TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Helena Ingermann 212001IVCM

Windows Subsystem for Android – Forensic Analysis

Master's thesis

Supervisor: Shaymaa Mamdouh Khalil MSc

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Helena Ingermann 212001IVCM

Windows Subsystem for Android - kriminalistiline analüüs

Magistritöö

Juhendaja: Shaymaa Mamdouh Khalil MSc

Tallinn 2023

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: Helena Ingermann

15.05.2023

Abstract

Microsoft introduced the Windows Subsystem for Android (WSA) in May 2021, with the goal to integrate Android application usage into a desktop environment. This Subsystem facilitates the seamless integration of Android applications with the Windows 11 operating system, bridging the gap between mobile and desktop experiences.

However, the rapid integration of Windows Subsystem for Android (WSA) into the Windows 11 operating system introduces new challenges for digital forensic investigators. As the number of users employing Android applications on their desktop environments grows, it becomes increasingly vital for digital forensic specialists to understand the ramifications of such a subsystem and the artifacts it produces within the host system.

This research addresses these challenges and provides a foundation for digital forensic investigators, ensuring they have the knowledge and resources to examine WSA-related artifacts and know of WSA's potential attack surfaces thoroughly and accurately.

A controlled experiment was conducted to facilitate a comprehensive overview of WSA as a subsystem including the essential artifacts associated with it. The research covers WSA installation artifacts, application execution data, file operations, log information, and a detailed examination of WSA from a network perspective. Due to structural similarities, WSA and Windows Subsystem for Linux 2 (WSL2) were compared.

In conclusion, the study highlights potential attack surfaces and identifies gaps in knowledge that warrant further research to enhance the understanding of the forensic implications of Windows Subsystem for Android in the Windows 11 operating system.

This thesis is written in English and is 75 pages long, including 7 chapters, 23 figures and 7 tables.

Annotatsioon

Windows Subsystem for Android kriminalistiline analüüs

Microsoft tutvustas 2021. aasta mais Windows Subsystem for Android (WSA) alamsüsteemi, mille eesmärk on hõlbustada Androidi rakenduste integreerimist Windowsi töölauakeskkonda. WSA alamsüsteem soodustab Androidi rakenduste probleemivaba integreerimist Windows 11 operatsioonisüsteemis, ühendades mobiilsete ja töölauarakenduste maailmad.

WSA aktiivne arendus Windows 11 operatsioonisüsteemiga toob paraku kaasa uusi väljakutseid digitaalkriminalistika uurijatele. Kuna Androidi rakenduste kasutajate arv töölauakeskkonnas kasvab, muutub digitaalkriminalistika spetsialistide jaoks üha olulisemaks mõista sellise alamsüsteemi kasutamise potentsiaalseid tagajärgi ning digitaalseid tõendeid, mis jäetakse host süsteemi.

Käesoleva lõputöö eesmärk on käsitleda WSA digitaalekspertiisi väljakutseid, pakkudes digitaalkriminalistika spetsialistidele teoreetilisi aluseid, ressursse ja vahendeid WSA-d puudutava põhjaliku ja täpse digitaal ekspertiisi läbiviimiseks ning teadlikkuse tõstmiseks WSA potentsiaalsete ründepindade osas.

Magistritöö raames viidi läbi kontrollitud eksperiment, et saavutada süstemaatiline ja põhjalik arusaam WSA alamsüsteemist ning sellega seotud digitaalsetest jälgedest. Eksperiment keskendus WSA installeerimise, rakenduste käivitamise, failitoimingute ja logiandmete digitaalsetele jälgedele ning WSA võrgutegevuse üksikasjalikule analüüsile. Arvestades struktuurilisi sarnasusi, võrreldi WSA-d Windows Subsystem for Linux 2 (WSL2) alamsüsteemiga. Lõputöö toob välja potentsiaalsed ründealad ja tuvastab teadmislüngad, mis nõuavad täiendavat uurimist, et paremini mõista Windows 11 operatsioonisüsteemile paigaldatud WSA digitaalsete jälgede mõju.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 75 leheküljel, 7 peatükki, 23 joonist, 7 tabelit.

List of abbreviations and terms

ADB	Android Debug Bridge
AOSP	Android Open Source Project
BAM	Background Activity Moderator
COM	Component Object Mode
DAM	Desktop Activity Moderator
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
MRU	Most Recently Used
MB	Megabyte
NAT	Network Address Translation
NetBIOS	Network Basic Input/Output System
NVMe	Nonvolatile Memory Express
OS	Operating System
RID	Relative Identifier
RPC	Remote Procedure Call
SID	Security Identifier
SMB	Server Message Block
SSD	Solid State Drive
TLS	Transport Layer Security
TTL	Time To Live
UEFI	Unified Extensible Firmware Interface
VHDX	Virtual Hard Disk Image
VM	Virtual Machine
VPN	Virtual Private Network
WSA	Windows Subsystem for Android

WSL	Windows Subsystem for Linux

Table of contents

List of figures

List of tables

1 Introduction

The popularity of Android as a mobile operating system and the expansion of mobile devices have led to a growing demand for solutions to integrate mobile applications into desktop environments seamlessly. Microsoft developed a technology called Windows Subsystem for Android (WSA), a compatibility layer providing the capability to run Android applications natively on Windows 11 devices to address this need on the market [1]. By leveraging virtualization and containerization technologies, WSA enables users to access their Android applications directly from their Windows 11 devices.

Similarly, to Windows Subsystem for Linux 2 (WSL2) [2], WSA utilizes same Hyper-V virtualization technology to create a virtualized environment where Android applications can run alongside Windows applications without needing a separate emulator or virtual machine. With the integration of essential Android operating system components and the Linux kernel, WSA offers users a seamless and intuitive experience, allowing access to Android applications as if they were native Windows applications.

The first announcement of the Subsystem was in May 2021 at a Microsoft Build developer conference with the goal of users accessing Android apps without switching devices. In March 2022, the first stable release of WSA became accessible in 31 markets. Since then, WSA has caused significant interest in end-users and developers even though the technology is still in active development.

Bridging a gap between two significant ecosystems may also raise potential malicious avenues for the threat actors to exploit. The literature review conducted for this thesis showed how limited research and documentation are available about subsystems and their forensic implications, making it a compelling area to research further. According to statistics, there are over 2.8 billion active Android users [3], and based on 2022 third-quarter statistics, 438,328 applications are available in Amazon Appstore [4]. WSA enables developers and end-users to run such applications in their Windows 11 operating system, raising new risks and potential attack vectors for Windows users. Understanding aspects such as the WSA architecture, its integration with Windows 11 operating system, potential attack surfaces, changes WSA makes to the registry, and documenting the artifacts left on the host system are crucial to assist digital forensic experts in future investigations.

1.1 Research Objectives

The primary focus of this thesis is to document the digital artifacts left behind by Windows Subsystem for Android (WSA) on the Windows 11 operating system. The thesis aims to answer two main questions:

Firstly, what artifacts are left behind by WSA, and where can they be found? The goal is to document the digital traces WSA leaves on the host system by providing insights into the components and processes that make up WSA. These artifacts may include registry entries, files, logs, and other digital remnants that can be used to reconstruct user activities or trace malicious behaviour.

Secondly, can digital artifacts found in WSA be compared with artifacts found in WSL2? By conducting the comparison, this thesis seeks to identify if there exist any similarities between the artifacts on the Windows 11 system. The comparison will focus on artifacts such as registry entry key locations, event logs, and network traffic to identify unique features or challenges each subsystem poses.

It is important to note that the scope of the forensic analysis of WSA remains solely on analysing the WSA environment and its artifacts and its implications on the host system rather than the Android operating system; therefore, traditional Android forensic techniques and methods are out of scope of this experiment.

1.2 Novelty

Currently, there is a lack of publicly available research studies specifically focused on WSA. Although a few studies have been published on WSL2, which utilizes the same Hyper-V virtualization technology as WSA, a substantial gap in the current literature persists. The increasing usage of WSA amongst both end-users and developers and the growing importance of digital forensics, creates a need for comprehensive documentation and research on this emerging technology.

This research aims to address this gap in the literature by providing an exploration of the digital artifacts left behind by WSA on the host Windows operating system. Examining WSA's architecture, artifacts, and potential attack vectors, this study plans to contribute a foundation for digital forensic investigators in the light of the growing popularity of Windows Subsystems. This research seeks to support the potential future forensic investigations of WSA by providing valuable insights to facilitate more efficient and effective forensic analysis of WSA for investigators. By comparing results with WSL2, which has a similar architecture, this thesis aims to improve the understanding of Windows subsystem forensics.

1.3 Navigating the Thesis Structure

In this chapter, we will provide an overview of the subsequent chapters, outlining the structure and content of each to facilitate a clear understanding of the topics covered. The chapter serves as a roadmap for the reader, highlighting the aspects of each chapter.

Chapter 2 offers essential background information on the Windows Subsystem for Android (WSA) and provides an overview of its predecessor, the Windows Subsystem for Linux (WSL2). This chapter also includes a discussion of the windows and network forensic techniques employed in this research and an overview of the tools utilized throughout the thesis.

Chapter 3 delves into a comprehensive literature review, examining the existing state of the art related to the topic to establish a foundation for the current research. The review primarily focuses on WSL2 since, at the time of writing, there were no research papers related to WSA as to the author's knowledge. Other relevant studies are also discussed in this chapter.

Chapter 4 describes the research methodology employed in this study, providing an in-depth overview of the controlled experiment setup. This chapter includes an experiment setup diagram to facilitate comprehension.

Chapter 5 presents the results and findings of the experiment, offering key insights derived from the analysis. The analysis is based on the controlled experiment detailed in Chapter 4. This chapter also includes a comparison between WSA and WSL2

Chapter 6 provides a discussion of the findings, connecting them to the broader context of the literature and the research questions posed. This chapter explores the implications, limitations, and potential avenues for future research.

Finally, Chapter 7 concludes the study, summarizing the research's main findings, contributions, and implications.

2 Background

On the 15th of February 2022, Microsoft announced the release of the first public preview build (1.8.32837.0) of Windows Subsystem for Android (WSA). As of April 2023, the latest available build is 2302.4000, which has been running on Android 13 [5]. Before the emergence of WSA, Android emulators such as Bluestacks[1] filled the void in the market for running Android applications on Windows operating systems. Compared to Android emulators, WSA is unique on several occasions. It allows Android applications to run natively on Windows, whereas Android emulators are third-party software applications that simulate the Android operating system on a Windows device. In addition, WSA utilizes Hyper-V virtualization technology, offering a more seamless and integrated experience, with Android applications appearing and behaving like native Windows apps. WSA is developed and supported by Microsoft, ensuring high compatibility with Windows devices and applications. In contrast, Android emulators can vary significantly in compatibility and performance, with some emulators being better suited for specific devices or applications than others.

2.1 WSA

Windows Subsystem for Android (WSA) enables Android applications accessible through the Amazon Appstore to run on Windows 11 devices [1]. There are two options to install the WSA on the user's device: installing the Amazon Appstore, which will also install the WSA in the background, or using Microsoft Store for retrieving the Android application, and in turn, the Amazon Appstore will be installed as well. The official requirement for installing Android applications on WSA is to utilize the Amazon Appstore. However, the Amazon Appstore is supported only in 31 countries [6]. To operate WSA on their devices, users must meet specific minimum system requirements. These requirements include the Windows 11 operating system, a minimum of 8GB of

[1] https://www.businessinsider.com/guides/tech/what-is-bluestacks (accessed Mar. 25, 2023)

RAM, an SSD storage type, and an x64 or ARM64 processor architecture [8]. Additionally, virtualization must be enabled on the host device.

WSA operates on the Linux kernel and Android operating system derived from the Android Open Source Project (AOSP) [7]. Starting from January 2023, WSA runs on version 13 of the Android operating system [5]. The subsystem functions within a Hyper-V virtual machine, similar to the operating environment for Windows Subsystem for Linux 2 (WSL2) [7]. Using the Android OS and Linux kernel, WSA provides an environment where Android applications can be run natively on Windows 11 devices [7]. The Hyper-V virtual machine creates a containerized environment that facilitates the seamless integration of Android applications alongside Windows applications [7]. In a collaborative effort, Microsoft has partnered with Intel to utilize the Intel Bridge Technology to enable Arm-only apps to run AMD and Intel devices to cover a wide range of Windows process types [7].

It is vital to consider that WSA is still in the developmental stages and may be subject to changes and updates in the future. Using the Linux kernel and Android OS based on the Android Open Source Project (AOSP) allows for compatibility with a broad range of Android applications. At the same time, the Hyper-V virtual machine environment ensures the efficient and effective operation of WSA.

2.1.1 WSA VM lifecycle

Within the Windows Subsystem for Android (WSA), applications can operate in three different states, each of which is influenced by the user's actions [1]. The transition between these states is typically prompted by user interaction, such as launching an application or receiving a notification. When an application is minimized, it is either paused or stopped, depending on the current state. The three distinct states are related to the virtual machine (VM) lifecycle and can be described in Figure 1 [1]:

1. Running
2. Lightweight Doze: Activated after no app activity for 3 minutes. Deactivated by user activity or an app notification.
3. Not Running: Activated after no app activity for 7 minutes.

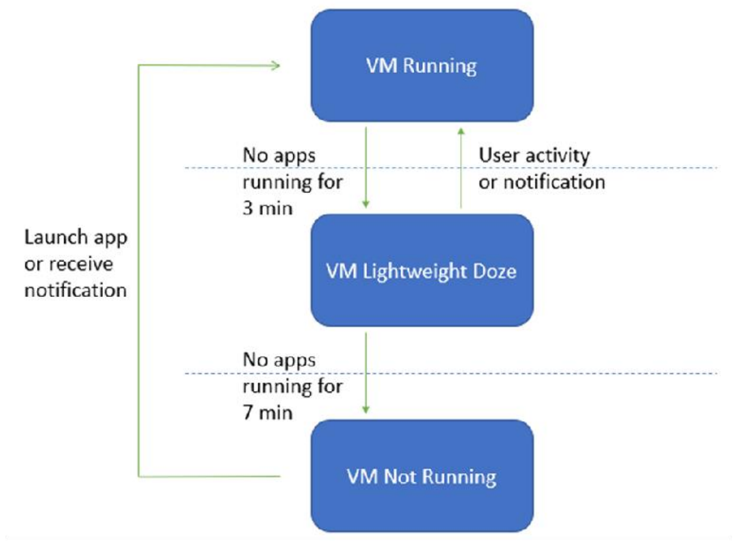

VM Running
No apps running for 3 min
User activity or notification
Launch app or receive notification
VM Lightweight Doze
No apps running for 7 min
VM Not Running

Figure 1. VM lifecycle considerations [1].

2.1.2 WSA Usage in Countries without Amazon Appstore Support

The rapid expansion of Windows Subsystem for Android's popularity has generated significant interest among end-users and developers, extending beyond the currently supported countries. Therefore, individuals seek alternative solutions for obtaining WSA, particularly in countries with restricted access to the Amazon Appstore. A comprehensive overview of these alternatives will be presented in the subsequent section.

One potential solution for users seeking to access Windows Subsystem for Android is manually installing the WSA using PowerShell. The required package can be obtained from a third-party website, allowing users to install the WSA on their Windows 11 device without using Microsoft Store and Amazon Appstore [9].

However, to install Android applications using WSA, users must find alternative sources for obtaining applications—one of the options is to manually sideload the desired applications, which eventually can get cumbersome. Another approach is to use Aurora Store, an open-source third-party app store that allows users to download Android applications directly onto their devices [10].

In addition to the previously mentioned methods of manually sideloading apps or using the Aurora Store[1], a graphical user interface (GUI) based open-source package manager utility called WSA Pacman offers a centralized platform solution for users, making managing and updating apps more convenient [11]. WSA Pacman can be obtained from its dedicated GitHub page and requires the WSA to run with developer and debugging modes enabled.

Utilizing WSA Pacman, a prevalent use case for users, is to obtain the desired applications from APKMirror, a renowned third-party website recognized for its vast assortment of applications. APKMirror asserts that it verifies the authenticity of each application prior to making it accessible for download, thereby ensuring the safety and integrity of the apps hosted on its platform [12] .

Even though there are numerous ways to use the WSA, it is crucial to exercise caution when downloading apps from third-party websites like APKMirror or using before mentioned alternative solutions for WSA to function. They may present potential security risks, and Microsoft does not officially support these solutions.

2.1.3 Use Cases of WSA amongst Users

As mentioned, WSA offers users a convenient experience to explore their favourite Android applications in Windows 11. In order to gain more insight into the use and the purposes for which the users employ WSA, a short overview follows to summarize some usual use cases based on available information on public forums. Upon investigating forum channel discussions on WSA's usefulness, it is evident that many advantages are afforded to end-users and developers through utilizing WSA.

From a developer's standpoint, WSA broadens the potential market by facilitating access to new user segments by creating applications tailored for Windows devices. Moreover, WSA streamlines the app testing process by enabling developers to capitalize on their pre-existing expertise in Android app development and repurpose their code and tools, resulting in decreased development time and effort [13].

[1] https://gitlab.com/AuroraOSS/AuroraStore (accessed Mar. 28, 2023)

End-users employ WSA for various purposes, including engaging in gaming experiences, managing IoT devices, accessing preferred social media platforms, and exploring various music services, among other applications [13]. This list highlights the versatility and adaptability of WSA in catering to a wide array of user needs and preferences.

It is imperative to note that the information outlined in this section is vital from the authors' perspective to understand better how the users use WSA and, therefore, can help support conducting forensic examinations more efficiently.

2.2 WSL2

Windows Subsystem for Linux 2 (WSL2) represents an advanced iteration of its predecessor, WSL1, and facilitates the execution of ELF64 Linux binaries on Windows systems. WSL2 operates on a complete Linux kernel, ensuring comprehensive system call compatibility [14]. Notably, WSL2 employs Hyper-V lightweight virtualization technology, which also serves as the foundation for the Windows Subsystem for Android (WSA).

Relative to WSA, the system requirements for WSL2 are notably less demanding. To operate WSL2, the minimum prerequisite is Windows 10, Version 1903 (Build 18362 or later), or Windows 11 [14]. In contrast, WSA is exclusively available for Windows 11, reflecting a higher threshold regarding system requirements.

WSL2 additionally enables the utilization of Linux GUI applications. Concerning system requirements, the host system needs to be on a minimum of Windows 10 Build 19044 and above or Windows 11 [15]. These applications operate like native Windows applications and can be initiated from the Start menu [15], exhibiting similarity to the functionality of WSA.

2.3 Windows Forensics

The experiment carried out in this research primarily relies on Windows and network forensic techniques. This section provides a more general overview of Windows registry hives and other valuable datapoints. Subsequent chapter, specifically the Chapter 5 - Analysis and Results, will delve into how each artifact contributes to the investigative process outlined in this thesis.

