL2 can be leaking," Mullvad VPN, 30 09 2020. [Võrgumaterjal]. Available: https://mullvad.net/en/blog/2020/9/30/linux-under-wsl2-can-be-leaking/. [Kasutatud 10 04 2022].

[30] L. Abrams, „Windows Subsystem for Linux 2 bypasses the Windows 10 Firewall," BleepingComputer, 01 10 2020. [Võrgumaterjal]. Available: https://www.bleepingcomputer.com/news/microsoft/windows-subsystem-for-linux-2-bypasses-the-windows-10-firewall/. [Kasutatud 10 04 2022].

[31] A. Matadar, „Investigating WSL Endpoints," %1 *The 11th Annual Open Source Digital Forensics Conference (OSDFCon)*, Virtual Event, 2020.

[32] „Caine," Caine Project, [Võrgumaterjal]. Available: https://www.caine-live.net/page8/page8.html. [Kasutatud 03 04 2022].

[33] „FTK® Imager," exterro, 25 02 2022. [Võrgumaterjal]. Available: https://www.exterro.com/ftk-imager. [Kasutatud 05 04 2022].

[34] „Arsenal Image Mounter," Arsenal Recon, [Võrgumaterjal]. Available: https://arsenalrecon.com/faq/#AIMFAQ. [Kasutatud 23 03 2022].

[35] „Kroll Artifact Parser And Extractor (KAPE)," KROLL, [Võrgumaterjal]. Available: https://www.kroll.com/en/services/cyber-risk/incident-response-litigation-support/kroll-artifact-parser-extractor-kape. [Kasutatud 23 03 2022].

[36] „NetworkUsageView v1.26 - Displays network usage information stored in the SRUDB.dat database of Windows 10/8.," NirSoft, [Võrgumaterjal]. Available: https://www.nirsoft.net/utils/network_usage_view.html. [Kasutatud 23 04 2022].

[37] „Magnet Axiom," Magnet Forensics, 29 03 2022. [Võrgumaterjal]. Available: https://www.magnetforensics.com/products/magnet-axiom/. [Kasutatud 03 04 2022].

[38] „About Wireshark," Wireshark, 12 03 2022. [Võrgumaterjal]. Available: https://www.wireshark.org/. [Kasutatud 23 03 2022].

[39] „.SWP File Extension," FileInfo.com, 31 08 2021. [Võrgumaterjal]. Available: https://fileinfo.com/extension/swp. [Kasutatud 04 04 2022].

[40] J. Lovato, „Mind the MPLog: Leveraging Microsoft Protection Logging for Forensic Investigations," Crowdstrike, 20 01 2022. [Võrgumaterjal]. Available: https://www.crowdstrike.com/blog/how-to-use-microsoft-protection-logging-for-forensic-investigations/. [Kasutatud 11 04 2022].

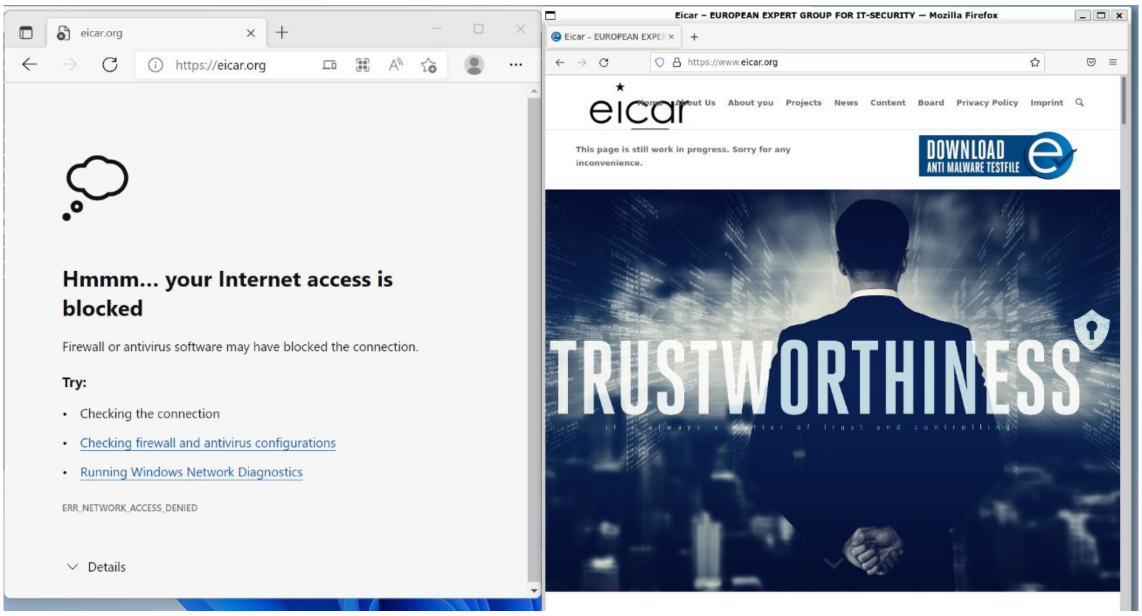
[41] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis1

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - Firewall bypass

One the left is Edge broser opened from Windows operating side, on the right is Firefox browser, which was opened from Ubuntu. The Firewall rule did not affect Internet browsing from WSL.