The Windows Registry is composed of both live and offline hives. These hives store crucial configuration information about the operating system, integrated applications, installed hardware, and user accounts [16]. For the forensic investigation in this thesis, the analysis focuses on offline database files, commonly referred to as hives. The overview of the offline hives is in the Table 1 [16]:

Table 1. Registry Hives [16].

Registry Hive	**Nickname**	**Offline file**
HKEY_LOCAL_MACHINE\SAM	HKLM\SAM	SAM
HKEY_LOCAL_MACHINE\Security	HKLM\Security	SECURITY
HKEY_LOCAL_MACHINE\System	HKLM\System	SYSTEM
HKEY_LOCAL_MACHINE\Software	HKLM\Software	SOFTWARE
HKEY_CURRENT_USER	HKCU	NTUSER.DAT, UsrClass.dat

The SAM, SYSTEM, SOFTWARE, and SECURITY hives are in the path C:\Windows\System32\config\. The SAM hive contains user-related information that can aid in investigating user activities. When profiling a user, it is essential to note the Relative Identifier (RID) found in the SAM hive. Numerous other artifacts, such as the Recycle Bin, Background Activity Moderator (BAM), and Event Logs, also employ RIDs [16]. Hence, recording the RID early in the investigation can enable the correlation of other potentially vital pieces of evidence.

The SYSTEM database file stores information such as the computer's hostname, essential for analysing log files or network connections and the system's configuration settings, including service and device driver configurations [16]. Information under SYSTEM\CurrentControlSet\Services can help determine whether a service application initiates at boot [16]. Furthermore, the SYSTEM hive contains subkeys for the BAM and Desktop Activity Moderator (DAM). BAM logs the full path of an executable, along with the date and time of its last execution, providing critical information about executed programs [16]. The User Security Identifier (SID) links BAM and DAM subkeys to users. The ShimCache, located within the registry key SYSTEM\ControlSet\Control\Session Manager\AppCompatCache, is a critical artifact in identifying executables executed on a system [17]. ShimCache entries are typically populated following a system reboot or shutdown [17].

The SOFTWARE hive retains configuration data about installed applications. For example, the Microsoft\Windows\CurrentVersion\CapabilityAccessManager has information about applications that control the system's microphone, camera, and location [16]. Furthermore, SOFTWARE\Microsoft\Windows NT\CurrentVersion contains data regarding the operating system's version, type, build number, and other details [16]. The SOFTWARE database file also assists in identifying network profiles and their initial and most recent connection times.

The SECURITY hive holds security-related information for the system, containing data on user and group Security Identifiers (SIDs) and system audit policies [16]. Its primary responsibility is maintaining settings and configurations about access control and user permissions.

In addition to system hives, each user has a dedicated registry hive located at C:\Users\<username>\NTUSER.dat, which contains user account-related configurations, application execution data, file and folder interactions, cloud storage settings, and information about internet activities [16]. For instance, the Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist key within NTUSER.dat can provide a wealth of details regarding GUI programs initiated through Windows Explorer, such as [16]:

- Last Run Time
- Run Count
- Name of the GUI application
- Focus Time
- Focus Count

Another user-related hive can be found at C:\Users\<username>\AppData\Local\Microsoft\Windows\UsrClass.dat, a virtualized registry root supporting the User Account Control (UAC) feature [16]. This hive also contains information about program execution and the folders that the user opened or closed.

Prefetch files, situated in C:\Windows\Prefetch, provide valuable information about the execution history of applications. These files reveal the executable's name, the first and last execution times, and the execution count. A Prefetch file is generated after an application is executed for the first time and can store data from up to eight execution instances. If multiple Prefetch files share the same executable name, it implies that the executable was run from different locations. It is crucial to recognize that the presence of a Prefetch file does not necessarily indicate successful execution; thus, it is essential to corroborate this information using additional artifacts that provide other execution evidence [16].

The Amcache hive, located at C:\Windows\appcompat\Programs\Amcache.hve, is a valuable repository of program execution data [18]. It contains information on file execution paths, installation executions, SHA1 hashes, and deletion times. Amcache is critical in analyzing portable programs, external storage devices, and anti-forensic tools during digital forensic investigations [18].

2.4 Network Forensics

Network forensics is a specialized domain of digital forensics, primarily concerned with the monitoring, capturing, and recording network activity for subsequent analysis [19]. This investigative process examines volatile and dynamic data, providing critical insights into a system's network behaviour and communication patterns [19].

In the present study, network forensics was employed to analyse the WSA from a network perspective. The open-source packet analysis tool, Wireshark [20], was utilized to achieve this objective. This powerful tool facilitates the real-time capture and examination of network traffic.

2.5 Tools

The selection of tools for this research was informed by the author's prior experience with various forensic utilities that have demonstrated their reliability and suitability in conducting similar experiments. Many of these tools have been developed and endorsed by renowned forensic specialists. By leveraging the author's expertise and familiarity with these tools, the experiment sought to ensure the results' accuracy and validity while benefiting from the well-established reputation and credibility of the chosen forensic applications.

2.5.1 CAINE (Computer Aided INvestigative Environment)

CAINE (Computer Aided INvestigative Environment) is an open-source forensic operating system with a graphical user interface (GUI) [21]. It was selected for this research due to its robust write-blocking methodology, which ensures the creation of forensically sound disk images. By default, CAINE mounts all disks in Read-Only mode [21], thereby preserving the integrity of the data during the forensic examination and preventing any inadvertent modifications to the actual evidence. This characteristic makes CAINE ideal for conducting a reliable and comprehensive forensic investigation.

2.5.2 FTK Imager

FTK Imager is a versatile imaging and data previewing tool[1] commonly employed in manual forensic investigations. For this experiment, the command-line version of FTK Imager was utilized to acquire disk images.

2.5.3 Arsenal Image Mounter

Arsenal Image Mounter is a sophisticated image mounting tool widely used in digital forensics. It enables the mounting of forensic images as drives or physical devices in read-only mode, preserving the integrity of the evidence [22]. The tool supports various image formats, including RAW/DD, E01, S01, AD1, and L01 [16]. By mounting images, investigators can interact with files using their native applications, run antivirus and

[1] https://www.exterro.com/ftk-imager (accessed Apr. 12, 2023)

malware detection tools on the mounted image, and maintain the forensic soundness of the image file due to its read-only capabilities.

2.5.4 KAPE

KAPE is a versatile triage imaging tool typically employed to collect critical evidence before conducting a full disk imaging process. It serves two primary functions: gathering files from various sources and processing the collected files using one or more programs[1]. KAPE can search mounted E01 images, live system drives, and specific paths, with the ability to output the collected targets as a virtual hard disk image (VHDX).

The Target configuration contains a list of file masks that facilitate identifying and collecting different items from the storage device [23]. On the other hand, the Modules configuration comprises information about the programs to be executed, which can be run against a live system or files collected by one or more target configurations [23].

In the context of this research, KAPE was utilized to target key artifacts that would support the investigation, focusing on the disk image previously acquired using CAINE. Module execution was carried out on the VHDX generated by the target collection.

2.5.5 Registry Explorer

Registry Explorer is an open-source registry viewer designed for digital forensics purposes. It offers the ability to view registry hives, parse registry keys using plugins, display deleted keys, and explore unallocated space within a hive [16]. One of its distinguishing features is the capacity to load multiple hives simultaneously, allowing for efficient searches across several hives at once.

2.5.6 MFTECmd.exe

MFTECmd.exe is a sophisticated command-line tool developed by Eric Zimmerman for parsing NTFS file systems in the context of digital forensics [24] . By delving into the MFT, forensic analysts can gain a wealth of information about files, including their creation, modification, access times, file ownership, and other essential attributes. The

[1] https://www.kroll.com/en/services/cyber-risk/incident-response-litigation-support/kroll-artifact-parser-extractor-kape (accessed Apr. 12, 2023)

tool's capabilities extend to parsing various MFT attributes, reconstructing file paths, and exporting the results in various output formats, such as CSV and JSON [24].

2.5.7 PECmd.exe

PECmd.exe is an open-source tool Eric Zimmerman created for parsing Prefetch files. Prefetch files are a crucial component of the Windows operating system, responsible for improving application launch performance by caching essential information related to frequently executed programs.

The PECmd tool is highly versatile, allowing forensic analysts to process individual Prefetch files or an entire Prefetch directory, depending on the scope of the investigation [25]. In addition, PECmd can output processed Prefetch data in CSV and JSON formats [25]. By analyzing these files, investigators can gather valuable insights into the applications used, their execution times, and their usage frequency, contributing to a more comprehensive understanding of user activities.

2.5.8 Event Log Explorer

Event Log Explorer is an advanced Windows event log analysis tool to enhance the process of examining event logs. This software is compatible with the Windows "legacy" format, EVT, and the newer EVTX format logs [26].

The utility has powerful filtering capabilities, enabling forensic analysts to identify specific events or patterns of interest quickly and accurately. Furthermore, Event Log Explorer simplifies generating and exporting of event log reports [26].

2.5.9 LECmd.exe

LECmd.exe is a specialized tool for parsing shortcut files, commonly known as LNK files. These files are typically generated within the directory C:\Users\<user>\AppData\Roaming\Microsoft\Windows\Recent, providing valuable information regarding user activity and file access patterns.

2.5.10 Regshot

Regshot is an open-source utility that facilitates the task of comparing modifications made to the registry. This tool can capture snapshots of the registry in two states, enabling a comparative analysis between them. The tool offers the option to generate the output in either "text" or "HTML" format, allowing users to choose the format that best suits their needs[1].

The present thesis leveraged the capabilities of Regshot to conduct a comparative analysis of the first and second disk images obtained during the investigation. Specifically, Regshot was utilized to determine any alterations made to the registry or file system due to the WSA and WSA Pacman installation.

2.5.11 Timeline Explorer

Timeline Explorer is a graphical user interface-based tool designed as an alternative to Excel for digital forensic investigators to perform their analysis. It offers advanced filtering options, timestamp formatting, search functions, and conditional formatting [2]. The tool was utilized in this master's thesis to analyse Shimcache, Amcache, Prefetch, $M, and $J, and its features helped to identify and visualize relevant information for the investigation efficiently.

[1] https://github.com/Seabreg/Regshot (accessed: May 11, 2023)

[2] https://aboutdfir.com/toolsandartifacts/windows/timeline-explorer/ (accessed May 11, 2023)

3 Related Work

The analysis of related studies concentrates on research supporting the WSA investigation, encompassing potential locations of artifacts, appropriate tools for analysis, and other factors. The primary emphasis of the research and articles in this chapter is on WSL2, as WSA is also based on the same virtualization technology. The literature review incorporates papers on Android, highlighting potential threats and security concerns associated with these platforms, as well as a paper discussing Hyper-V virtualization technology. To the author's knowledge, no WSA-related papers have been published at the time of writing, indicating a gap in the current state of the art.

In 2022, Boigner. and Luh, published a paper, "WSL2 Forensics: Detection, Analysis & Revirtualization [27]". The authors proposed a hypothesis to detect WSL artifacts based on Windows artifact categories such as Windows Registry, Prefetch, Jumplist, and AmCache. They used open-source tools like RegRipper, WinPrefetchView, JumpListView, and Windows internal Event Viewer. The author's approach is experiment-driven by first determining the artifact categories for the WSL installation, setting up necessary testing environments for the acquisition, and further analysis. As a result, the previously placed evidence, such as documents, images, and archived files, can be extracted, proving their proposed hypothesis. The paper did not cover memory analysis which might be necessary for future investigations.

The "Forensic Analysis of Windows Subsystem for Linux on Windows 11" thesis employed an experimental approach, concentrating on artifacts remaining on the host system [28]. The author conducted basic user activities in the experimental environment, captured network traffic, and obtained disk images. The investigation primarily focused on Windows and network forensics, excluding memory forensics. Most utilized tools were open source, endorsed by SANS Institute instructors, and a single commercial tool Magnet Axiom. The study involved a comparison between WSL1 and WSL2. The author ultimately determined that examining multiple artifacts is essential to associate them with WSL and emphasized the potential benefit of incorporating memory forensics for more valuable evidence.

Article [29] highlighted the potential risk of WSL2 inadvertently exposing internet traffic by circumventing the Windows Filtering Platform (WFP) layers [29]. WSL2 relies on Hyper-V virtual networking, and the Hyper-V Virtual Ethernet Adapter transmits traffic without the host firewall checking the packets on the host machine. Within the OSI layer two, the NATed packets appear exclusively as Ethernet frames [29]. The article claims this concern persists even when using VPN software [29]. Although VPN software developers assert, they are addressing the issue [29], no subsequent literature confirms whether the problem has been resolved.

During the Digital Forensic Research Conference, Asif Matadar delivered a presentation entitled "Investigating WSL Endpoints," primarily concentrating on WSL2 [30]. Matadar executed nine experiments to assess potential attack methods targeting WSL2. The objective was to identify forensic artifacts of interest from the Digital Forensics and Incident Response perspective. According to the presenter, these techniques represent a limited selection of potential attack vectors within WSL2. Given the growing attention from threat actors directed toward WSL2, it is reasonable to anticipate a possible increase in interest concerning WSA as well.

Paper [31] conducted a comparative analysis of three virtualization technologies in her paper "Comparison between common virtualization solutions: VMware Workstation, Hyper-V and Docker". The findings indicated that Hyper-V outperformed its counterparts in most domains, surpassing VMware, while Docker demonstrated the most optimal performance among the three [31]. The paper's thorough examination of Hyper-V architecture contributes to understanding the underlying technology employed in WSA.

Allix et al. emphasized the need for additional research concerning Android malware from a forensic perspective [32]. Their study involved obtaining numerous applications and utilizing customized crawlers in marketplaces to analyse application content. The authors determined that the design of Android applications enables the extraction of essential artifacts, thereby facilitating a deeper understanding of application developers. Although Android malware forensics falls outside the purview of this thesis, the paper offers valuable background knowledge to support the investigation.

In 2020, a paper titled "Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems" examined the progression of Android application development, addressing the associated security concerns [33]. The authors discussed emerging security frameworks and systems specific to the Android platform. As emphasized in the paper, Android applications present security risks that warrant consideration in the context of WSA running in Windows 11 operating system.

Kanchhal and Murugaanandam gave an exhaustive overview of Android architecture, how malware spreads, and how to detect it [34]. For their experiment, they developed an Android malware application and injected it into an Android device to show how data from a victim's device may be captured. They collected the dataset they produced in their experiment and used it to develop a malware detection model using machine learning, which proved to be 99.37 percent accurate. They plan to include deep learning and other results optimization in their future research.

As stated at the beginning of the literature review, the author sought research papers that would offer valuable contextual insights to facilitate the examination of the selected thesis topic. Given the lack of research directly associated with WSA, the author relied upon the existing state of the art surrounding WSL2 and will lean on Viitmaa's [28] thesis to perform the comparison required for the previously mentioned problem statement. Papers examining Android and a single study on Hyper-V serve as essential background knowledge, facilitating the effective conduct of the research. The Android-related papers provide insights into potential attack vectors associated with using the Android operating system, thereby enhancing the understanding of the relevant security concerns. Meanwhile, the paper on Hyper-V offers a deeper comprehension of the underlying technology upon which WSA is built, further strengthening the author's knowledge to conduct her experiment.

4 Methodology

This research utilizes a controlled-experiment methodology to explore the subject matter in-depth. The following chapter presents a comprehensive outline of this thesis's experimental setup, including the specifics of the disk acquisition process. The totality of the controlled experiment is illustrated in Figure 2. This figure provides a comprehensive graphical representation of the experiment.

4.1 Roadmap for the Controlled Experiment

The experiment contains five stages, each with specific objectives and tasks.

- Stage 1 involves installing the Windows 11 operating system and Sysmon, followed by acquiring the first disk image. This image will be compared with the Stage 2 disk image to facilitate registry analysis in Section 5.2 - Registry Evidence.

- Stage 2 entails the necessary steps for setting up WSA. The second disk image is acquired and will be used in conjunction with the first disk image to support registry analysis in Section 5.2 - Registry Evidence and Section 5.4 - Application Artifacts.

- Stage 3 focuses on installing applications that assist in conducting the required tests for Stage 4. The resulting artifacts related to application installation are discussed in Section 5.2 - Registry Evidence and Section 5.4 - Application Artifacts.

- Stage 4 consists of two file operation tests. Test 1 examines the possibility of transferring files between the host and WSA while also assessing the forensic implications of these movements. Test 2 involves downloading files from the WSA Firefox application and the host Firefox to evaluate the effectiveness of Microsoft Defender. Firefox was chosen based on the author's personal preference, as the outcome of Test 2 is not dependent on the specific browser used. The results of Stage 4 are covered in Section 5.5 - File Operations and Artifacts.

- Stage 5 aims to understand WSA from a network perspective. Tests 3, 4a, 4b, and 5 are described in more detail in Section 4.2 - Experiment Setup. The final disk image is acquired to facilitate forensic analysis of Stages 3 through 5.

Section 5.6 - Event Logs explores Sysmon and other relevant event logs that provide application download evidence and additional data that may support this research. A more detailed overview of the experimental setup is presented in Section 4.2 - Experiment Setup.

In Section 5.8 - Comparison Between WSA and WSL2, a comparative analysis is carried out, drawing upon the findings of the controlled experiment conducted in this research and the results of the thesis entitled "Forensic Analysis of Windows Subsystem for Linux on Windows 11” [28].

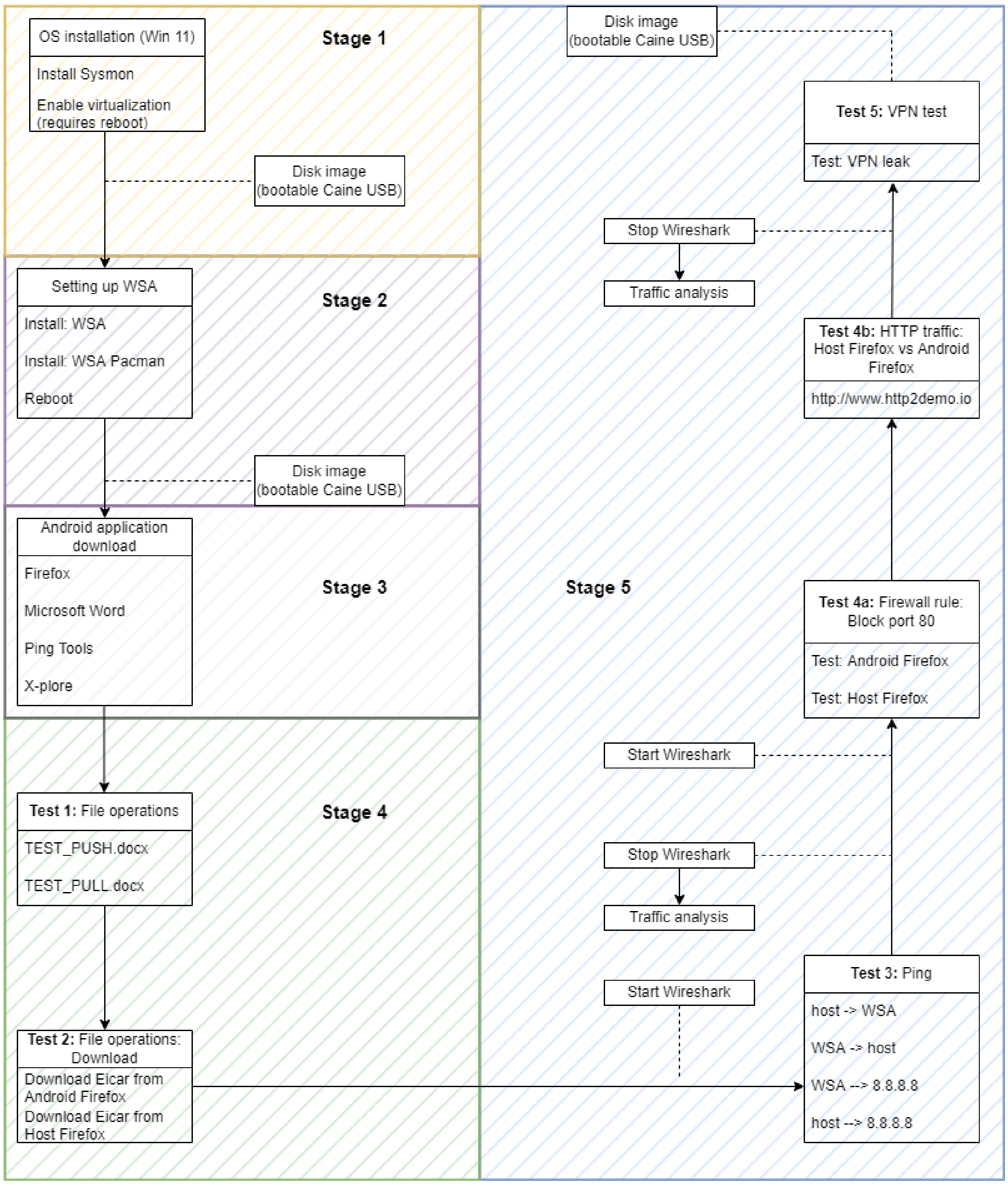


Figure 2. Experiment setup diagram.

4.2 Experiment Setup

The experiment was carried out in Estonia, a region not currently included in the supported areas for the Amazon Appstore. The author employed an alternative WSA installation solution outlined in Chapter 2. - Background as the basis for conducting the experiment. Table 2 presents the device and Windows specifications utilized in the controlled experiment.

Table 2. Device and Windows specifications.

Hostname	THESIS23
Processor	Intel(R) Core (TM) i7-8850H CPU @ 2.60GHz
Installed RAM	16.0 GB (15.7 GB usable)
Architecture	64-bit operating system, x64-based processor
OS Edition	Windows 11 Education
OS Version	22H2
OS build	22621.1413

4.2.1 Stage 1

The experiment began with the installation of the Windows 11 operating system. After that, Sysmon was installed to provide more in-depth security and system monitoring. WSA requires the activation of virtualization in Unified Extensible Firmware Interface (UEFI) settings and enabling Hyper-V, Virtual Machine Platform, and Windows Hypervisor Platform features within the Control Panel (Windows Features). After enabling these Windows features, a system reboot was needed to apply the changes effectively. The first disk image was acquired.

4.2.2 Stage 2

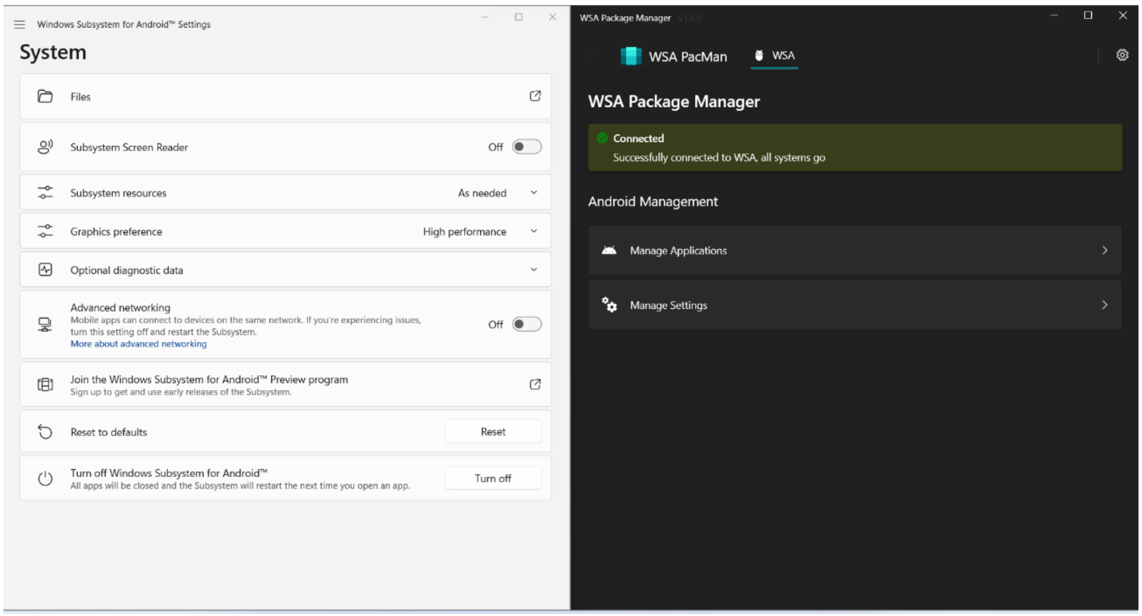
Following the initial disk acquisition, the WSA MSIX Windows app package was obtained from MS Store Generation Project[1] website, which allows downloading package directly from the Microsoft server[2]. This package was subsequently installed manually by employing PowerShell commands. In the next phase of the experiment, the most recent version of the GUI package manager, WSA Pacman, was installed. WSA Pacman was installed by downloading the necessary files from its dedicated GitHub repository[3]. Once the installation of WSA Pacman was completed, it was necessary to reboot the system to finalize the configuration and ensure the proper functioning of the installed components.

[1] https://store.rg-adguard.net/ (accessed Apr. 05, 2023)

[2] https://softwarekeep.com/help-center/is-storerg-adguardnet-safe-and-legal (accessed Apr. 05, 2023)

[3] https://github.com/alesimula/wsa_pacman (accessed: Apr. 08, 2023)

Upon rebooting the system, the WSA Client was launched. Developer mode, which also automatically enables USB debugging, was enabled to provide WSA Pacman with the necessary permissions to manage applications, sideload packages, and access specific system settings usually limited in a standard user environment. Alongside activating developer mode, Android Debug Bridge (ADB) debugging was authorized for WSA Pacman, enabling the installation and removal of applications, access to the filesystem, and the execution of commands in the Android environment [35]. The second disk image was acquired.


Windows Subsystem for Android™ Settings

System

Files

Subsystem Screen Reader — Off

Subsystem resources — As needed

Graphics preference — High performance

Optional diagnostic data

Advanced networking
Mobile apps can connect to devices on the same network. If you're experiencing issues, turn this setting off and restart the Subsystem.
More about advanced networking — Off

Join the Windows Subsystem for Android™ Preview program
Sign up to get and use early releases of the Subsystem.

Reset to defaults — Reset

Turn off Windows Subsystem for Android™
All apps will be closed and the Subsystem will restart the next time you open an app. — Turn off

WSA Package Manager v1.4.0

WSA PacMan | WSA

WSA Package Manager

Connected
Successfully connected to WSA, all systems go

Android Management

Manage Applications

Manage Settings

Figure 3. Visual representation of WSA Client and WSA Pacman GUI.

The steps mentioned above were necessary because the author could not access the official Amazon Store due to regional restrictions, needing to sideload the apps, and acquiring packages from alternative sources. Developers frequently employ sideloading to test applications before submitting them to official app stores.

4.2.3 Stage 3

After the second disk image, three applications - Firefox, Microsoft Word, X-plore, and Ping Tools - were downloaded from APKmirror. The Ping Tools application was chosen to conduct ping tests. To gain a deeper understanding of network connectivity between the host and WSA, Wireshark [1] and command-line tools such as netstat and ipconfig were used.

4.2.4 Stage 4 and Stage 5

The Firefox application was explicitly acquired to evaluate the effectiveness of the Firewall port 80 block rule. The results were compared with those of a previous experiment conducted on the WSL2, as documented in the "Forensic Analysis of Windows Subsystem for Linux on Windows 11" thesis [28]. Furthermore, the Firefox browser was utilized to generate HTTP traffic and to facilitate the Eicar test file download experiment. This experiment aimed to assess the ability of Windows Defender to identify and respond to potentially malicious files accessed through various channels, including the host Firefox browser and the WSA Firefox application.

The X-plore file manager application facilitated the investigation of file-sharing capabilities between the WSA and the host system. This examination aimed to understand the possibilities of file transfers between the two environments, assess the movement of files between the host and WSA, and analyse the implications of such transfers from a digital forensics' perspective.

EICAR test files[2] were downloaded utilizing the host Firefox browser and the Windows Subsystem for Android (WSA) Firefox browser. This test aimed to examine the functionality of Microsoft Windows Defender's detection capabilities when downloading files through a WSA application compared to the host system. After the completion of the file downloads, event logs were analysed to determine what forensic evidence could be found when a "malicious" file was downloaded from WSA.

To obtain further insights from a network forensic standpoint, the author visited a Hypertext Transfer Protocol (HTTP) site using both the host Firefox and Android application Firefox to obtain additional artifacts to differentiate between host traffic and WSA, such as User Agent, allowing for a more comprehensive understanding of the

[1] https://www.wireshark.org/ (accessed Apr. 05, 2023)

[2] https://www.eicar.org/download-anti-malware-testfile/ (accessed Apr. 07, 2023)

network behaviour of WSA, which could provide valuable insights for forensic investigations.

The article [29] published in 2020 investigated the potential leakage of internet traffic in WSL2. In order to determine whether WSA exhibits similar vulnerabilities, Surfshark Virtual Private Network (VPN) [1] was used as the testing tool. Within the Surfshark VPN settings, the kill switch feature was activated, ensuring that any potential IP leaks were mitigated by disabling the internet connection during rare disruptions. Additionally, the VPN was configured to operate in strict mode, safeguarding against internet connectivity in the event of VPN disconnections or interruptions. Enabling the kill switch and setting it to strict mode was required to see if it was possible to replicate the issue. Afterward, the third disk image was acquired.

4.3 Disk acquisition

The disk acquisition process was conducted by removing the NVMe[2] SSD from the target machine and placing it into an NVMe enclosure. Subsequently, the forensic investigation device was booted up using bootable USB flash drive with CAINE operating system, and the target machine's NVMe SSD was connected. The disk was acquired utilizing the FTK Imager command-line tool.

The acquisition process involved the creation of a forensically sound E01 image of the target disk, segmented into 2TB fragments, with no compression applied. The process included relevant metadata, such as case number, evidence number, description, examiner, and notes, and concluded with a verification process to ensure the integrity of the acquired image. The resulting disk image provided a solid foundation for conducting a thorough and reliable forensic investigation.

[1] https://surfshark.com/one (accessed Apr. 07, 2023)

[2] https://www.netapp.com/data-storage/nvme/what-is-nvme/ (accessed Apr. 15, 2023)

5 Analysis and Results

This chapter presents an examination of the forensic analysis and the findings obtained from the experiment setup outlined in Figure 2. This evaluation delves into the various aspects of the WSA environment and the interaction between the host system and the Android applications. The artifacts are discussed throughout this chapter, and comparisons are drawn to validate the findings.

The analysis conducted in this research involved comparing the first and second disk images to identify the differences in the registry and file system after the installation of Windows Subsystem for Android and WSA Pacman. The Regshot tool was employed to facilitate this comparison, which aided in capturing and comparing snapshots of the system's state.

The second and third disk images were used to conduct a comprehensive analysis as presented in Chapter 5 - Analysis and Results. This analysis involved examining various artifacts and data points to gain insights into the functioning and behaviour of WSA, its impact on the host system, and the presence of relevant forensic evidence.

By using multiple disk images and conducting thorough analysis, this research aimed to provide a thorough understanding of the changes and artifacts associated with the installation and operation of WSA, allowing for a more informed interpretation of the experiment findings and their implications.

5.1 WSA Virtual Hard Disk Evidence

The Windows Subsystem for Android (WSA) utilizes a virtual hard disk (VHDX) format for storing its data and operating system resources. Specifically, the WSA data is stored within two VHDX files located at the following paths:

- Users\<username>\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalCache\userdata.vhdx

- Users\<username>\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalCache\metadata.vhdx

Furthermore, the fully qualified path for the WSA service executable is:

- C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\WsaService\WsaService.exe

Additionally, Android maintains an application log within the WSA environment, which can be found at:

- \Users\<username>\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalState\diagnostics\logcat

During the analysis of the Regshot output, two essential files related to ADB (Android Debug Bridge) were discovered. These files, namely "adbkey" and "adbkey.pub," are for establishing a secure connection between the host system and the Windows Subsystem for Android.

The "adbkey" file represents the private key, while the "adbkey.pub" file corresponds to the public key. These keys are used for authentication, ensuring the integrity and security of the communication between the host and WSA.

The keys are stored in the following location:

- C:\Users\<username>\.android\adbkey
- C:\Users\<username>\.android\adbkey.pub

5.2 Registry Evidence

The following section examines the second disk image, containing the analysis of the NTUSER.dat, UsrClass.dat, SYSTEM, and SOFTWARE registry hives. This section provides insights into the most relevant artifacts from the author's perspective, equipping forensic examiners with essential key locations for conducting similar forensic investigations.

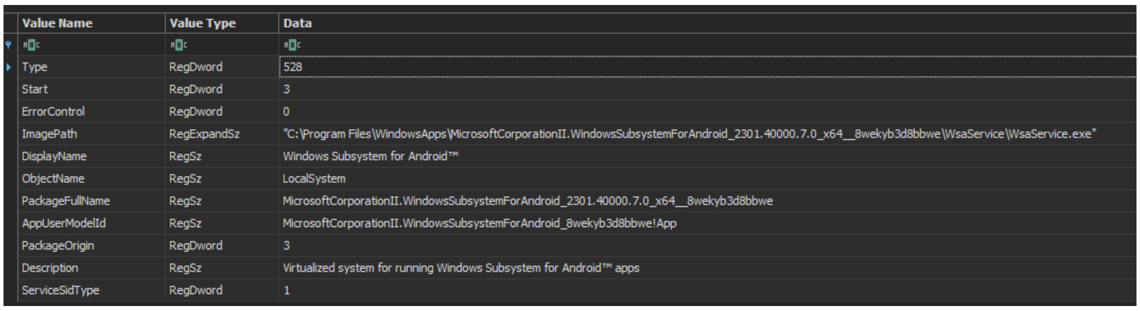
5.2.1 NTUSER.dat Hive

Table 3 presents the significant registry artifacts found in the NTUSER.dat registry hive, providing valuable evidence of WSA's existence.

Table 3. NTUSER.dat registry evidence.

Registry Key	**Value**
Software\Microsoft\Windows\CurrentVersion\App Paths\WsaClient.exe	Path
Software\RegisteredApplications	
Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store	Path (WSA-pacman.exe, WSA-pacman-v1.4.0-installer.exe)
\SOFTWARE\Microsoft\Windows\CurrentVersion\CloudStore	Binary format (WSA Pacman, WSA)
Software\Microsoft\Windows\CurrentVersion\UFH\SHC	Path (WSA PacMan.lnk, WSA-pacman.exe)
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs	Opened Extension Last Opened
Software\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\AppSwitched	Name Data

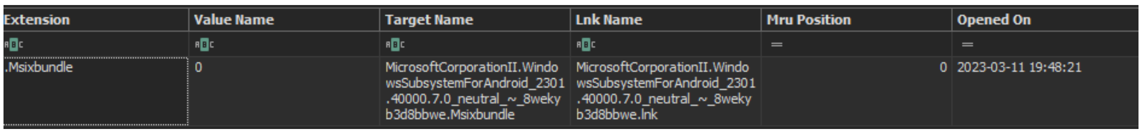
The AppPaths subkey delivers vital distribution information as seen in Figure 4, while the RegisteredApplications subkey contains data concerning the default programs linked to specific applications for the operating system. The CompatibilityAssistant\Store subkey has the potential to provide investigators with insights about third-party applications that have been executed on the system. Moreover, the SHC key encompasses information about GUI applications.



Value Name	Value Type	Data
Type	RegDword	528
Start	RegDword	3
ErrorControl	RegDword	0
ImagePath	RegExpandSz	"C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\WsaService\WsaService.exe"
DisplayName	RegSz	Windows Subsystem for Android™
ObjectName	RegSz	LocalSystem
PackageFullName	RegSz	MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe
AppUserModelId	RegSz	MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe!App
PackageOrigin	RegDword	3
Description	RegSz	Virtualized system for running Windows Subsystem for Android™ apps
ServiceSidType	RegDword	1

Figure 4. WsaService distribution information.

The RecentDocs key is a significant registry entry that monitors opened files and folders. For example, it documented the WSA MSIX Windows app package opening at 19:48:21. Notably, the RecentDocs key discloses evidence of employing the Windows feature "Turn Windows Features on or off". The Windows feature was used to activate the virtualization platforms required for the WSA installation. Consequently, the RecentDocs subkey can offer intriguing insights relevant to a WSA investigation. It sheds light on the WSA's presence in the system and the user's activities and interactions regarding the subsystem's installation.



Extension	Value Name	Target Name	Lnk Name	Mru Position	Opened On
.Msixbundle	0	MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_neutral_~_8wekyb3d8bbwe.Msixbundle	MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_neutral_~_8wekyb3d8bbwe.lnk	0	2023-03-11 19:48:21

Figure 5. RecentDocs subkey: Evidence of MSIX Windows app package opening.

The AppSwitched subkey offers valuable insights into application execution, including the number of times an application has come into focus. This information is particularly beneficial for understanding user interactions with various applications. By analysing the AppSwitched key, forensic investigators can better understand the user's behaviour, application usage patterns, and the frequency of engagement with specific applications.

5.2.2 UsrClass.dat Hive

Table 4 presents artifacts extracted from the UsrClass.dat file.

Table 4. UsrClass.dat registry evidence.

Registry Key	**Value**
Local Settings\Software\Microsoft\Windows\Shell\MuiCache	FriendlyName
Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe	ApplicationName PackageRootFolder PackageID PackageSid
Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\SystemAppData\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe	

The UsrClass.dat file Packages subkey, and the previously mentioned registry NTUSER.dat AppPaths subkey can offer valuable distribution information regarding the investigation. Another noteworthy aspect is the Muicache, which retains references to the file descriptions located within an executable's resource when the executable is launched. This information can be beneficial during a forensic examination, providing insights into the executed applications.

The SystemAppData registry subkey reveals information related to the startup of applications. By examining this key, forensic investigators can gain a deeper understanding of the applications initiated on the system, shedding light on potential user activities and behaviours.