Hmmm... your Internet access is blocked

Firewall or antivirus software may have blocked the connection.

Try:

- Checking the connection
- Checking firewall and antivirus configurations
- Running Windows Network Diagnostics

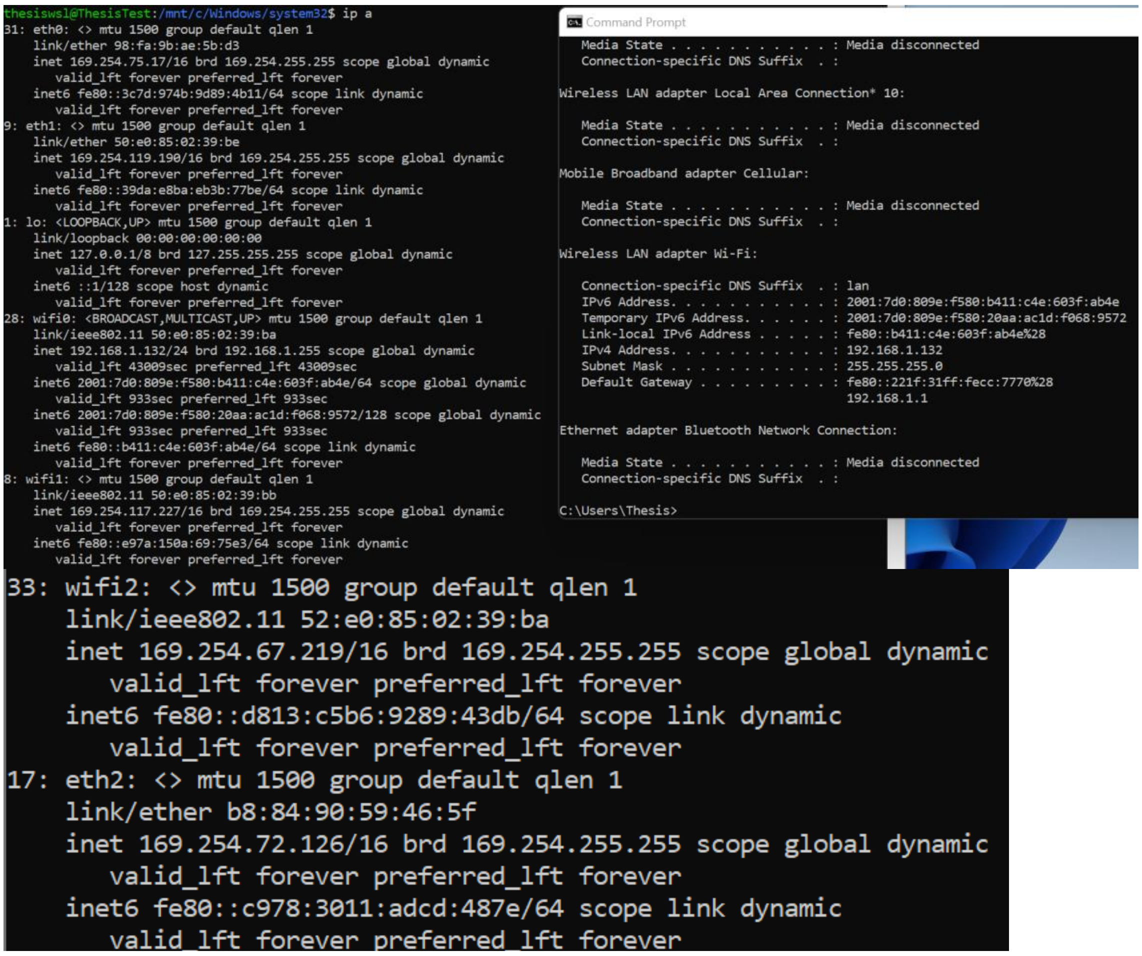
ERR_NETWORK_ACCESS_DENIED

Details

eicar.org
https://eicar.org
Eicar – EUROPEAN EXPERT GROUP FOR IT-SECURITY — Mozilla Firefox
https://www.eicar.org
eicar
Home
About Us
About you
Projects
News
Content
Board
Privacy Policy
Imprint
This page is still work in progress. Sorry for any inconvenience.
DOWNLOAD ANTI MALWARE TESTFILE
TRUSTWORTHINESS

Appendix 3 - WSL1 IP addresses

IP information realated to WSL version 1.