5.2.3 SYSTEM Registry Hive

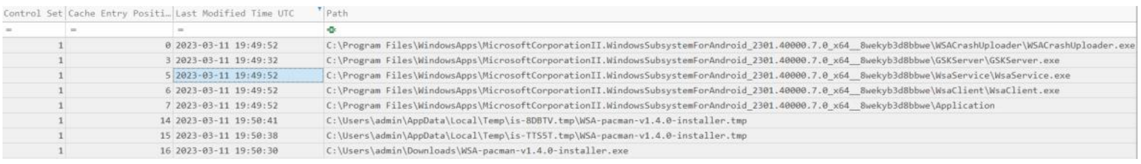
The following Table 5 provides an overview of the key artifacts that were observed in the SYSTEM registry hive.

Table 5. SYSTEM registry hive evidence

Registry Key	**Value**
ControlSet001\Services\WsaService	ImagePath DisplayName ObjectName PackageFullName AppUserModelId PackageOrigin Description ServiceSidType Start
ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\RestrictedServices\AppIso\FirewallRules	Rule
ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\FirewallRules	Action Active Dir Protocol LPort Name App
ControlSet001\Control\Session Manager\AppCompatCache	Full path Last modified time Cache entry position Control set
ControlSet001\Services\bam\State\UserSettings\S-1-5-21-1840747146-4148207901-798464498-1000	Path Data

The SYSTEM hive revealed several compelling artifacts associated with the presence of WSA in the system. The WSAService subkey was created under the Services subkey during installation, containing information such as ImagePath, PackageFullName, Start, and more, as detailed in Table 5. The Start RegDword was set to 3, meaning the WsaService will start if the user or process requests it. The BAM artifact held information on WSA and WSA Pacman execution time and data. Evidence of FirewallRules subkey can be found in Appendix 2.

Of particular interest was the Shimcache, located within the AppCompatCache subkey. The KAPE Module was employed to parse the AppCompatCache, and the data was outputted in CSV. Shimcache retains program execution data, and upon further examination in conjunction with Prefetch and Amcache, it became evident that Shimcache contains the earliest information on some of the WSA-related program and application executions.



Control Set	Cache Entry Positi…	Last Modified Time UTC	Path
1	0	2023-03-11 19:49:52	C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\WSACrashUploader\WSACrashUploader.exe
1	3	2023-03-11 19:49:32	C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\GSKServer\GSKServer.exe
1	5	2023-03-11 19:49:52	C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\WsaService\WsaService.exe
1	6	2023-03-11 19:49:52	C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\WsaClient\WsaClient.exe
1	7	2023-03-11 19:49:52	C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\Application
1	14	2023-03-11 19:50:41	C:\Users\admin\AppData\Local\Temp\is-8DBTV.tmp\WSA-pacman-v1.4.0-installer.tmp
1	15	2023-03-11 19:50:38	C:\Users\admin\AppData\Local\Temp\is-TTS5T.tmp\WSA-pacman-v1.4.0-installer.tmp
1	16	2023-03-11 19:50:30	C:\Users\admin\Downloads\WSA-pacman-v1.4.0-installer.exe

Figure 6. Shimcache: Application and program execution evidence

Furthermore, within the Shimcache, program execution artifacts associated with WSA indicated a last modified time of 19:49:52. Cross-referencing this information with the NTUSER.dat RecentDocs subkey recorded the opening of the WSA MSIX Windows app package at 19:48:21, a specific timeframe for the installation can be established. This correlation of data serves to enhance the accuracy and comprehensiveness of the forensic investigation.

5.2.4 SOFTWARE Hive

Table 6 comprises a collection of registry evidence extracted from the SOFTWARE hive. These keys provide insights into the installation, configuration, and operation of WSA on the host system.

Table 6. SOFTWARE registry hive evidence.

Registry Key	**Value**
Microsoft\Windows\CurrentVersion\Uninstall\WSA PacMan_is1	DisplayName DisplayIcon UninstallString, DisplayVersion Publisher InstallDate EstimatedSize
Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\PackageRepository\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe	
Microsoft\SecurityManager\CapAuthz\ApplicationsEx\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe	PackageSid
Microsoft\Windows\CurrentVersion\AppModel\StagingInfo\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe	DownloadSize
Classes\PackagedCom\Package\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe	**Subkeys:** Class Interface Proxy Server
Classes\PackagedCom\ClassIndex\{0D391720-9780-4575-88FF-BB89716B081F}\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe	

In the Windows Registry, the Microsoft\Windows\CurrentVersion\Uninstall key contains valuable information regarding the installation of various applications, including the WSA Pacman. By examining this key, investigators can obtain detailed insights into the WSA Pacman's installation, such as the installation date, the publisher of the application, and other relevant data. Furthermore, the SOFTWARE hive provides information about the WSA Pacman, such as its Package SID and Download size.

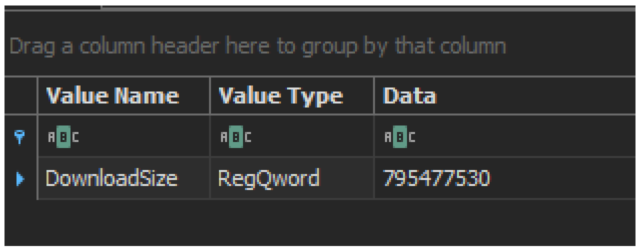
Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\PackageRepository\Packages indicates the presence of WSA package. The subkey contained the full path of the package and references to WsaClient.exe. WsaClient is an executable file responsible for managing and executing Android applications within the Windows environment.

Under the registry path Microsoft\SecurityManager\CapAuthz\ApplicationsEx, valuable information on the security manager's authorization settings for Windows Subsystem for Android (WSA) can be found. Within this subkey, a specific value named "PackageSID" represents a unique security identifier assigned to facilitate communication between WSA applications and the endpoint [1].

The installation and staging process of Windows Subsystem for Android is under the registry key Microsoft\Windows\CurrentVersion\AppModel\StagingInfo. Within this key, there is a subkey named DownloadSize, which provides information about the size of the downloaded files during the installation process.

The DownloadSize value is stored as REG_QWORD, a data type representing a 64-bit integer. In this case, the value represents the size of the downloaded files in bytes. By analyzing this value, it is possible to gain insights into the size of the installation package.

As seen in Figure 7, the DownloadSize is 795477539, this value can be converted to a more familiar unit like megabytes (MB). The conversion yields a size of 759MB. It is important to note that this calculation does not include decimal places, and the resulting value represents the approximate size of the download.



Value Name	Value Type	Data
DownloadSize	RegQword	795477530

Figure 7. DownloadSize under Microsoft\Windows\CurrentVersion\AppModel\StagingInfo.

The registry subkey Classes\PackagedCom\Package contains valuable data regarding the proxy, server, and interface information associated with WSA.

[1] https://learn.microsoft.com/en-us/gaming/game-bar/guide/communicating-apps (accessed May 10, 2023)

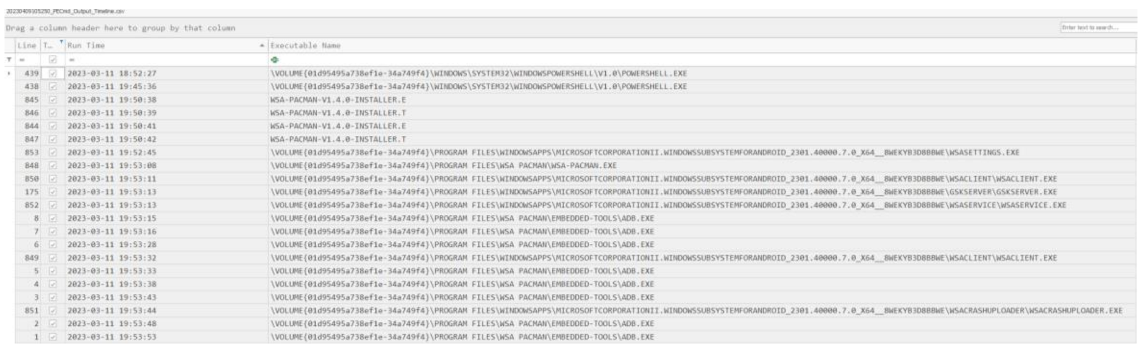
5.3 Prefetch and Amcache: Program and Application Execution:

Prefetch, a Windows feature, and the Amcache hive are valuable resources for obtaining information on application execution and tracking installed applications and programs that have been executed or are present on the system. The PECmd command-line tool is employed to parse Prefetch files, while the KAPE module was used to extract program execution data from the Amcache hive.

A thorough examination of the Prefetch files reveals a wealth of information about the executables associated with the WSA installation. Interestingly, the Amcache hive contains evidence of the WSA Pacman application only. When comparing the data from both sources, it becomes evident that the Amcache hive records execution events a few seconds earlier than the Prefetch files.

Moreover, Prefetch documented the execution of powershell.exe with its most recent run time at 19:45:36 (the first Powershell execution was related to Sysmon installation) which can be viewed as a significant artifact in determining the WSA installation timeline. As previously mentioned, the RecentDocs subkey recorded the WSA MSIX Windows app package opening at 19:48:21, and Shimcache registered the execution of WSAClient.exe at 19:49:52.

As demonstrated in Figures 8 and 9, the evidence of program execution is documented, showcasing the relevant data obtained from the analysis of Prefetch and Amcache hives. These figures visually represent the execution events, enabling a clearer understanding of the associated timeline and user activities related to the WSA installation and usage.

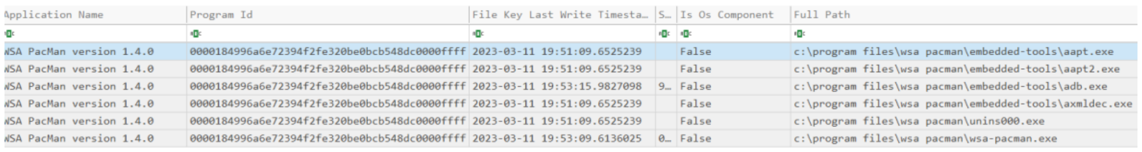


20230409105250_PECmd_Output_Timeline.csv

Drag a column header here to group by that column

Line	T...	Run Time	Executable Name
439	✓	2023-03-11 18:52:27	\VOLUME{01d95495a738ef1e-34a749f4}\WINDOWS\SYSTEM32\WINDOWSPOWERSHELL\V1.0\POWERSHELL.EXE
438	✓	2023-03-11 19:45:36	\VOLUME{01d95495a738ef1e-34a749f4}\WINDOWS\SYSTEM32\WINDOWSPOWERSHELL\V1.0\POWERSHELL.EXE
845	✓	2023-03-11 19:50:38	WSA-PACMAN-V1.4.0-INSTALLER.E
846	✓	2023-03-11 19:50:39	WSA-PACMAN-V1.4.0-INSTALLER.T
844	✓	2023-03-11 19:50:41	WSA-PACMAN-V1.4.0-INSTALLER.E
847	✓	2023-03-11 19:50:42	WSA-PACMAN-V1.4.0-INSTALLER.T
853	✓	2023-03-11 19:52:45	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WINDOWSAPPS\MICROSOFTCORPORATIONII.WINDOWSSUBSYSTEMFORANDROID_2301.40000.7.0_X64__8WEKYB3D8BBWE\WSASETTINGS.EXE
848	✓	2023-03-11 19:53:08	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\WSA-PACMAN.EXE
850	✓	2023-03-11 19:53:11	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WINDOWSAPPS\MICROSOFTCORPORATIONII.WINDOWSSUBSYSTEMFORANDROID_2301.40000.7.0_X64__8WEKYB3D8BBWE\WSACLIENT\WSACLIENT.EXE
175	✓	2023-03-11 19:53:13	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WINDOWSAPPS\MICROSOFTCORPORATIONII.WINDOWSSUBSYSTEMFORANDROID_2301.40000.7.0_X64__8WEKYB3D8BBWE\GSKSERVER\GSKSERVER.EXE
852	✓	2023-03-11 19:53:13	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WINDOWSAPPS\MICROSOFTCORPORATIONII.WINDOWSSUBSYSTEMFORANDROID_2301.40000.7.0_X64__8WEKYB3D8BBWE\WSASERVICE\WSASERVICE.EXE
8	✓	2023-03-11 19:53:15	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\EMBEDDED-TOOLS\ADB.EXE
7	✓	2023-03-11 19:53:16	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\EMBEDDED-TOOLS\ADB.EXE
6	✓	2023-03-11 19:53:28	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\EMBEDDED-TOOLS\ADB.EXE
849	✓	2023-03-11 19:53:32	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WINDOWSAPPS\MICROSOFTCORPORATIONII.WINDOWSSUBSYSTEMFORANDROID_2301.40000.7.0_X64__8WEKYB3D8BBWE\WSACLIENT\WSACLIENT.EXE
5	✓	2023-03-11 19:53:33	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\EMBEDDED-TOOLS\ADB.EXE
4	✓	2023-03-11 19:53:38	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\EMBEDDED-TOOLS\ADB.EXE
3	✓	2023-03-11 19:53:43	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\EMBEDDED-TOOLS\ADB.EXE
851	✓	2023-03-11 19:53:44	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WINDOWSAPPS\MICROSOFTCORPORATIONII.WINDOWSSUBSYSTEMFORANDROID_2301.40000.7.0_X64__8WEKYB3D8BBWE\WSACRASHUPLOADER\WSACRASHUPLOADER.EXE
2	✓	2023-03-11 19:53:48	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\EMBEDDED-TOOLS\ADB.EXE
1	✓	2023-03-11 19:53:53	\VOLUME{01d95495a738ef1e-34a749f4}\PROGRAM FILES\WSA PACMAN\EMBEDDED-TOOLS\ADB.EXE

Figure 8. Prefetch: Application execution evidence.

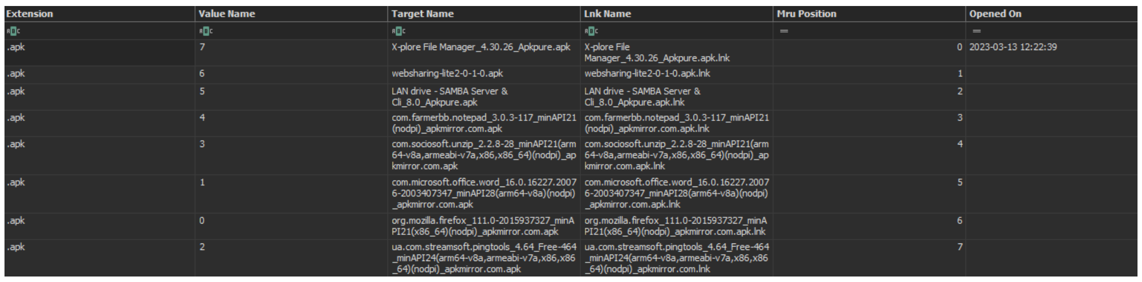


Application Name	Program Id	File Key Last Write Timesta…	S…	Is Os Component	Full Path
WSA PacMan version 1.4.0	0000184996a6e72394f2fe320be0bcb548dc0000ffff	2023-03-11 19:51:09.6525239		False	c:\program files\wsa pacman\embedded-tools\aapt.exe
WSA PacMan version 1.4.0	0000184996a6e72394f2fe320be0bcb548dc0000ffff	2023-03-11 19:51:09.6525239		False	c:\program files\wsa pacman\embedded-tools\aapt2.exe
WSA PacMan version 1.4.0	0000184996a6e72394f2fe320be0bcb548dc0000ffff	2023-03-11 19:53:15.9827098	9…	False	c:\program files\wsa pacman\embedded-tools\adb.exe
WSA PacMan version 1.4.0	0000184996a6e72394f2fe320be0bcb548dc0000ffff	2023-03-11 19:51:09.6525239		False	c:\program files\wsa pacman\embedded-tools\axmldec.exe
WSA PacMan version 1.4.0	0000184996a6e72394f2fe320be0bcb548dc0000ffff	2023-03-11 19:51:09.6525239		False	c:\program files\wsa pacman\unins000.exe
WSA PacMan version 1.4.0	0000184996a6e72394f2fe320be0bcb548dc0000ffff	2023-03-11 19:53:09.6136025	0…	False	c:\program files\wsa pacman\wsa-pacman.exe

Figure 9. Amcache: Program execution evidence.

5.4 Application Artifacts

By examining the Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs registry key under NTUSER.dat, it can be determined the most recently used application for each file extension, as this key maintains a Most Recently Used (MRU) list for each file type. Focusing on the .apk extension, it is evident that the X-plore file manager was the latest application in use, with an opening date and time of 2023-03-13 12:22:39 as seen in Figure 10. This information is particularly valuable, as it provides insight into the applications that users have utilized, including those that have been removed, thereby offering a comprehensive overview of user behaviour.



Extension	Value Name	Target Name	Lnk Name	Mru Position	Opened On
.apk	7	X-plore File Manager_4.30.26_Apkpure.apk	X-plore File Manager_4.30.26_Apkpure.apk.lnk	0	2023-03-13 12:22:39
.apk	6	websharing-lite2-0-1-0.apk	websharing-lite2-0-1-0.apk.lnk	1	
.apk	5	LAN drive - SAMBA Server & Cli_8.0_Apkpure.apk	LAN drive - SAMBA Server & Cli_8.0_Apkpure.apk.lnk	2	
.apk	4	com.farmerbb.notepad_3.0.3-117_minAPI21(nodpi)_apkmirror.com.apk	com.farmerbb.notepad_3.0.3-117_minAPI21(nodpi)_apkmirror.com.apk.lnk	3	
.apk	3	com.sociosoft.unzip_2.2.8-28_minAPI21(arm64-v8a,armeabi-v7a,x86,x86_64)(nodpi)_apkmirror.com.apk	com.sociosoft.unzip_2.2.8-28_minAPI21(arm64-v8a,armeabi-v7a,x86,x86_64)(nodpi)_apkmirror.com.apk.lnk	4	
.apk	1	com.microsoft.office.word_16.0.16227.20076-2003407347_minAPI28(arm64-v8a)(nodpi)_apkmirror.com.apk	com.microsoft.office.word_16.0.16227.20076-2003407347_minAPI28(arm64-v8a)(nodpi)_apkmirror.com.apk.lnk	5	
.apk	0	org.mozilla.firefox_111.0-2015937327_minAPI21(x86_64)(nodpi)_apkmirror.com.apk	org.mozilla.firefox_111.0-2015937327_minAPI21(x86_64)(nodpi)_apkmirror.com.apk.lnk	6	
.apk	2	ua.com.streamsoft.pingtools_4.64_Free-464_minAPI24(arm64-v8a,armeabi-v7a,x86,x86_64)(nodpi)_apkmirror.com.apk	ua.com.streamsoft.pingtools_4.64_Free-464_minAPI24(arm64-v8a,armeabi-v7a,x86,x86_64)(nodpi)_apkmirror.com.lnk	7	

Figure 10. RecentDocs subkey: Evidence of application usage.