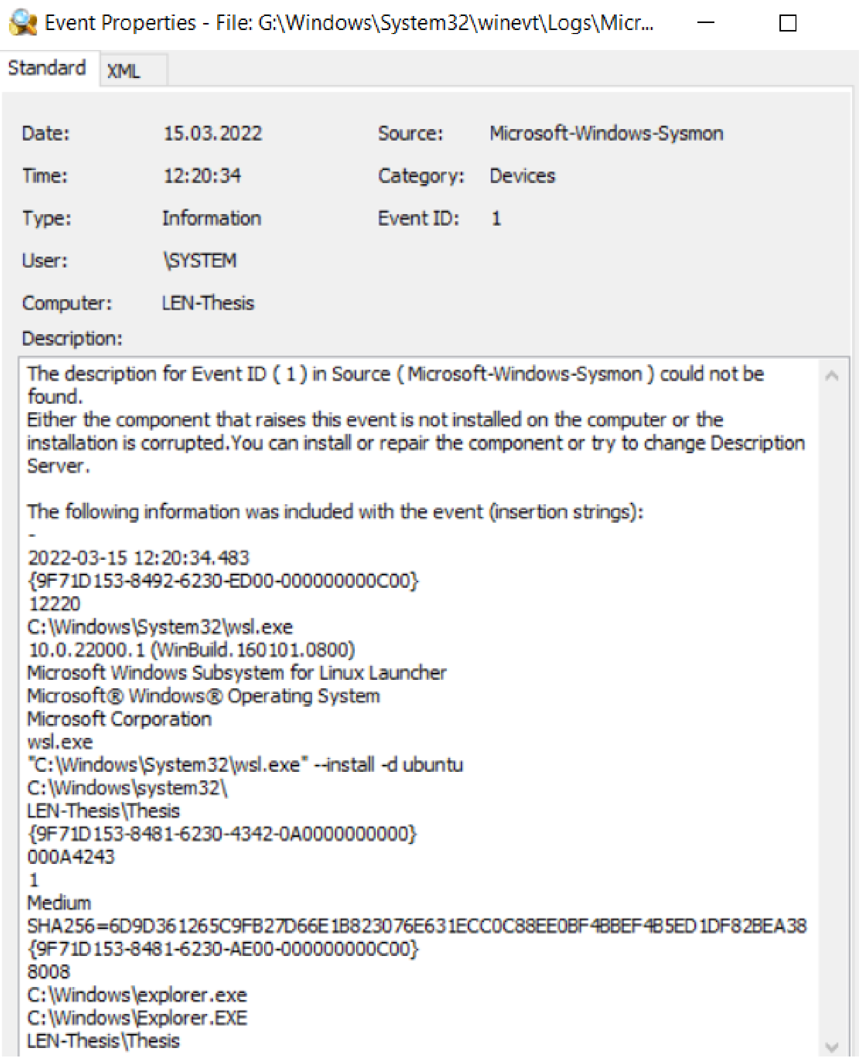
```
thesiswsl@ThesisTest:/mnt/c/Windows/system32$ ip a
31: eth0: <> mtu 1500 group default qlen 1
    link/ether 98:fa:9b:ae:5b:d3
    inet 169.254.75.17/16 brd 169.254.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::3c7d:974b:9d89:4b11/64 scope link dynamic
       valid_lft forever preferred_lft forever
9: eth1: <> mtu 1500 group default qlen 1
    link/ether 50:e0:85:02:39:be
    inet 169.254.119.190/16 brd 169.254.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::39da:e8ba:eb3b:77be/64 scope link dynamic
       valid_lft forever preferred_lft forever
1: lo: <LOOPBACK,UP> mtu 1500 group default qlen 1
    link/loopback 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host dynamic
       valid_lft forever preferred_lft forever
28: wifi0: <BROADCAST,MULTICAST,UP> mtu 1500 group default qlen 1
    link/ieee802.11 50:e0:85:02:39:ba
    inet 192.168.1.132/24 brd 192.168.1.255 scope global dynamic
       valid_lft 43009sec preferred_lft 43009sec
    inet6 2001:7d0:809e:f580:b411:c4e:603f:ab4e/64 scope global dynamic
       valid_lft 933sec preferred_lft 933sec
    inet6 2001:7d0:809e:f580:20aa:ac1d:f068:9572/128 scope global dynamic
       valid_lft 933sec preferred_lft 933sec
    inet6 fe80::b411:c4e:603f:ab4e/64 scope link dynamic
       valid_lft forever preferred_lft forever
8: wifi1: <> mtu 1500 group default qlen 1
    link/ieee802.11 50:e0:85:02:39:bb
    inet 169.254.117.227/16 brd 169.254.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::e97a:150a:69:75e3/64 scope link dynamic
       valid_lft forever preferred_lft forever
```

```
Command Prompt
   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Mobile Broadband adapter Cellular:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : lan
   IPv6 Address. . . . . . . . . . . : 2001:7d0:809e:f580:b411:c4e:603f:ab4e
   Temporary IPv6 Address. . . . . . : 2001:7d0:809e:f580:20aa:ac1d:f068:9572
   Link-local IPv6 Address . . . . . : fe80::b411:c4e:603f:ab4e%28
   IPv4 Address. . . . . . . . . . . : 192.168.1.132
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::221f:31ff:fecc:7770%28
                                       192.168.1.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\Thesis>
```

```
33: wifi2: <> mtu 1500 group default qlen 1
    link/ieee802.11 52:e0:85:02:39:ba
    inet 169.254.67.219/16 brd 169.254.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::d813:c5b6:9289:43db/64 scope link dynamic
       valid_lft forever preferred_lft forever
17: eth2: <> mtu 1500 group default qlen 1
    link/ether b8:84:90:59:46:5f
    inet 169.254.72.126/16 brd 169.254.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::c978:3011:adcd:487e/64 scope link dynamic
       valid_lft forever preferred_lft forever
```

Appendix 4 – Sysmon

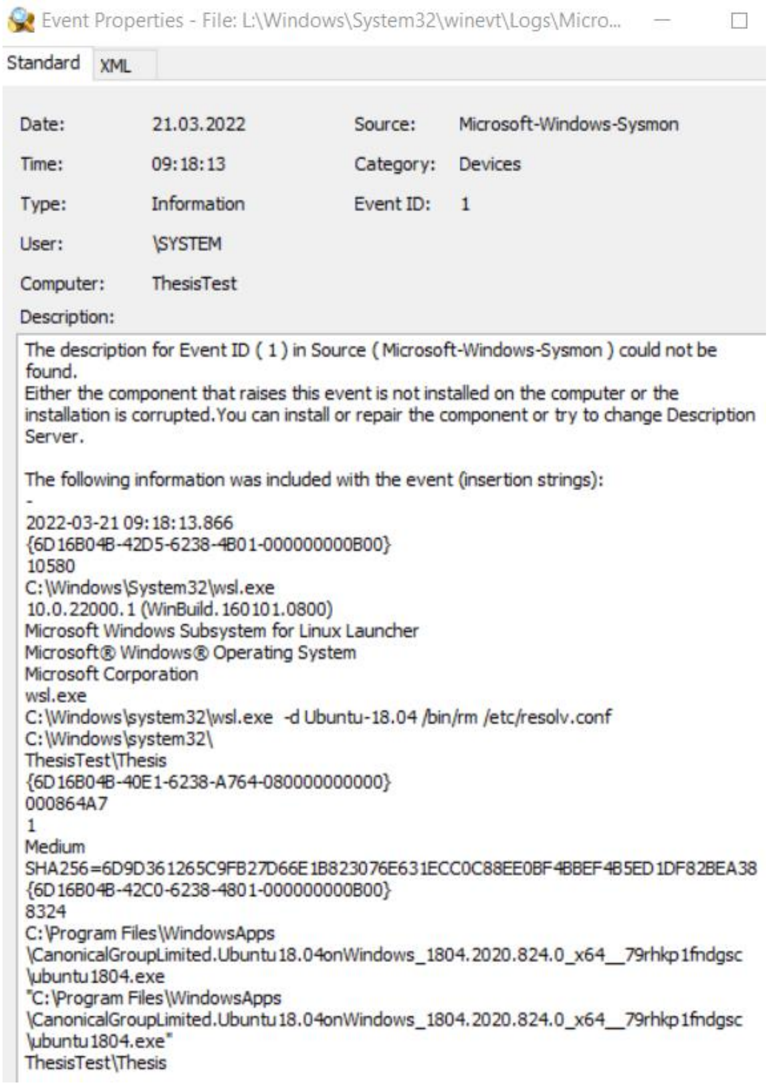
Sysmon event id 1 (Process creation). The event provides information about process creation with full command line, SHA256 hash, process GUID.


Event Properties - File: G:\Windows\System32\winevt\Logs\Micr...
Standard
XML

Date:	15.03.2022	Source:	Microsoft-Windows-Sysmon
Time:	12:20:34	Category:	Devices
Type:	Information	Event ID:	1
User:	\SYSTEM		
Computer:	LEN-Thesis		

Description:

```
The description for Event ID ( 1 ) in Source ( Microsoft-Windows-Sysmon ) could not be
found.
Either the component that raises this event is not installed on the computer or the
installation is corrupted.You can install or repair the component or try to change Description
Server.

The following information was included with the event (insertion strings):
-
2022-03-15 12:20:34.483
{9F71D153-8492-6230-ED00-000000000C00}
12220
C:\Windows\System32\wsl.exe
10.0.22000.1 (WinBuild.160101.0800)
Microsoft Windows Subsystem for Linux Launcher
Microsoft® Windows® Operating System
Microsoft Corporation
wsl.exe
"C:\Windows\System32\wsl.exe" --install -d ubuntu
C:\Windows\system32\
LEN-Thesis\Thesis
{9F71D153-8481-6230-4342-0A0000000000}
000A4243
1
Medium
SHA256=6D9D361265C9FB27D66E1B823076E631ECC0C88EE0BF4BBEF4B5ED1DF82BEA38
{9F71D153-8481-6230-AE00-000000000C00}
8008
C:\Windows\explorer.exe
C:\Windows\Explorer.EXE
LEN-Thesis\Thesis
```



Event Properties - File: L:\Windows\System32\winevt\Logs\Micro...