In addition, cross-referencing the evidence found in RecentDocs with artifacts in NTUSER\Software\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\AppSwitched, the AppSwitched subkey provides knowledge on how many times each application has been executed. The added data further enhances our understanding of application usage patterns and user interactions on the system.

5.5 File Operations and Artifacts

5.5.1 Test 1: File Transfer

Transferring files between the host system and Windows Subsystem for Android can be facilitated using third-party applications. In the context of this experiment, the X-plore file manager application was chosen to manage file transfers.

File Transfer Protocol (FTP) sharing was configured within the X-plore application to enable file transfer between the host system and WSA. To access the FTP location, a shortcut was created in Windows Explorer. During the experiment, it became apparent that X-plore did not satisfy the needs of this experiment since FTP sharing is Read-only access for the free version.

Consequently, the SDK-Platform tools[1] were acquired to utilize the Android Debug Bridge (ADB)[2] command-line tool to transfer files between the host system and the Windows Subsystem for Android (WSA). The command "adb.exe connect 127.0.0.1:58526" was executed to establish a connection with the WSA environment.

Following the successful connection, files could be transferred between the host and WSA using the "push" and "pull" commands. The "push" command enables the transfer of files from the host system to the WSA, while the "pull" command facilitates the transfer of files from the WSA to the host system.

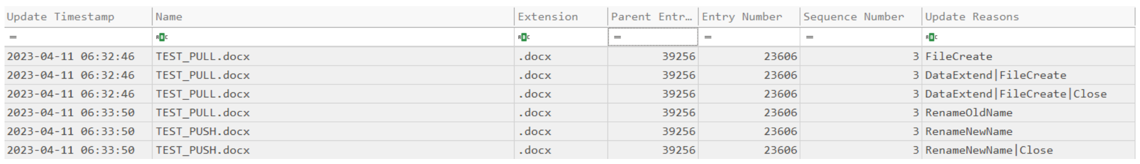
To conduct file operation testing, an Android Word application was used to create a file named TEST_PULL.docx, which was then saved to the WSA /storage/emulated/0/Documents folder. The ADB command-line tool was used to transfer the file from WSA to the host system. Once the TEST_PULL.docx file was successfully transferred, it was deleted from the WSA. Subsequently, the file was renamed to TEST_PUSH.docx on the host machine and "pushed" back to the /storage/emulated/0/Documents folder in the WSA. Therefore, a KAPE VHDX image of

[1] https://developer.android.com/tools/releases/platform-tools (accessed Apr. 10, 2023)

[2] https://developer.android.com/tools/adb (accessed Apr. 10, 2023)

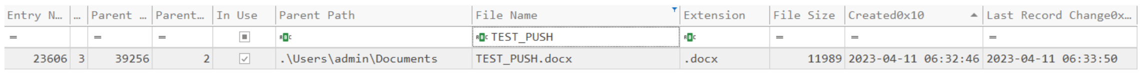
the $MFT and $J was processed using MFTECmd command-line tool, with the output generated in CSV format.

Upon examining the $MFT output, no record of the TEST_PULL.docx file was found. However, while analysing the $J output, it was observed that the TEST_PULL.docx file was created at 06:32:46 and renamed at 06:33:50, as seen in Figure 11. Upon analysing the Master File Table ($MFT), it was discovered that TEST_PUSH.docx was created at precisely 06:32:46, with its most recent record modification taking place at 06:33:50, as seen in Figure 12. By cross-referencing the timestamps with the output from the $J journal, it became evident that the original name of the file was TEST_PULL.docx. This conclusion is supported by Figure 11, which clearly shows that TEST_PULL.docx was renamed to TEST_PUSH.docx.



Update Timestamp	Name	Extension	Parent Entr…	Entry Number	Sequence Number	Update Reasons
=			=	=	=	
2023-04-11 06:32:46	TEST_PULL.docx	.docx	39256	23606	3	FileCreate
2023-04-11 06:32:46	TEST_PULL.docx	.docx	39256	23606	3	DataExtend|FileCreate
2023-04-11 06:32:46	TEST_PULL.docx	.docx	39256	23606	3	DataExtend|FileCreate|Close
2023-04-11 06:33:50	TEST_PULL.docx	.docx	39256	23606	3	RenameOldName
2023-04-11 06:33:50	TEST_PUSH.docx	.docx	39256	23606	3	RenameNewName
2023-04-11 06:33:50	TEST_PUSH.docx	.docx	39256	23606	3	RenameNewName|Close

Figure 11. $J.



Entry N…	…	Parent …	Parent…	In Use	Parent Path	File Name	Extension	File Size	Created0x10	Last Record Change0x…
=		=	=			TEST_PUSH		=	=	=
23606	3	39256	2	✓	.\Users\admin\Documents	TEST_PUSH.docx	.docx	11989	2023-04-11 06:32:46	2023-04-11 06:33:50

Figure 12. $MFT.

Through the analysis of the Master File Table ($MFT) and journal files ($J), no evidence was found suggesting that a file was created on the WSA subsystem rather than the host system. Additionally, the author used LECmd.exe to parse the TEST_PUSH.docx.lnk file to identify any data that could establish that TEST_PUSH.docx was created within WSA and not on the host. However, further investigation is necessary, and the current tools and techniques were deemed insufficient for finding the necessary artifacts that could link the TEST_PUSH.docx file back to the WSA subsystem. As a result, additional research is needed to address this issue in the future.

5.5.2 Test 2: EICAR File Download

The Eicar test file was employed in an experiment to evaluate the detection capabilities of Windows Defender when dealing with potentially malicious files accessed through both the host Firefox browser and the WSA Firefox application. The primary objective of this test was to ascertain whether Windows Defender could effectively identify and respond to the "malicious" test file in different scenarios.

Windows Defender successfully detected the Eicar test file when downloaded using the host browser, as evidenced by the corresponding entry in the Microsoft-Windows-Windows Defender%4Operational.evtx log. In contrast, the file downloaded through the WSA Firefox application went undetected. Notably, the eicar.com.txt file could not be downloaded using the Android Firefox browser, necessitating using the eicar_com.zip file for this experiment.

Interestingly, when the eicar_com.zip file was downloaded using the Android Firefox application, users were prompted to choose whether they wanted to open the file, with "Windows default preference" appearing as the first option. It was downloaded to the host system after selecting "Windows default preference" to open the file. However, Windows Defender did not automatically detect and remove the eicar_com.zip file unless the user interacted with it.

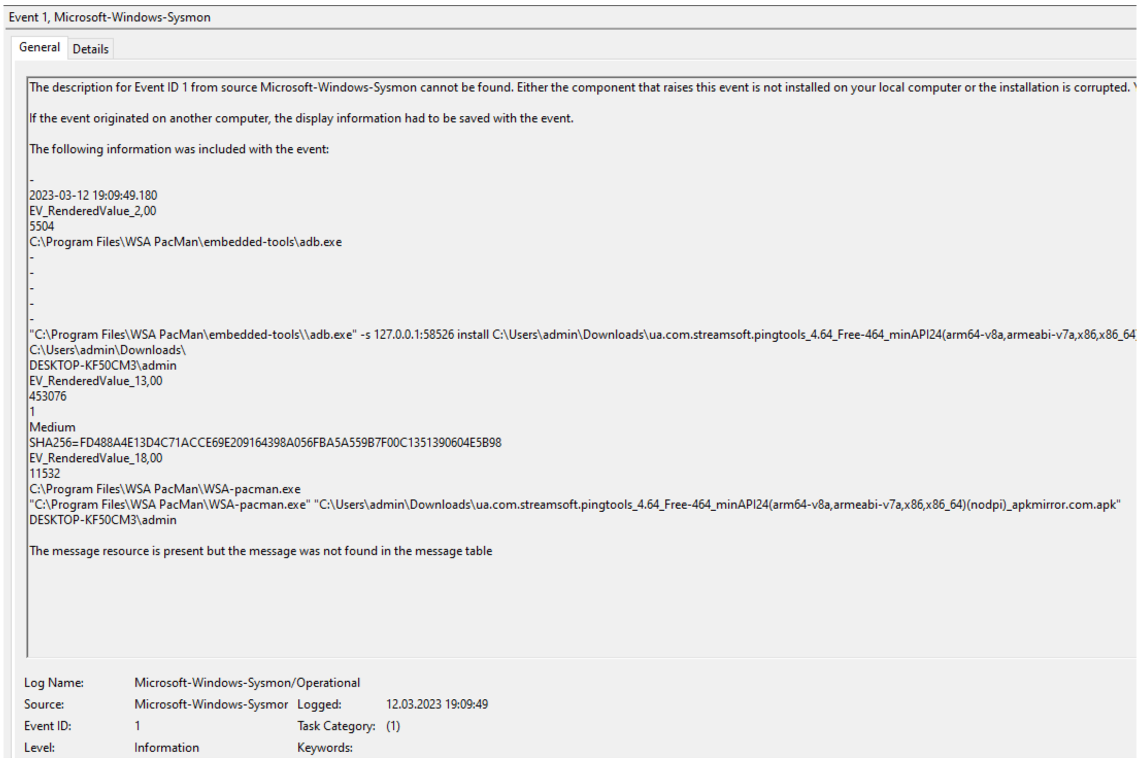
The author reported the issue through official Microsoft Support channels. In response, Microsoft Support explained that the performance of Windows Defender is intentionally configured to avoid overloading the operating system during regular operation. This design allows for quick system response times for the user, with scanning delays only occurring when necessary. Thus, it appears the lack of detection is a deliberate design choice by Microsoft to prioritize system performance over exhaustive malware detection.

The present controlled experiment has uncovered a notable limitation in the ability of Windows Defender to detect files accessed via the WSA environment. The WSA stores files in a VHDX container, so Windows Defender cannot detect them unless they appear on the host system. As a result, this poses a potential threat to the host system, particularly in scenarios where a malicious file is downloaded onto the WSA subsystem. Although the Windows Subsystem for Android does not have root access to the host system's resources; however, depending on the level of sophistication of the malware, there is a possibility of exploiting potential vulnerabilities to infect the host system.

5.6 Event Logs

The examination of event logs focused on identifying evidence of application downloads and any other data that could be of value to this investigation. Windows built-in Event Viewer and Event Log Explorer were used to analyse the event logs.

Sysmon captured several intriguing artifacts that shed light on the downloaded applications. With Event ID 1, Sysmon documented the installation of Word, PingTools, Firefox, and X-plore applications. This included the complete path to the location of the .apk file, the command used for installation ("adb.exe -s 127.0.0.1:58426 install"), the time of the event, and the SHA256. Further information can be found in Figure 13.



```
Event 1, Microsoft-Windows-Sysmon
General  Details

The description for Event ID 1 from source Microsoft-Windows-Sysmon cannot be found. Either the component that raises this event is not installed on your local computer or the installation is corrupted. Y

If the event originated on another computer, the display information had to be saved with the event.

The following information was included with the event:

-
2023-03-12 19:09:49.180
EV_RenderedValue_2,00
5504
C:\Program Files\WSA PacMan\embedded-tools\adb.exe
-
-
-
-
-
"C:\Program Files\WSA PacMan\embedded-tools\\adb.exe" -s 127.0.0.1:58526 install C:\Users\admin\Downloads\ua.com.streamsoft.pingtools_4.64_Free-464_minAPI24(arm64-v8a,armeabi-v7a,x86,x86_64
C:\Users\admin\Downloads\
DESKTOP-KF50CM3\admin
EV_RenderedValue_13,00
453076
1
Medium
SHA256=FD488A4E13D4C71ACCE69E209164398A056FBA5A559B7F00C1351390604E5B98
EV_RenderedValue_18,00
11532
C:\Program Files\WSA PacMan\WSA-pacman.exe
"C:\Program Files\WSA PacMan\WSA-pacman.exe" "C:\Users\admin\Downloads\ua.com.streamsoft.pingtools_4.64_Free-464_minAPI24(arm64-v8a,armeabi-v7a,x86,x86_64)(nodpi)_apkmirror.com.apk"
DESKTOP-KF50CM3\admin

The message resource is present but the message was not found in the message table

Log Name:     Microsoft-Windows-Sysmon/Operational
Source:       Microsoft-Windows-Sysmor  Logged:         12.03.2023 19:09:49
Event ID:     1                         Task Category:  (1)
Level:        Information               Keywords:
```

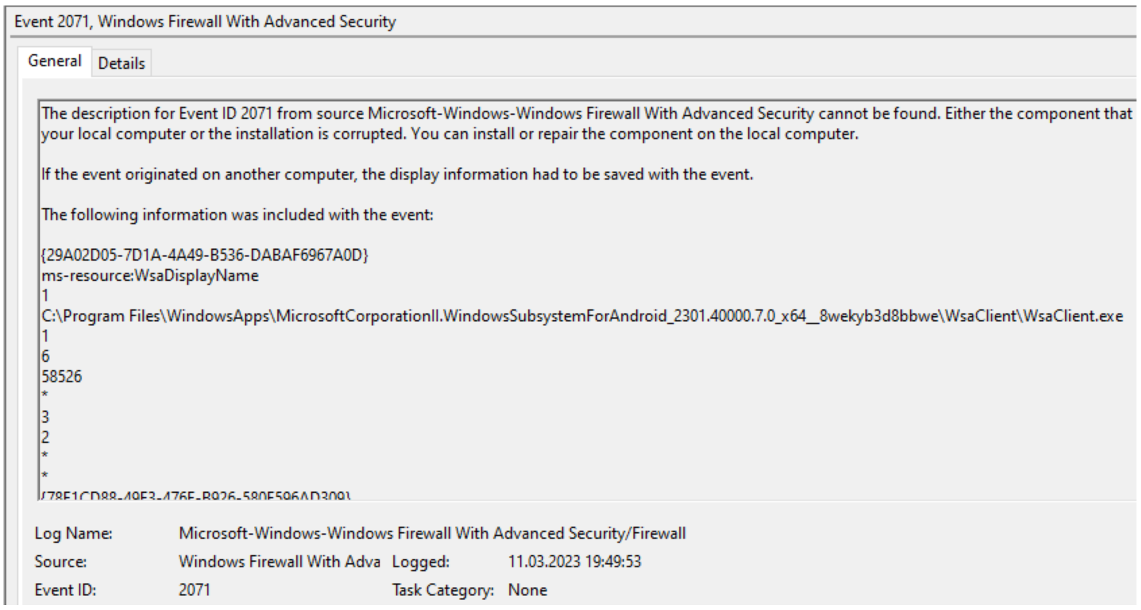
Figure 13. Sysmon: PingTools installation.

The Microsoft-Windows-Sysmon/Operational log records a dedicated event each time an app is launched, offering valuable insight to form a pattern in application usage with other artifacts such as the AppSwitched subkey under NTUSER.dat. In addition, Sysmon provides a comprehensive overview of the WSA Pacman installation process, corroborating the findings from the Prefetch analysis.

In the Sysmon event log depicted in Figure 13, the installation took place at 19:09:49. Examining the Microsoft-Windows-Shell-Core/Operational event log with Event ID 28115[1] shows the installation of PingTools slightly later, at 19:10:18. The same information is available for all other WSA applications installed for the purpose of the experiment.

The Microsoft-Windows-VHDMP/Operational event log contains records related to the WSA virtual disk. The first event occurred at 19:53:15, providing the full path of the virtual disk location, user SID, and details about the logged event.

The Microsoft-Windows-Windows Firewall With Advanced Security Firewall event log documented the creation of a rule for WsaClient.exe at 19:49:53, as seen in Figure 14. This log entry corroborates the installation timeline for the Windows Subsystem for Android (WSA) within the system. Furthermore, the Shimcache registry evidence supports this timeline, as it recorded the earliest instance of WsaClient.exe execution at 19:49:52. This consistency in the recorded events provides a reliable understanding of the WSA installation process. It helps establish a precise chronology of events during a forensic investigation.



```
The description for Event ID 2071 from source Microsoft-Windows-Windows Firewall With Advanced Security cannot be found. Either the component that
your local computer or the installation is corrupted. You can install or repair the component on the local computer.

If the event originated on another computer, the display information had to be saved with the event.

The following information was included with the event:

{29A02D05-7D1A-4A49-B536-DABAF6967A0D}
ms-resource:WsaDisplayName
1
C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\WsaClient\WsaClient.exe
1
6
58526
*
3
2
*
*
{78F1CD88-49F3-476E-B926-580F596AD309}

Log Name:     Microsoft-Windows-Windows Firewall With Advanced Security/Firewall
Source:       Windows Firewall With Adva   Logged:          11.03.2023 19:49:53
Event ID:     2071                         Task Category:   None
```

Figure 14. Rule creation for WsaClient.exe.

[1] https://nasbench.medium.com/finding-forensic-goodness-in-obscure-windows-event-logs-60e978ea45a3 (accessed Apr. 11, 2023)

Table 7 is provided below, presenting all the event logs associated with the Windows Subsystem for Android (WSA), WSA Pacman, and the installed applications. This table offers a thorough overview of the relevant events, enabling forensic investigators to understand the interactions and activities related to the WSA environment during the investigation.

Table 7. Overview of the event logs.

Event log	**Data**
Microsoft-Windows-Sysmon%4Operational .evtx	Path Command SHA256
Microsoft-Windows-Shell-Core%4Operational.evtx	Application installation information
Microsoft-Windows-VHDMP/Operational.evtx	Path SID
Microsoft-Windows-Hyper-V-VmSwitch-Operational.evtx	Virtual network configuration information
Microsoft-Windows-AppXDeployment-Server%4Operational.evtx	Deployment information SID Path
Application.evtx	Full path of the virtual disk
Microsoft-Windows-Hyper-V-Compute-Operational.evtx	
Microsoft-Windows-AppModel-Runtime%4Admin.evtx	Application process creation info Package name
Microsoft-Windows-Application-Experience%4Program-Compatibility-Assistant.evtx	SID Path
Microsoft-Windows-Store%4Operational.evtx	Path Build SID Package name
Microsoft-Windows-Windows Firewall With Advanced Security%4Firewall.evtx	Path

The event logs in the table include event IDs, event descriptions, paths, timestamps, and other information related to the subject under investigation, offering valuable insights into the installation, configuration, and usage of WSA, WSA Pacman, and the corresponding applications. By carefully analysing this detailed data, investigators can identify patterns, verify timelines, and uncover additional evidence pertinent to the case.

5.7 Network Artifacts

Results of network analysis depends on the type of traffic being observed and whether it is captured from the host physical adapter or virtual adapter, local area network, or public network. In this thesis, network capture was done using physical or virtual adapter.

The following subsection outlines the results of various tests, such as ping, firewall port 80 blocking, HTTP traffic, and VPN tests. Figure 16 shows a conceptual representation of the WSA network architecture created by the author based on her knowledge after conducting several experiments. The network architecture concept was developed to address questions raised by the outcomes of Test 4a and to provide an overview of the WSA network architecture.

To construct the network architecture concept for WSA, the author utilized the article "What is the Hyper-V Virtual Switch and How Does it Work?" [36], Figure 15 [37], and the results of the tests conducted in Stage 5. The author employed this approach to gain a more comprehensive understanding of how the WSA network architecture operates.