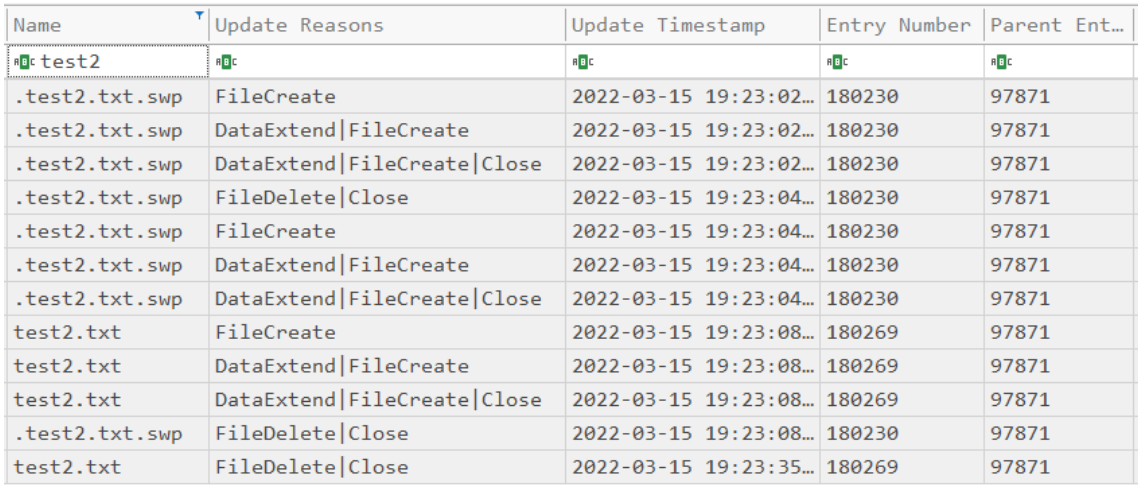
Standard XML

```
Date:       21.03.2022      Source:     Microsoft-Windows-Sysmon
Time:       09:18:13        Category:   Devices
Type:       Information     Event ID:   1
User:       \SYSTEM
Computer:   ThesisTest
Description:
```

```
The description for Event ID ( 1 ) in Source ( Microsoft-Windows-Sysmon ) could not be
found.
Either the component that raises this event is not installed on the computer or the
installation is corrupted.You can install or repair the component or try to change Description
Server.

The following information was included with the event (insertion strings):
-
2022-03-21 09:18:13.866
{6D16B04B-42D5-6238-4B01-000000000B00}
10580
C:\Windows\System32\wsl.exe
10.0.22000.1 (WinBuild.160101.0800)
Microsoft Windows Subsystem for Linux Launcher
Microsoft® Windows® Operating System
Microsoft Corporation
wsl.exe
C:\Windows\system32\wsl.exe  -d Ubuntu-18.04 /bin/rm /etc/resolv.conf
C:\Windows\system32\
ThesisTest\Thesis
{6D16B04B-40E1-6238-A764-080000000000}
000864A7
1
Medium
SHA256=6D9D361265C9FB27D66E1B823076E631ECC0C88EE0BF4BBEF4B5ED1DF82BEA38
{6D16B04B-42C0-6238-4801-000000000B00}
8324
C:\Program Files\WindowsApps
\CanonicalGroupLimited.Ubuntu18.04onWindows_1804.2020.824.0_x64__79rhkp1fndgsc
\ubuntu1804.exe
"C:\Program Files\WindowsApps
\CanonicalGroupLimited.Ubuntu18.04onWindows_1804.2020.824.0_x64__79rhkp1fndgsc
\ubuntu1804.exe"
ThesisTest\Thesis
```

Appendix 5 – Test2.txt

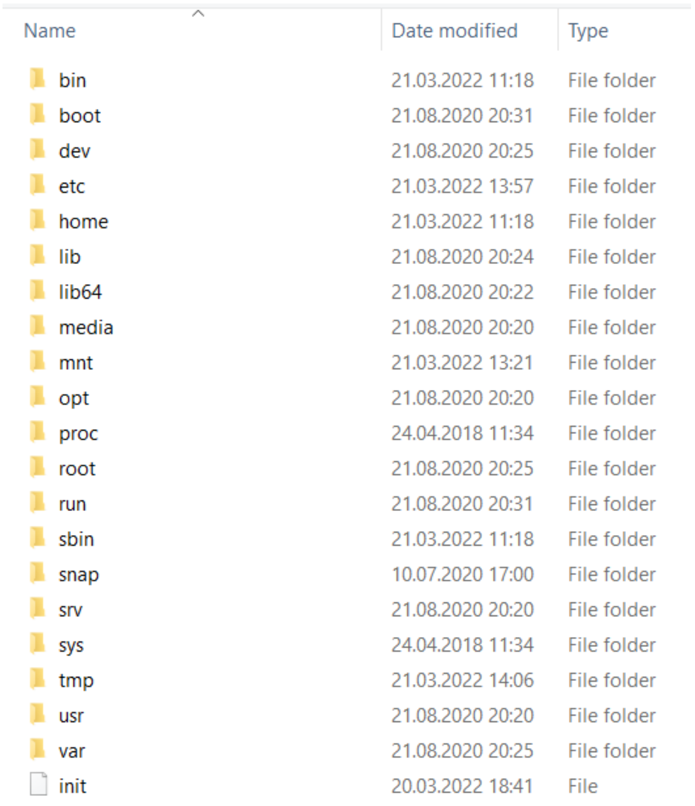
Journaling information about the second file “test2.txt”, the file was deleted after creation.