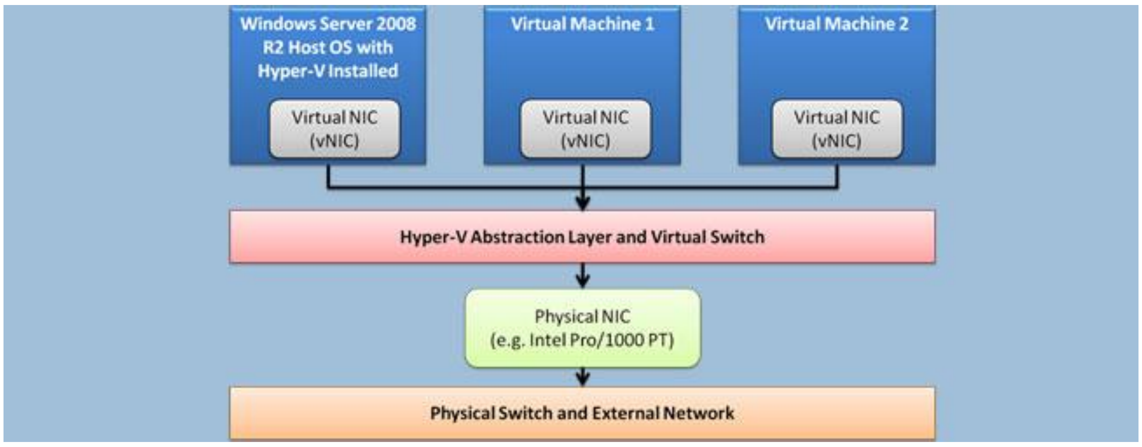

Windows Server 2008 R2 Host OS with Hyper-V Installed
Virtual NIC (vNIC)
Virtual Machine 1
Virtual NIC (vNIC)
Virtual Machine 2
Virtual NIC (vNIC)
Hyper-V Abstraction Layer and Virtual Switch
Physical NIC (e.g. Intel Pro/1000 PT)
Physical Switch and External Network

Figure 15. Hyper-V network illustration [37].

When Hyper-V is enabled on the host, it creates a network abstraction layer, which manages the host's network connectivity through the Hyper-V Extensible Switch. The host's physical network adapter (pNIC) connects to the Hyper-V Extensible Switch. A virtual network adapter (vNIC) is created for the host operating system, also known as the management operating system. The network traffic passes between the host and its physical network adapter through the virtual switch, allowing the host to share its physical network adapter with virtual machines. The WSL Core adapter (Hyper-V virtual switch) provides network connectivity to the WSA instance. The entirety of the WSA network architecture concept is visually represented in Figure 16. The IP address of the WSA Client will change upon each restart, resulting in varying IP addresses observed throughout the research process.

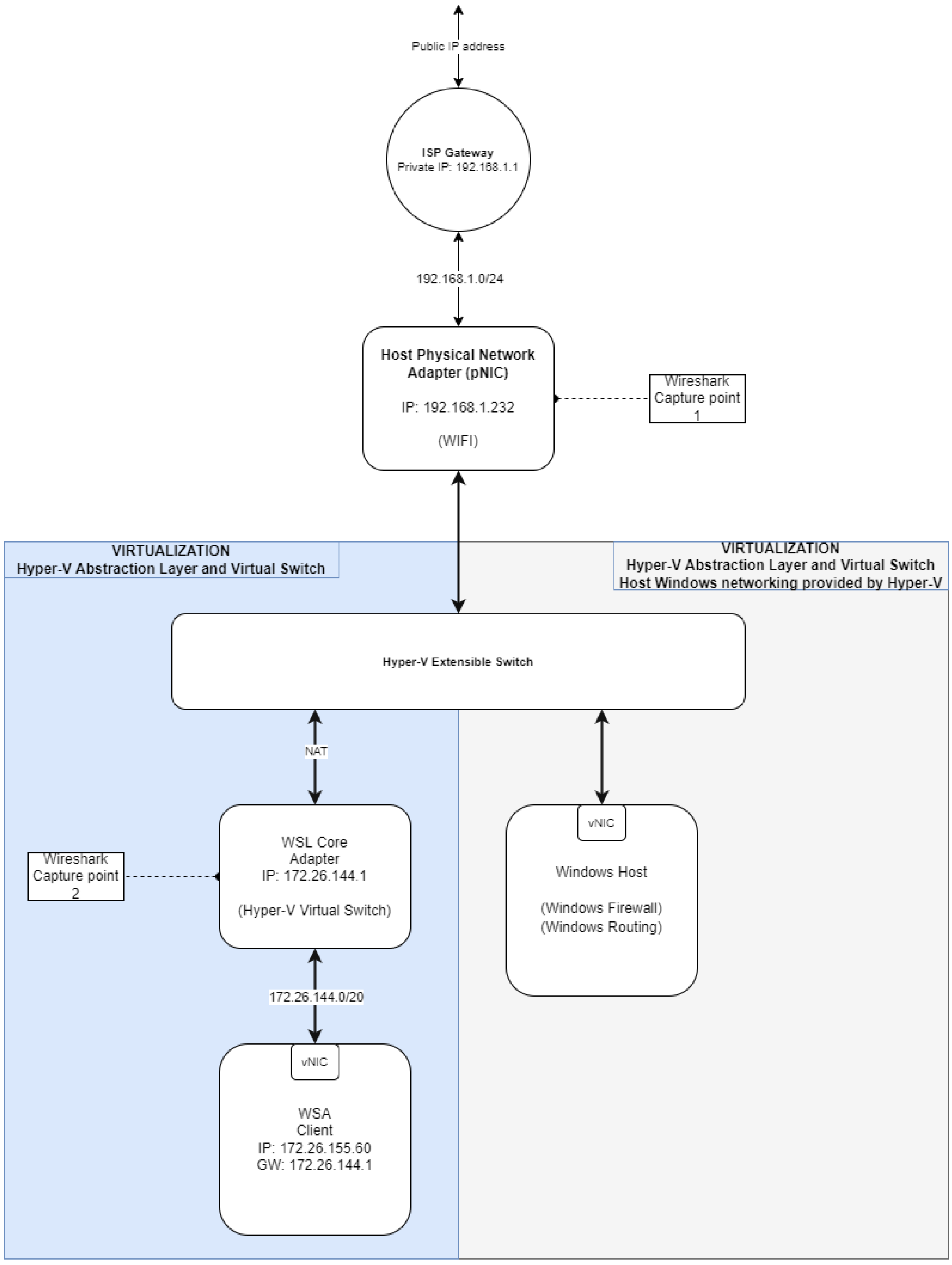
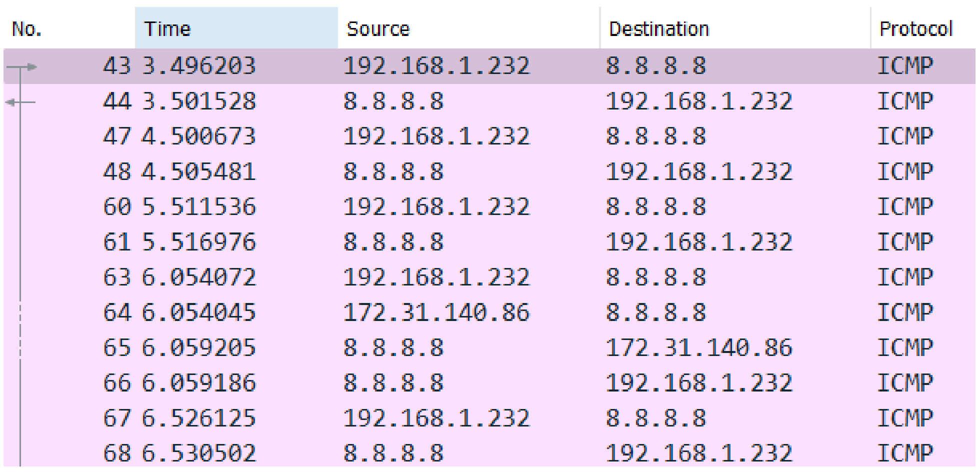


Figure 16. WSA network architecture concept.

5.7.1 Test 3: Ping Test Analysis

In order to assess network connectivity between the host system and the WSA, the network diagnostic tool, ping, was utilized. The initial ping test involved sending an Internet Control Message Protocol (ICMP) echo request from the host to WSA, resulting in a successful echo reply. Conversely, the second ICMP echo request was unsuccessful, originating from WSA to host using the Ping Tools application. The Windows host firewall blocks all incoming traffic by default, including ICMP, Server Message Block (SMB), Network Basic Input/Output System (NetBIOS), and Remote Procedure Call (RPC). The inbound File and Printer Sharing (Echo Request - ICMPv4-In) rule must be enabled to permit incoming ICMP traffic. The subsequent ping test from WSA to the host proved successful upon activating this rule.

Executing a ping to 8.8.8.8 from WSA using Ping Tools was successful similar to the hosts. This ping was monitored via Wireshark, with traffic capture performed on both the Wi-Fi (host) and vEthernet (WSLCore) interfaces. Analysing the captured network traffic from both interfaces made it possible to distinguish ping traffic based on the IP addresses. Specifically, the host was identified as having an IP address of 192.168.1.232, while the WSA possessed an IP address of 172.31.140.86.

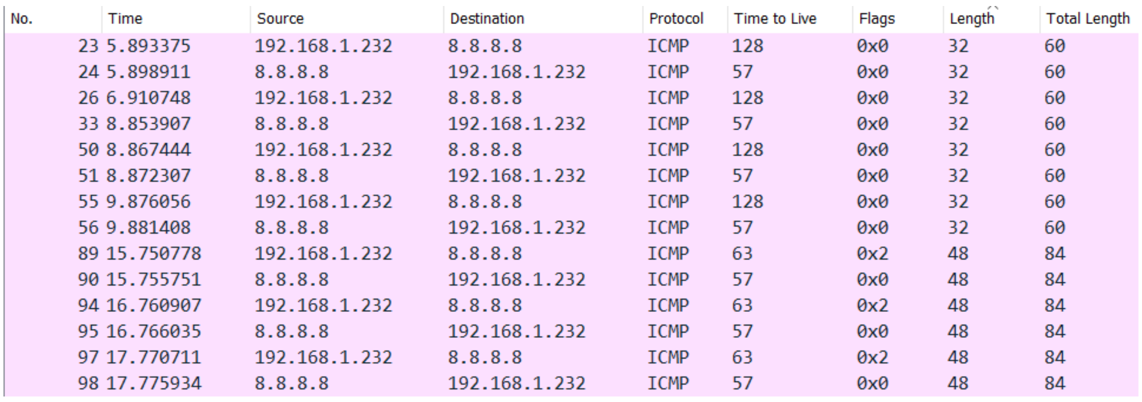


No.	Time	Source	Destination	Protocol
43	3.496203	192.168.1.232	8.8.8.8	ICMP
44	3.501528	8.8.8.8	192.168.1.232	ICMP
47	4.500673	192.168.1.232	8.8.8.8	ICMP
48	4.505481	8.8.8.8	192.168.1.232	ICMP
60	5.511536	192.168.1.232	8.8.8.8	ICMP
61	5.516976	8.8.8.8	192.168.1.232	ICMP
63	6.054072	192.168.1.232	8.8.8.8	ICMP
64	6.054045	172.31.140.86	8.8.8.8	ICMP
65	6.059205	8.8.8.8	172.31.140.86	ICMP
66	6.059186	8.8.8.8	192.168.1.232	ICMP
67	6.526125	192.168.1.232	8.8.8.8	ICMP
68	6.530502	8.8.8.8	192.168.1.232	ICMP

Figure 17. Wireshark capture of the host physical and WSA adapter.

The host IP serves as the default route for the WSA to establish internet connectivity, capturing network traffic from the host physical adapter results in translating (Network Address Translation) the WSA IP address to the host IP. Therefore, distinguishing WSA and host traffic by IP addresses is not possible. When analysing network traffic captured from the host physical interface, one approach to differentiate between pings originating from the host (pinging 8.8.8.8) and those from the WSA (also pinging 8.8.8.8) is to examine the ICMP packet payload, such as Time to Live (TTL) values, payload data sizes, and packet lengths.

As WSA is based on the Linux kernel, its default Time to Live (TTL) value for ICMP echoes is 64, identical to Linux[1]. However, since packets are routed through the host, the TTL value decreases by one. Thus, when capturing traffic from the host interface, Wireshark displays a TTL value of 63 for pings originating from WSA, while a TTL value of 128 denotes pings executed by the host. Additionally, as stated, pings can be distinguished by analysing the payload sizes. For WSA, the data size is 48 bytes, while the host's data size is 32 bytes. The Flags are set for WSA 0x2 and the host 0x0, as shown in Figure 18.



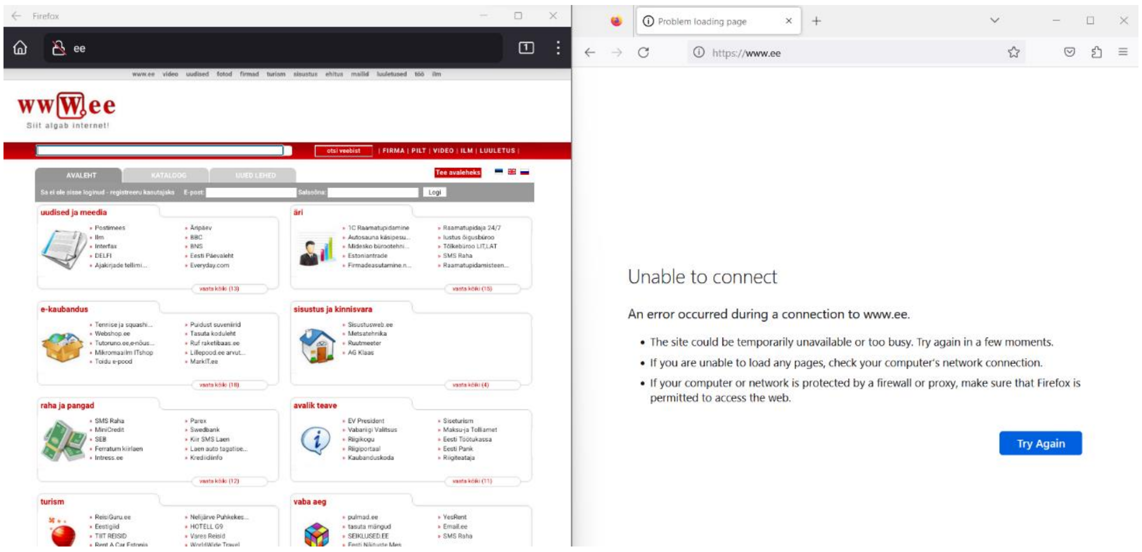
No.	Time	Source	Destination	Protocol	Time to Live	Flags	Length	Total Length
23	5.893375	192.168.1.232	8.8.8.8	ICMP	128	0x0	32	60
24	5.898911	8.8.8.8	192.168.1.232	ICMP	57	0x0	32	60
26	6.910748	192.168.1.232	8.8.8.8	ICMP	128	0x0	32	60
33	8.853907	8.8.8.8	192.168.1.232	ICMP	57	0x0	32	60
50	8.867444	192.168.1.232	8.8.8.8	ICMP	128	0x0	32	60
51	8.872307	8.8.8.8	192.168.1.232	ICMP	57	0x0	32	60
55	9.876056	192.168.1.232	8.8.8.8	ICMP	128	0x0	32	60
56	9.881408	8.8.8.8	192.168.1.232	ICMP	57	0x0	32	60
89	15.750778	192.168.1.232	8.8.8.8	ICMP	63	0x2	48	84
90	15.755751	8.8.8.8	192.168.1.232	ICMP	57	0x0	48	84
94	16.760907	192.168.1.232	8.8.8.8	ICMP	63	0x2	48	84
95	16.766035	8.8.8.8	192.168.1.232	ICMP	57	0x0	48	84
97	17.770711	192.168.1.232	8.8.8.8	ICMP	63	0x2	48	84
98	17.775934	8.8.8.8	192.168.1.232	ICMP	57	0x0	48	84

Figure 18. Wireshark capture of the host adapter.

5.7.2 Test 4a and Test 4b: Firewall Rule and HTTP Traffic Analysis

As described in Stage 5, the following experiment entailed blocking port 80 to inhibit users from accessing HTTP websites. This approach effectively restricted the host Firefox browser from accessing a designated HTTP site (www.ee); however, the WSA Firefox application remained unaffected by the rule and could continue browsing HTTP sites without hindrance. A similar observation was made with the WSL2, where the firewall rule did not impose any constraints on its functionality. Figure 19 represents the effectiveness of the firewall rule on WSA, with WSA being on the left side and the host on the right.

[1] https://ostechnix.com/identify-operating-system-ttl-ping/ (accessed Apr. 08, 2023)



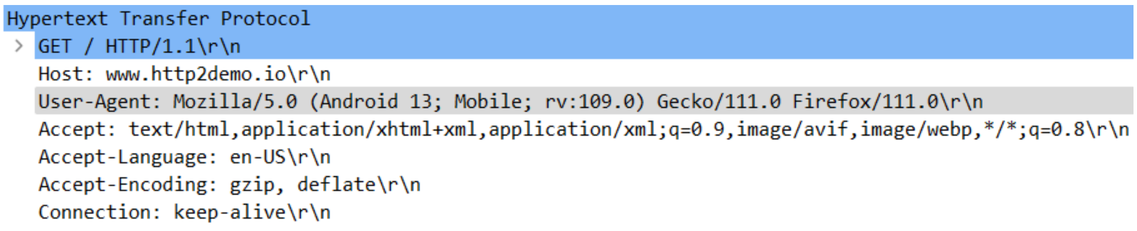
Unable to connect

An error occurred during a connection to www.ee.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

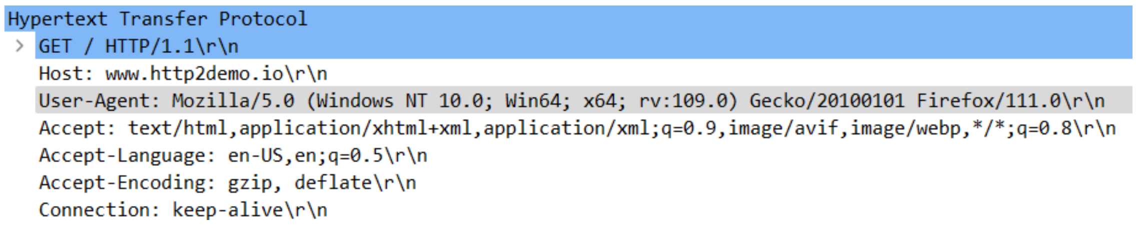
Figure 19. WSA Firefox and host Firefox after enabling the firewall rule.