Name	Update Reasons	Update Timestamp	Entry Number	Parent Ent…
test2				
.test2.txt.swp	FileCreate	2022-03-15 19:23:02…	180230	97871
.test2.txt.swp	DataExtend|FileCreate	2022-03-15 19:23:02…	180230	97871
.test2.txt.swp	DataExtend|FileCreate|Close	2022-03-15 19:23:02…	180230	97871
.test2.txt.swp	FileDelete|Close	2022-03-15 19:23:04…	180230	97871
.test2.txt.swp	FileCreate	2022-03-15 19:23:04…	180230	97871
.test2.txt.swp	DataExtend|FileCreate	2022-03-15 19:23:04…	180230	97871
.test2.txt.swp	DataExtend|FileCreate|Close	2022-03-15 19:23:04…	180230	97871
test2.txt	FileCreate	2022-03-15 19:23:08…	180269	97871
test2.txt	DataExtend|FileCreate	2022-03-15 19:23:08…	180269	97871
test2.txt	DataExtend|FileCreate|Close	2022-03-15 19:23:08…	180269	97871
.test2.txt.swp	FileDelete|Close	2022-03-15 19:23:08…	180230	97871
test2.txt	FileDelete|Close	2022-03-15 19:23:35…	180269	97871

Appendix 6 – Linux file system 0n Windows file system (v1)

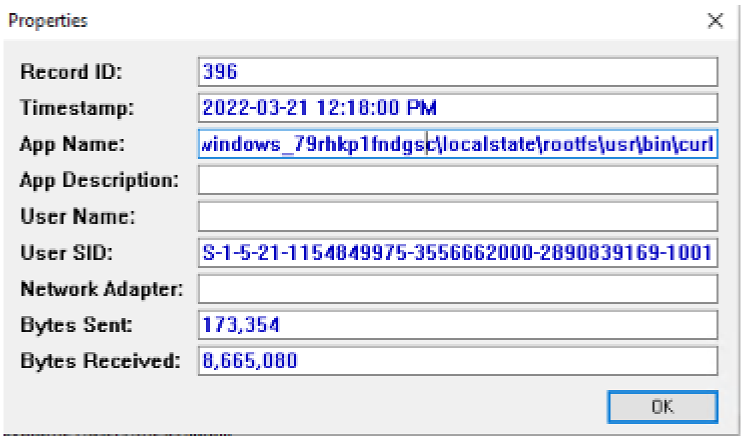
WSL1 files are stored directly on Windows. WSL2 files are stored on virtual disk.



Name	Date modified	Type
bin	21.03.2022 11:18	File folder
boot	21.08.2020 20:31	File folder
dev	21.08.2020 20:25	File folder
etc	21.03.2022 13:57	File folder
home	21.03.2022 11:18	File folder
lib	21.08.2020 20:24	File folder
lib64	21.08.2020 20:22	File folder
media	21.08.2020 20:20	File folder
mnt	21.03.2022 13:21	File folder
opt	21.08.2020 20:20	File folder
proc	24.04.2018 11:34	File folder
root	21.08.2020 20:25	File folder
run	21.08.2020 20:31	File folder
sbin	21.03.2022 11:18	File folder
snap	10.07.2020 17:00	File folder
srv	21.08.2020 20:20	File folder
sys	24.04.2018 11:34	File folder
tmp	21.03.2022 14:06	File folder
usr	21.08.2020 20:20	File folder
var	21.08.2020 20:25	File folder
init	20.03.2022 18:41	File

Appendix 7 – SRUM.dat

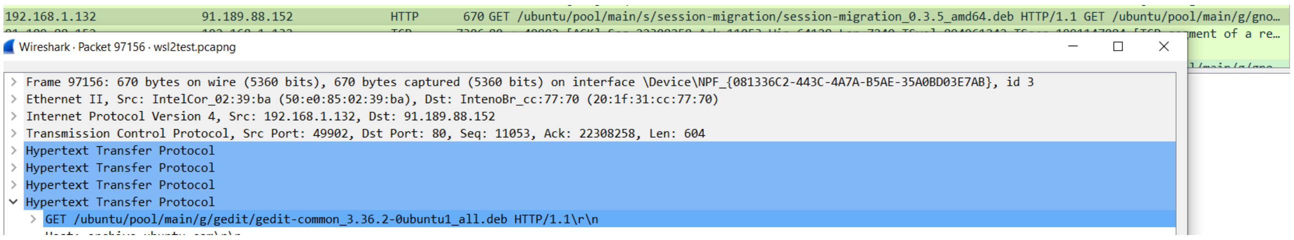
WSL1 SRUM analysis contains more information than WSL2 SRUM. We can see that app name that was used for network operations is refers to WSL, in case of WSL2 the app name line was empty.