To gather more data, Wireshark was used, and an HTTP site was revisited—after removing the firewall rule that blocked port 80—to capture browser fingerprints that help differentiate network traffic at the highest network layer. The HTTP protocol was chosen because it is plaintext, while Hypertext Transfer Protocol Secure (HTTPS) uses Transport Layer Security (TLS). This method allowed for the collection of different artifacts related to WSA. At the time of conducting this experiment, the user-agent for the WSA latest Firefox application is Mozilla/5.0 (Android 13; Mobile; rv:109.0) Gecko/111.0 Firefox/111.0\r\n. Figures 20 and 21 depict Wireshark captures of the Hypertext Transfer Protocol (HTTP) packets that contain the User-Agent fields of both the host and the WSA.



```
Hypertext Transfer Protocol
 > GET / HTTP/1.1\r\n
   Host: www.http2demo.io\r\n
   User-Agent: Mozilla/5.0 (Android 13; Mobile; rv:109.0) Gecko/111.0 Firefox/111.0\r\n
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
   Accept-Language: en-US\r\n
   Accept-Encoding: gzip, deflate\r\n
   Connection: keep-alive\r\n
```

Figure 20. User-Agent of the WSA.

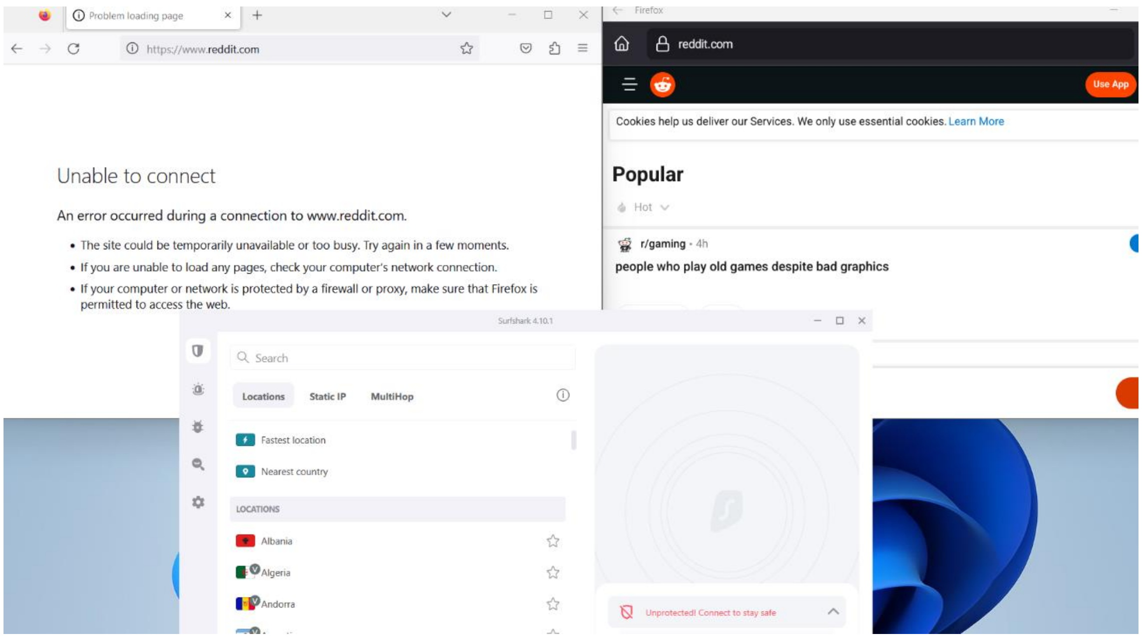


```
Hypertext Transfer Protocol
> GET / HTTP/1.1\r\n
  Host: www.http2demo.io\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Connection: keep-alive\r\n
```

Figure 21. User-Agent of the host.

5.7.3 Test 5: VPN Test Analysis

The concluding experiment aimed to reproduce the concerns highlighted in the article "Linux under WSL2 can be leaking" [29]. Surfshark VPN was installed and configured as detailed in Section 4.2 - Experiment Setup. Upon disconnecting the VPN service, the host system could not access any websites, while the WSA Firefox application retained the ability to navigate to any specified website. This observation suggests that the issue persists and applies to the WSA environment. Figure 22 displays how Surfshark VPN is disconnected, and host Firefox is unable to connect to the Reddit website; meanwhile, the WSA maintains the ability to access Reddit.



Unable to connect

An error occurred during a connection to www.reddit.com.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

Cookies help us deliver our Services. We only use essential cookies. Learn More

Popular

r/gaming • 4h

people who play old games despite bad graphics

Unprotected! Connect to stay safe

Figure 22. Visual representation of Surfshark VPN test outcome.

5.8 Comparison Between WSA and WSL2

The comparative analysis presented in this study is based upon the findings of a previous thesis titled "Forensic Analysis of Windows Subsystem for Linux on Windows 11 [28]." Both investigations adopt a similar controlled experimental approach. Specific experiments in the current research are chosen to facilitate direct comparisons between the Windows Subsystem for Linux 2 (WSL2) and Windows Subsystem for Android (WSA).

This analysis will focus on evaluating the behaviour of both subsystems in the context of various scenarios, including the application of Windows Firewall port 80 block rules, Windows Defender's ability to detect malicious files, file operations between the host and subsystems, and the identification of key evidence on program and application execution artifacts. By drawing parallels between the WSA and WSL2, this analysis aims to understand their respective forensic implications and potential attack surfaces in a Windows 11 environment.

It is essential to acknowledge that the experiments in this study were conducted on two distinct Windows 11 operating system versions. Consequently, the comparative analysis will not delve into the registry artifacts specifics, as their relevance may be contingent upon the specific OS version. Instead, the focus will be on identifying more definitive similarities and differences between the WSL2 and WSA, which can provide meaningful insights and conclusions regarding their respective forensic characteristics in the broader context of the subsystems in the Windows 11 environment. To facilitate a more comprehensive and accurate comparison between the WSL2 and WSA, performing an additional controlled experiment involving both subsystems within the same experimental setting would be essential.

5.8.1 Firewall Rule Test

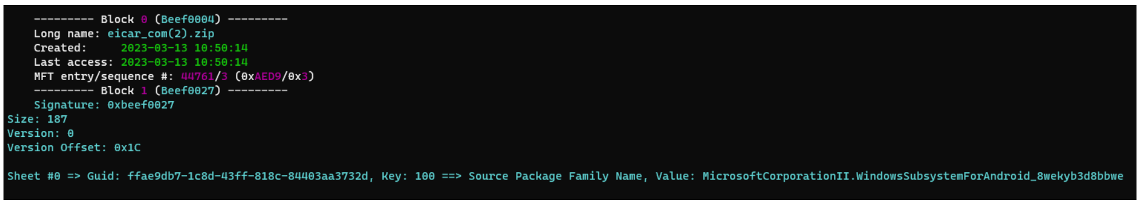
In both experiments, firewall rules were implemented to restrict traffic to specific ports. For the WSL2 experiment, ports 80, 8080, and 443 were blocked [28], while in the WSA, only HTTP traffic to port 80 was blocked, as this was deemed sufficient for the investigation. After the enforcement of the firewall rules, the host browser and the GUI application Firefox were utilized to assess the effectiveness of these restrictions. The results demonstrated a consistent pattern across both subsystems, with the host internet access being successfully restricted. In contrast, the GUI Firefox applications within the respective subsystems retained their connectivity, meaning that the applied firewall rules do not impact the network access of the subsystems.

5.8.2 Eicar File Test

In the second comparison, the focus is on the Eicar test file. In the WSL2 experiment, this test involved changing file permissions and moving files between the subsystem and the host. However, due to the absence of root access in the WSA experiment, only the file transfer capability between the host and subsystem was assessed, including finding indicators to file creation location.

In the WSL2 experiment, it was possible to identify files originating from WSL2 within the host system by the SWP file extension [28]. In contrast, the WSA experiment did not allow such distinction by examining the $MFT and $J data streams. To prove the file originated from the WSA, the author employed additional pivot points, including the TEST_PUSH.docx.lnk shortcut file analysis. However, this method yielded insignificant results. As such, additional research is necessary to identify relevant artifacts and establish a link between files originating from the WSA.

Nevertheless, when downloading the Eicar file using the WSA Firefox application and opting to open the file with "Windows default preferences," which also downloads the file to the host system, an LNK file is created. By employing the LECmd.exe command-line tool to decode the information within the downloaded Eicar test file, some indication that the file's existence is related to WSA can be determined as seen in Figure 23.



```
    --------- Block 0 (Beef0004) ---------
    Long name: eicar_com(2).zip
    Created:     2023-03-13 10:50:14
    Last access: 2023-03-13 10:50:14
    MFT entry/sequence #: 44761/3 (0xAED9/0x3)
    --------- Block 1 (Beef0027) ---------
    Signature: 0xbeef0027
Size: 187
Version: 0
Version Offset: 0x1C

Sheet #0 => Guid: ffae9db7-1c8d-43ff-818c-84403aa3732d, Key: 100 ==> Source Package Family Name, Value: MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe
```

Figure 23. Evidence of the Eicar test file relations to WSA.

5.8.3 Program and Application Execution Artifacts

The Amcache hive, AppCompatCache registry key, and Prefetch offered a wealth of information for both subsystems. Notably, the earliest recorded data on the installation of

WSA and WSL2 were consistent for both subsystems, as evidenced by the AppCompatCache registry key. This highlights the significance of these forensic artifacts in providing valuable insights into the installation timeline and usage patterns of WSA and WSL2 subsystems within a comprehensive forensic examination context.

6 Discussion

The Discussion chapter thoroughly discusses this thesis's findings, highlighting pivotal points for future research. This study aimed to address two primary research questions:

- What artifacts are left behind by WSA, and where can they be found?
- Can digital artifacts found in WSA be compared with artifacts found in WSL2?

A controlled experiment was conducted to address these questions, providing insights into the research questions posed. The investigation sought to comprehend the components comprising the Windows Subsystem for Android (WSA), trace user activities, and establish a detailed timeline concerning the installation of the subsystem. This chapter emphasizes some of the most intriguing artifacts discovered during the research, which may prove beneficial for future forensic investigations of a similar nature. The author acknowledges the value of these artifacts in contributing to a deeper understanding of the WSA and its forensic implications.

Throughout the experiment, we identified the virtual hard disk (VHDX) files that WSA employs to store its data and operating system resources. These files were found in the following locations:

- Users<username>\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalCache\userdata.vhdx
- Users<username>\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalCache\metadata.vhdx

Furthermore, the WsaService.exe was located at C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\WsaService\WsaService.exe. This essential core service plays a pivotal role in ensuring the proper functioning of the WSA subsystem.

Another noteworthy discovery within the WSA environment is the Android-specific application log, located at \Users<username>\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalState\diagnostics\logcat. This log presents a potential focal point for future research, given that the present thesis did not delve into a comprehensive analysis of this log.

Transitioning to the analysis of registry hives, the thesis examined NTUSER.dat, UsrClass.dat, SYSTEM, and SOFTWARE, revealing key findings such as artifacts that can assist in identifying evidence of WSA's existence, distribution information, and application execution. Additionally, the research uncovered artifacts that may facilitate tracking user activities related to WSA, thereby enriching the understanding of this subsystem from a digital forensics' perspective.

One of the most intriguing findings in this research was the registry key HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs, which recorded the opening of the WSA MSIX Windows app package at 19:48:21. This timestamp was identified as the earliest evidence of the WSA's presence on the system. However, the author hypothesizes that if WSA were installed through either the Amazon Appstore or Microsoft Store, this registry key might not exist. As such, this artifact warrants further investigation. Additional testing is needed to compare the installation method utilized in this thesis with the official methods of installing WSA through the Amazon Appstore or Microsoft Store. This comparison would help validate whether the identified registry key is exclusive to the installation method employed in this research or if it also appears when using the official installation methods.

In addition, the RecentDocs subkey plays a significant role in understanding user activities in the system, particularly when cross-referenced with NTUSER\Software\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\AppSwitched. RecentDocs maintains a record of the Most Recently Used (MRU) list of applications, thus enabling the identification of the last application used. Together, the AppSwitch subkey offers insights into the frequency of application usage, further enriching the understanding of user behaviour.

The most significant artifacts for tracking program and application usage include the Amcache hive, Prefetch, and AppCompatCache/Shimcache. When cross-referenced, these artifacts can offer a specific timeframe of the events that have occurred. These artifacts have also proven helpful in investigations of WSL2, with AppCompatCache/Shimcache providing the earliest timestamps of activity occurrence [28].

In Stage 4, Test 1 aimed to verify the feasibility of transferring files between the host and WSA using the ADB command-line tool. The analysis of the $MFT and $J files demonstrated that, by examining dates, timestamps, and the "Update Reasons" column in $J, it is possible to deduce that TEST_PUSH.docx was previously named TEST_PULL.docx and was not created on the host system. Regrettably, analysing the $MFT and $J files did not reveal the file's origin. In contrast, the WSL2 research allowed the author to determine the file's source by examining the SWP file extension [28]. This finding presents another potential area for future research, seeking artifacts that could help associate the file's origin with WSA.

To further clarify the author's intent behind conducting Test 1, the first objective was to confirm whether transfers from the host system to WSA were feasible. Second, the author considered hypothetical scenarios, such as data theft or malware infection of the host system, that would require digital forensic examiners to think critically when investigating devices with WSA installed. WSA could serve as a pathway for the exfiltration of files. In future work, the author intends to investigate further artifacts related to file movements and seek evidence to establish the file's origin.

In Test 2, the author sought to determine if Windows Defender behaved similarly to its performance in WSL2, as reported in the relevant thesis paper [28], where it failed to detect the Eicar test file downloaded via a GUI application. The results for WSA mirrored those of WSL2, with the file downloaded from the WSA Firefox application remaining undetected. Moreover, no evidence of the file installation was found in the Windows Defender%4Operational.evtx log, unlike the Eicar file downloaded from the host browser.

The network analysis offered a comprehensive understanding of WSA from a network standpoint. Tests conducted in Stage 5 enabled the development of the WSA network architecture concept depicted in Figure 15, which can aid forensic examiners in their investigations. Furthermore, based on the tests performed in Stage 5, WSA traffic can be differentiated from host traffic by factors such as TTL, payload data size, flags, and packet lengths. At the time of writing this thesis, the user-agent for the latest WSA Firefox application was Mozilla/5.0 (Android 13; Mobile; rv:109.0) Gecko/111.0 Firefox/111.0\r\n.

Another significant finding in this research is the results of Test 4a. In Test 4a, the firewall rules did not apply to the WSA GUI application, allowing it to continue browsing the web despite the host being restricted. This test was also performed on WSL2 [28], which inspired the investigation into whether WSA would be similarly affected. The test results prompted the author to conduct further research to investigate why the rules did not apply to the WSA. As a result, the author gained a deeper understanding of the WSA network architecture, as demonstrated in Section 5.7 - Network Artifacts, Figure 16. Hyper-V creates a network abstraction layer that manages the host network connectivity through the Hyper-V Extensible Switch. At the same time, the WSA client connection is provided by the WSL Core adapter (Hyper-V virtual switch) that is connected to the Hyper-V Extensible Switch. Consequently, any traffic originating from the WSA will bypass the host, rendering the rules inapplicable to the WSA.

Test 5 aimed to evaluate the impact of VPN on WSA and revealed that the issue, previously identified in a related article [29] also impacted WSA. As with the firewall, the traffic generated by WSA is not routed through the host, which explains the outcomes of the VPN test. According to the article "Linux under WSL2 can be leaking" [29], the issue is being addressed, but it remains unclear how the developers plan to achieve a solution since the traffic of both WSA and WSL2 does not pass through the host. Given this situation, the author remains sceptical regarding the effectiveness of finding a solution to remediate the issue.

7 Conclusion

This research was carried out in Estonia, a region outside the supported areas of the Amazon Appstore, which is necessary for the official installation of Windows Subsystem for Android (WSA). Therefore, the author employed alternative approaches to conducting the controlled experiment for this thesis.

This thesis hopes to serve as a foundation for the forensic investigation of WSA by offering an overview of key artifacts and nuances related to WSA. The author successfully identified crucial forensic artifacts that can help ascertain the presence of WSA on a system, construct a timeline of when the subsystem was installed, and track user activities related to application usage and installation. Additionally, the research provided an overview of artifacts within the NTUSER.dat, UsrClass.dat, SYSTEM, and SOFTWARE registry hives that may prove valuable. A review of event logs was also conducted, demonstrating their value in investigating WSA.

Several experiments were conducted to investigate the WSA from network forensic aspect and explore potential attack vectors discussed in the literature review. The research offers a comprehensive review of these observations and the distinctive characteristics of WSA. This includes a network architecture concept developed by the author, which clarifies why the application of firewall rules does not impact the WSA or when VPN is configured to operate in strict mode to mitigate against any Internet connectivity in case of VPN disconnections or interruptions, WSA can still visit websites even when the host is not.

The research compared WSA and WSL2, based on the "Forensic Analysis of Windows Subsystem for Linux on Windows 11" thesis [28]. The comparison examined file artifacts, program and application execution artifacts, and the firewall rule test was conducted in both experiments. This comparison aimed to determine similarities between these two subsystems. However, additional research is required to provide more validated findings for a more exhaustive comparison since the experiments were done on different Windows 11 operating system versions.

As previously noted, this research serves as a starting point for forensic investigations of WSA. The available documentation on the Windows Subsystem for Android is rather limited, and the fundamental understanding of this subsystem is fragmented across various open sources. Hence, the author intended to establish a comprehensive knowledge base related to the WSA to fill this gap.

Again, it is vital to highlight that the experiment was conducted in Estonia, a region that has yet to be included in the supported areas for the Amazon Appstore. Future research could analyse WSA downloaded from the official Amazon Appstore and explore other aspects, such as incorporating memory forensics, which can provide additional valuable artifacts.

References

[1] “Windows Subsystem for Android^{TM},” Feb. 13, 2023. https://learn.microsoft.com/en-us/windows/android/wsa/ (accessed Mar. 25, 2023).

[2] “What is Windows Subsystem for Linux,” Aug. 12, 2022. https://learn.microsoft.com/en-us/windows/wsl/about (accessed Mar. 25, 2023).

[3] “Android Statistics (2023),” *Business of Apps*. https://www.businessofapps.com/data/android-statistics/ (accessed Mar. 25, 2023).

[4] “Amazon Appstore: number of available apps by quarter 2022,” *Statista*. https://www.statista.com/statistics/307330/number-of-available-apps-in-the-amazon-appstore/ (accessed Mar. 25, 2023).

[5] “Windows Subsystem for Android^{TM} Release Notes,” Mar. 21, 2023. https://learn.microsoft.com/en-us/windows/android/wsa/release-notes (accessed Mar. 25, 2023).

[6] “Countries and regions that support Amazon Appstore on Windows - Microsoft Support.” https://support.microsoft.com/en-us/windows/countries-and-regions-that-support-amazon-appstore-on-windows-d8dd17c7-5994-4187-9527-ddb076f9493e (accessed Mar. 26, 2023).