Record ID:	396
Timestamp:	2022-03-21 12:18:00 PM
App Name:	windows_79rhkp1fndgsc\localstate\rootfs\usr\bin\curl
App Description:	
User Name:	
User SID:	S-1-5-21-1154849975-3556662000-2890839169-1001
Network Adapter:	
Bytes Sent:	173,354
Bytes Received:	8,665,080

Appendix 8 – HTTP request

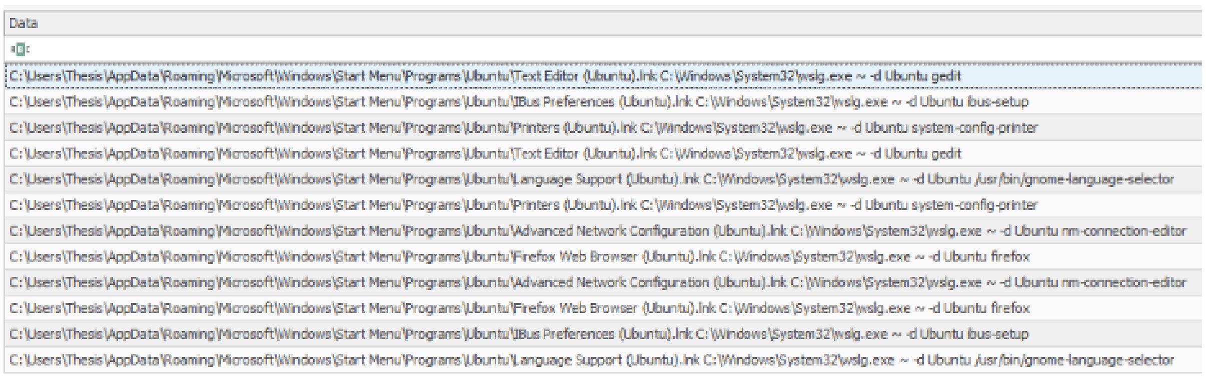
How downloading the Gedit text editor look like from network perspective.



```
192.168.1.132    91.189.88.152    HTTP    670 GET /ubuntu/pool/main/s/session-migration/session-migration_0.3.5_amd64.deb HTTP/1.1 GET /ubuntu/pool/main/g/gno…
Wireshark · Packet 97156 · wsl2test.pcapng
> Frame 97156: 670 bytes on wire (5360 bits), 670 bytes captured (5360 bits) on interface \Device\NPF_{081336C2-443C-4A7A-B5AE-35A0BD03E7AB}, id 3
> Ethernet II, Src: IntelCor_02:39:ba (50:e0:85:02:39:ba), Dst: IntenoBr_cc:77:70 (20:1f:31:cc:77:70)
> Internet Protocol Version 4, Src: 192.168.1.132, Dst: 91.189.88.152
> Transmission Control Protocol, Src Port: 49902, Dst Port: 80, Seq: 11053, Ack: 22308258, Len: 604
> Hypertext Transfer Protocol
> Hypertext Transfer Protocol
> Hypertext Transfer Protocol
v Hypertext Transfer Protocol
  > GET /ubuntu/pool/main/g/gedit/gedit-common_3.36.2-0ubuntu1_all.deb HTTP/1.1\r\n
```

Appendix 9 – NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\UFH\SHC

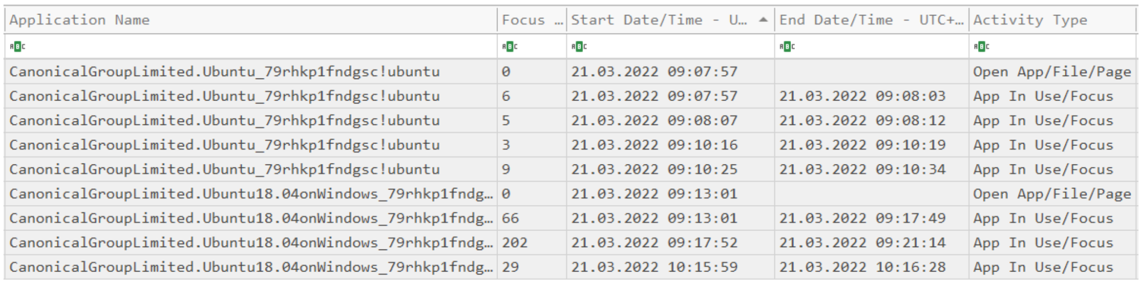
NTUSER.DAT registry hive Software\Microsoft\Windows\CurrentVersion\UFH\SHC contains data about graphical applications, not just these that were installed by a user, but also additional components like language support, ibus-setup, advanced network configuration (nm-connection-editor), printers (system.config-printer), these components were installed automatically.



Data
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Text Editor (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu gedit
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\IBus Preferences (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu ibus-setup
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Printers (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu system-config-printer
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Text Editor (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu gedit
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Language Support (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu /usr/bin/gnome-language-selector
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Printers (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu system-config-printer
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Advanced Network Configuration (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu nm-connection-editor
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Firefox Web Browser (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu firefox
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Advanced Network Configuration (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu nm-connection-editor
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Firefox Web Browser (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu firefox
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\IBus Preferences (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu ibus-setup
C:\Users\Thesis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Ubuntu\Language Support (Ubuntu).lnk C:\Windows\System32\wslg.exe ~ -d Ubuntu /usr/bin/gnome-language-selector

Appendix 10 - windows timeline activity, WSL1

Windows Timeline Activity showed the time spent on WSL and Ubuntu, when was the application opened and closed.



Application Name	Focus ...	Start Date/Time - U...	End Date/Time - UTC+...	Activity Type
CanonicalGroupLimited.Ubuntu_79rhkp1fndgsc!ubuntu	0	21.03.2022 09:07:57		Open App/File/Page
CanonicalGroupLimited.Ubuntu_79rhkp1fndgsc!ubuntu	6	21.03.2022 09:07:57	21.03.2022 09:08:03	App In Use/Focus
CanonicalGroupLimited.Ubuntu_79rhkp1fndgsc!ubuntu	5	21.03.2022 09:08:07	21.03.2022 09:08:12	App In Use/Focus
CanonicalGroupLimited.Ubuntu_79rhkp1fndgsc!ubuntu	3	21.03.2022 09:10:16	21.03.2022 09:10:19	App In Use/Focus
CanonicalGroupLimited.Ubuntu_79rhkp1fndgsc!ubuntu	9	21.03.2022 09:10:25	21.03.2022 09:10:34	App In Use/Focus
CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndg...	0	21.03.2022 09:13:01		Open App/File/Page
CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndg...	66	21.03.2022 09:13:01	21.03.2022 09:17:49	App In Use/Focus
CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndg...	202	21.03.2022 09:17:52	21.03.2022 09:21:14	App In Use/Focus
CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndg...	29	21.03.2022 10:15:59	21.03.2022 10:16:28	App In Use/Focus

Appendi 11 - Registry summary

All registry entries that were found in examination.

Table 6. Registry Summary

Registry	WSL2	WSL1
Amcache.hve: Root\InventoryApplicationFile\wsl.exe|88d5e8ee009f35b9	x	x
SOFTWARE: Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products\E752FE635D127F4448157029A3C864E5\InstallProperties	x	-
SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\Folder	x	-
Amcache.hve: Root\InventoryApplication\00007c914b0dd2764fd645ed6e2c9620551900000904	x	x
Amcache.hve: Root\InventoryApplicationFile\wslhost.exe|e4351ae5a193337	x	x
UsrClass.dat\Local Settings\Software\Microsoft\Windows\Shell\MuiCache	x	-
SOFTWARE: Microsoft\Windows\CurrentVersion\AppModel\StagingInfo\CanonicalGroupLimited.UbuntuonWindows_2004.2020.424.0_x64__79rhkp1fndgsc	x	-
SOFTWARE: Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\PackageRepository\Packages\CanonicalGroupLimited.UbuntuonWindows_2004.2020.424.0_x64__79rhkp1fndgsc	x	x
UsrClass.dat: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppContainer\Mappings\S-1-15-2-202137915-1588483389-988967374-857029487-3114683470-1999124116-284053513	x	x
UsrClass.dat: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Packages\CanonicalGroupLimited.UbuntuonWindows_2004.2020.424.0_x64__79rhkp1fndgsc	x	x
NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\App Paths\ubuntu.exe	x	x
NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\Lxss\{17f1eb54-04fa-48ab-98d6-a654d1f70f47}	x	x

Registry	**WSL2**	**WSL1**
SYSTEM: ControlSet001\Services\bam\State\UserSettings\S-1-5-21-2705457751-833678465-653486118-1001	x	x
SYSTEM: ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\RestrictedServices\AppIso\FirewallRules	x	x
NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\UFH\SHC	x	-
SOFTWARE: Classes\Directory\(background)\shell\WSL		x
SOFTWARE: Microsoft\Windows\CurrentVersion\Explorer\IdListAliasTranslations\WSL		x
SOFTWARE: Classes\CLSID\{615a13be-241d-48b1-89b0-8e1d40ffd287}		x
SOFTWARE: Classes\CLSID\{B2B4A4D1-2754-4140-A2EB-9A76D9D7CDC6}\Instance\InitPropertyBag		x
SOFTWARE: Classes\Directory [or \Drive\]\shell\WSL\command		x
SYSTEM: ControlSet001\Control\Session Manager\AppCompatCache		x
NTUSER.DAT: Software\Microsoft\Windows\CurrentVersion\Lxss\{4c42a344-8010-42b7-8aa7-d2595c169358}		x