[7] W. I. Blog, “Introducing Android^{TM} Apps on Windows 11 to Windows Insiders,” *Windows Insider Blog*, Oct. 20, 2021. https://blogs.windows.com/windows-insider/2021/10/20/introducing-android-apps-on-windows-11-to-windows-insiders/ (accessed Mar. 26, 2023).

[8] “Everything You Should Know About Windows Subsystem for Android in Windows 11,” May 23, 2022. https://learndelphi.org/everything-you-should-know-about-windows-subsystem-for-android-in-windows-11/ (accessed Apr. 03, 2023).

[9] A. Mohammed, “How to set up Windows Subsystem for Android on your Windows 11 PC,” *Android Police*, Sep. 18, 2022. https://www.androidpolice.com/set-up-wsa-windows-11-android-apps/ (accessed Mar. 26, 2023).

[10] “Aurora OSS / AuroraStore · GitLab,” *GitLab*, Sep. 25, 2022. https://gitlab.com/AuroraOSS/AuroraStore (accessed Mar. 28, 2023).

[11] S. Hazarika, “WSA PacMan is a GUI package manager and package installer for Windows Subsystem for Android,” *XDA Developers*, Dec. 23, 2021. https://www.xda-developers.com/wsa-pacman-gui-package-manager-windows-subsystem-for-android/ (accessed Mar. 26, 2023).

[12] “APKMirror - Free APK Downloads - Free and safe Android APK downloads,” *APKMirror*. https://www.apkmirror.com/ (accessed Mar. 28, 2023).

[13] “Please list use cases of WSA,” *r/Windows11*, Oct. 23, 2021. www.reddit.com/r/Windows11/comments/qe8036/please_list_use_cases_of_wsa/ (accessed Apr. 03, 2023).

[14] “Comparing WSL Versions,” Mar. 20, 2023. https://learn.microsoft.com/en-us/windows/wsl/compare-versions (accessed Apr. 03, 2023).

[15] “Run Linux GUI apps with WSL,” Mar. 20, 2023. https://learn.microsoft.com/en-us/windows/wsl/tutorials/gui-apps (accessed Apr. 03, 2023).

[16] R. Lee, C. Tilbury, and O. Carroll, *FOR500 Windows Forensic Analysis*, H01_01. in Registry Analysis, Application Execution, and Cloud Storage Forensics, no. 500.2. SANS Institute, 2022.
[17] “ShimCache.” https://forensafe.com/blogs/shimcache.html (accessed Apr. 13, 2023).
[18] “AmCache Blog.” https://forensafe.com/blogs/AmCache.html (accessed Apr. 13, 2023).
[19] J. Buric and D. Delija, “Challenges in network forensics,” in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2015, pp. 1382–1386. doi: 10.1109/MIPRO.2015.7160490.
[20] “Wireshark · About,” *Wireshark*. https://www.wireshark.org/ (accessed Apr. 05, 2023).
[21] “CAINE Live USB/DVD - computer forensics digital forensics.” https://www.caine-live.net/ (accessed Apr. 12, 2023).
[22] “Arsenal Recon.” https://arsenalrecon.com/products/arsenal-image-mounter (accessed Apr. 12, 2023).
[23] R. Lee, C. Tilbury, and O. Carroll, *FOR500 Windows Forensic Analysis*, H01_01. in Digital Forensics and Advanced Triage, no. 500.1. SANS Institute, 2022.
[24] Eric, “MFTECmd.” Apr. 03, 2023. Accessed: Apr. 12, 2023. [Online]. Available: https://github.com/EricZimmerman/MFTECmd
[25] Eric, “PECmd.” Apr. 10, 2023. Accessed: Apr. 12, 2023. [Online]. Available: https://github.com/EricZimmerman/PECmd/blob/99abe48e27240dae1012692f02a4ae1299f09a80/README.md
[26] “Windows event log analysis software, view and monitor system, application and security event logs — FSPro Labs.” https://eventlogxp.com/ (accessed Apr. 12, 2023).
[27] P. Boigner and R. Luh, “WSL2 Forensics: Detection, Analysis & Revirtualization,” in *The 17th International Conference on Availability, Reliability and Security*, in ARES 2022. Vienna, Austria: Association for Computing Machinery, 2022. doi: https://doi.org/10.1145/3538969.3544439.
[28] T. Viitmaa, “Forensic analysis of Windows Subsystem for Linux on Windows 11,” Tallinn University of Technology, Tallinn, 2022. Accessed: Jan. 11, 2022. [Online]. Available: https://digikogu.taltech.ee/en/item/0631248d-7129-46d1-91f9-62728984e5f5
[29] “Linux under WSL2 can be leaking - Blog,” *Mullvad VPN*. https://mullvad.net/en/blog/2020/9/30/linux-under-wsl2-can-be-leaking/ (accessed Apr. 04, 2023).
[30] A. Matadar, “Investigating Windows Subsystem for Linux (WSL) Endpoints,” presented at the The Digital Forensic Research Conference, 2020. Accessed: Jun. 11, 2022. [Online]. Available: https://dfrws.org/wp-content/uploads/2020/07/2020_USA_pres-investigating_windows_subsystem_for_linux_wsl_endpoints.pdf
[31] Z. Li, “Comparison between common virtualization solutions: VMware Workstation, Hyper-V and Docker,” in *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, Nov. 2021, pp. 701–707. doi: 10.1109/ICFTIC54370.2021.9647226.
[32] K. Allix, Q. Jerome, T. Bissyandé, J. Klein, R. State, and Y. Le Traon, “A Forensic Analysis of Android Malware -- How is Malware Written and How it

Could Be Detected?," presented at the Proceedings - International Computer Software and Applications Conference, Jul. 2014. doi: 10.1109/COMPSAC.2014.61.

[33] A. Sarkar, A. Goyal, D. Hicks, D. Sarkar, and S. Hazra, "Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems," in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Dec. 2019, pp. 73–79. doi: 10.1109/I-SMAC47947.2019.9032440.

[34] Y. Kanchhal and M. Se, "An Enhanced Solution for Detection of Injected Android Malware Application," Jul. 2022, pp. 1–11. doi: 10.1109/ICSES55317.2022.9914175.

[35] "Android Debug Bridge (adb)," *Android Developers*. https://developer.android.com/tools/adb (accessed Apr. 05, 2023).

[36] "What is the Hyper-V Virtual Switch and How Does it Work?," *Altaro DOJO | Hyper-V*, Sep. 26, 2019. https://www.altaro.com/hyper-v/the-hyper-v-virtual-switch-explained-part-1/ (accessed May 08, 2023).

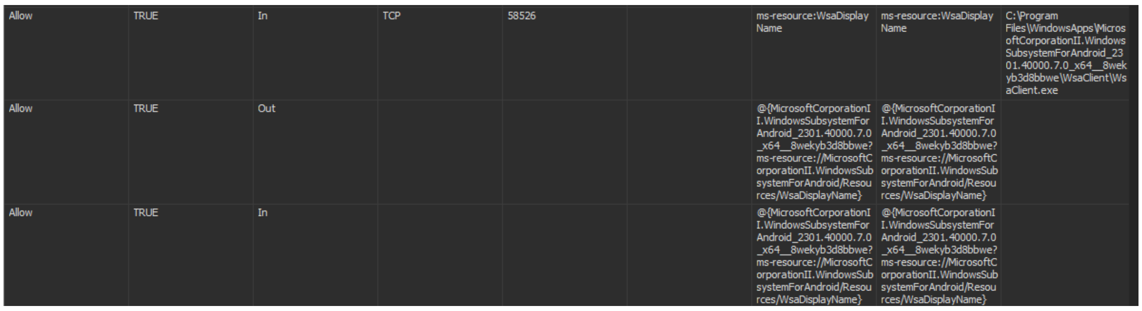
[37] "Welcome to vPlanet Technologies : : : Banaglore." http://vplanetech.com/hyperv.html (accessed May 08, 2023).

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

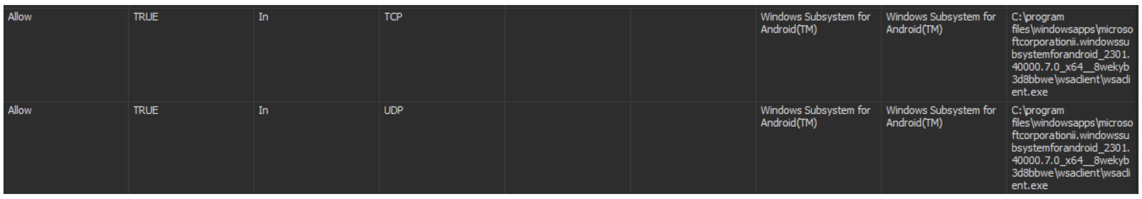
1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes 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 Rules

Evidence of Firewall rules under SYSTEM hive. ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\FirewallRules



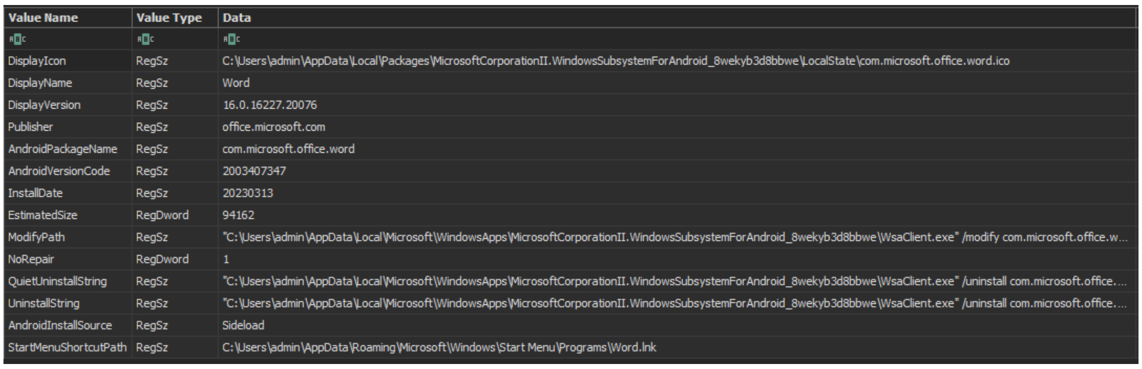
Allow	TRUE	In	TCP	58526		ms-resource:WsaDisplayName	ms-resource:WsaDisplayName	C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe\WsaClient\WsaClient.exe
Allow	TRUE	Out				@{MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe?ms-resource://MicrosoftCorporationII.WindowsSubsystemForAndroid/Resources/WsaDisplayName}	@{MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe?ms-resource://MicrosoftCorporationII.WindowsSubsystemForAndroid/Resources/WsaDisplayName}	
Allow	TRUE	In				@{MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe?ms-resource://MicrosoftCorporationII.WindowsSubsystemForAndroid/Resources/WsaDisplayName}	@{MicrosoftCorporationII.WindowsSubsystemForAndroid_2301.40000.7.0_x64__8wekyb3d8bbwe?ms-resource://MicrosoftCorporationII.WindowsSubsystemForAndroid/Resources/WsaDisplayName}	



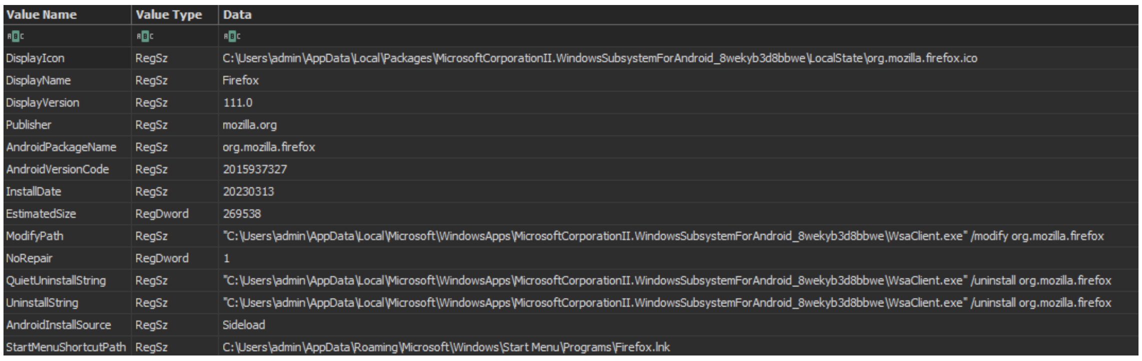
Allow	TRUE	In	TCP			Windows Subsystem for Android(TM)	Windows Subsystem for Android(TM)	C:\program files\windowsapps\microsoftcorporationii.windowssubsystemforandroid_2301.40000.7.0_x64__8wekyb3d8bbwe\wsaclient\wsaclient.exe
Allow	TRUE	In	UDP			Windows Subsystem for Android(TM)	Windows Subsystem for Android(TM)	C:\program files\windowsapps\microsoftcorporationii.windowssubsystemforandroid_2301.40000.7.0_x64__8wekyb3d8bbwe\wsaclient\wsaclient.exe

Appendix 3 – Installed Applications

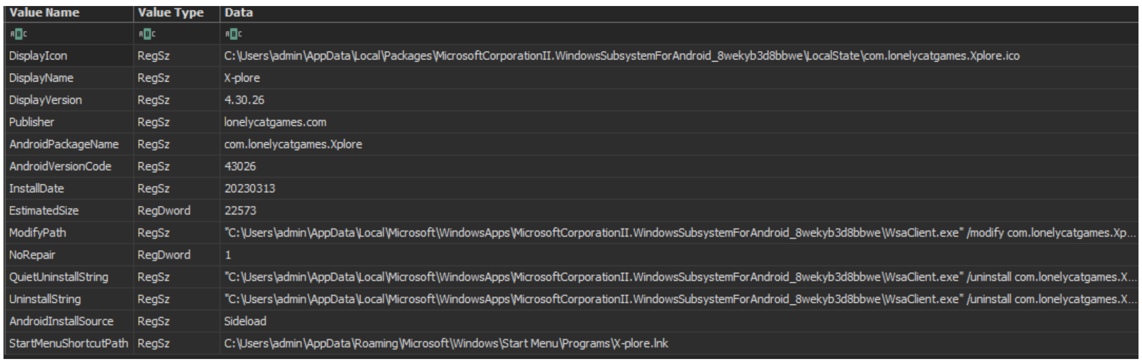
Evidence of installed applications under NTUSER.dat hive Software\Microsoft\Windows\CurrentVersion\Uninstall



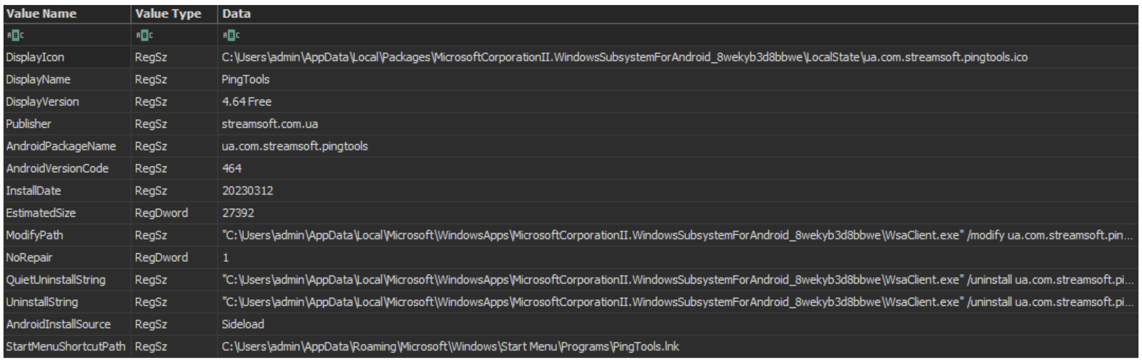
Value Name	Value Type	Data
DisplayIcon	RegSz	C:\Users\admin\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalState\com.microsoft.office.word.ico
DisplayName	RegSz	Word
DisplayVersion	RegSz	16.0.16227.20076
Publisher	RegSz	office.microsoft.com
AndroidPackageName	RegSz	com.microsoft.office.word
AndroidVersionCode	RegSz	2003407347
InstallDate	RegSz	20230313
EstimatedSize	RegDword	94162
ModifyPath	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /modify com.microsoft.office.w…
NoRepair	RegDword	1
QuietUninstallString	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /uninstall com.microsoft.office.…
UninstallString	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /uninstall com.microsoft.office.…
AndroidInstallSource	RegSz	Sideload
StartMenuShortcutPath	RegSz	C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Word.lnk



Value Name	Value Type	Data
DisplayIcon	RegSz	C:\Users\admin\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalState\org.mozilla.firefox.ico
DisplayName	RegSz	Firefox
DisplayVersion	RegSz	111.0
Publisher	RegSz	mozilla.org
AndroidPackageName	RegSz	org.mozilla.firefox
AndroidVersionCode	RegSz	2015937327
InstallDate	RegSz	20230313
EstimatedSize	RegDword	269538
ModifyPath	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /modify org.mozilla.firefox
NoRepair	RegDword	1
QuietUninstallString	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /uninstall org.mozilla.firefox
UninstallString	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /uninstall org.mozilla.firefox
AndroidInstallSource	RegSz	Sideload
StartMenuShortcutPath	RegSz	C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Firefox.lnk



Value Name	Value Type	Data
DisplayIcon	RegSz	C:\Users\admin\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalState\com.lonelycatgames.Xplore.ico
DisplayName	RegSz	X-plore
DisplayVersion	RegSz	4.30.26
Publisher	RegSz	lonelycatgames.com
AndroidPackageName	RegSz	com.lonelycatgames.Xplore
AndroidVersionCode	RegSz	43026
InstallDate	RegSz	20230313
EstimatedSize	RegDword	22573
ModifyPath	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /modify com.lonelycatgames.Xp…
NoRepair	RegDword	1
QuietUninstallString	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /uninstall com.lonelycatgames.X…
UninstallString	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /uninstall com.lonelycatgames.X…
AndroidInstallSource	RegSz	Sideload
StartMenuShortcutPath	RegSz	C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\X-plore.lnk



Value Name	Value Type	Data
DisplayIcon	RegSz	C:\Users\admin\AppData\Local\Packages\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\LocalState\ua.com.streamsoft.pingtools.ico
DisplayName	RegSz	PingTools
DisplayVersion	RegSz	4.64 Free
Publisher	RegSz	streamsoft.com.ua
AndroidPackageName	RegSz	ua.com.streamsoft.pingtools
AndroidVersionCode	RegSz	464
InstallDate	RegSz	20230312
EstimatedSize	RegDword	27392
ModifyPath	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /modify ua.com.streamsoft.pin…
NoRepair	RegDword	1
QuietUninstallString	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /uninstall ua.com.streamsoft.pi…
UninstallString	RegSz	"C:\Users\admin\AppData\Local\Microsoft\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForAndroid_8wekyb3d8bbwe\WsaClient.exe" /uninstall ua.com.streamsoft.pi…
AndroidInstallSource	RegSz	Sideload
StartMenuShortcutPath	RegSz	C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\PingTools.lnk