WSL2 Forensics: Detection, Analysis & Revirtualization

Philipp Boigner
is191833@fhstp.ac.at
St. Pölten University of Applied Sciences
Austria

Robert Luh
robert.luh@fhstp.ac.at
St. Pölten UAS & University of Vienna
Austria

ABSTRACT

The development and integration of the Windows Subsystem for Linux, version 2 (WSL2) into Microsoft's operating systems has brought together two worlds that were, from a consumer's perspective, previously disjunct. This comes with new challenges for incident handling and computer forensics in particular, since workflows rarely had to consider both ecosystems at time same time. With WSL2 now becoming an integral part of Windows 10 and 11, tools and techniques have to be revisited with the new environment in mind.

In this paper, we look at the detection, acquisition and post-mortem analysis of WSL2 instances. We explore through experimentation how WSL2 guests can be quickly identified and provide investigators with an easy means to automate the process. Since it can also be helpful to an investigation to revirtualize an acquired image, the process of getting up and running a WSL2 instance on another host is discussed as well. This is complemented by a surface analysis of the extracted data, where we assess whether current open-source suites are compatible with Microsoft's take on Linux.

Ultimately, this work provides a concise guide for investigators dealing with WSL2 instances and updates the current state-of-the-art, which predominantly focuses on the first iteration of WSL.

CCS CONCEPTS

• **Applied computing** → **System forensics**; **Investigation techniques**.

KEYWORDS

digital forensics, windows, linux, wsl, virtual machine

ACM Reference Format:
Philipp Boigner and Robert Luh. 2022. WSL2 Forensics: Detection, Analysis & Revirtualization. In *The 17th International Conference on Availability, Reliability and Security (ARES 2022), August 23–26, 2022, Vienna, Austria.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3538969.3544439

1 INTRODUCTION

Due to its market share of around 76% [24], Microsoft Windows remains the most widely used operating system. This is also apparent in computer forensics, where – outside of mobile forensics – investigators most often deal with images of Windows installations using the NTFS file system. Linux, as ecosystem predominately found in server environments, used to be largely disconnected from this consumer-centric world. The conception of the Windows Subsystem for Linux (WSL) changed this: It is now possible to run various Linux distributions within a Windows 10 or 11 installation. While WSL1 was limited in its emulation capabilities, version 2 of WSL embraced a virtual machine approach [17]. It allows the seamless execution of Linux apps with few limitations and blurs the line between host and guest operating system.

This comes not only with advantages for the user, but also with a number of issues and challenges in regard to incident handling. Firstly, the attack surface of Windows has increased significantly due to this development. Several types of WSL malware have already been discovered, some still in development, that are primed to run on WSL instances [12]. This will lead to a rise in demand for forensic investigations in response to breaches and other cyber-attacks that focus on Linux. Secondly, suspect users themselves now have a wider range of tools at their disposal, which leave artifacts not only in Windows, but within WSL as well. Both will have to be analyzed to create a complete picture of malicious user activity.

Herein lies the motivation for this work. We aim to investigate how well WSL2 forensics integrates into established processes and which tools can be used to extract and analyze images of WSL2 installations. We argue that most scientific work to date focuses in the first iteration of the subsystem [10, 11, 14], which was built on a different technical foundation than its successor. We want provide forensics experts with a means to quickly identify WSL2 instances within Windows and provide a guideline for its acquisition and analysis, while exploring revirtualization as additional avenue of investigation.

Specifically, we endeavor to answer the following questions:

(1) How can forensic investigators easily detect and identify WSL2 instances on a Windows host?
(2) How can investigators create forensically sound WSL2 images for analysis and revirtualization?
(3) How can extracted images most effectively be analyzed using open-source tools?

We hypothesize that common Windows artifact categories (Registry, event logs, jump lists) can be utilized to answer question 1 and that this process can be automated for faster and more accurate results. For extraction, we want to confirm the assumption that not only we can acquire an image from a WSL2 disk, but also revirtualize it for further live analysis. Question 3 comes with the hypothesis that current open-source forensics suites support WSL2 images out of the box, provided the image follows the Hyper-V convention of using the default `.vhdx` format.

The remainder of this paper is structured as follows: Section 2 provides a brief overview of both versions of WSL and discusses the artifact categories investigated for our WSL detection tool. Section 3 provides general information about our experiments and the

ARES 2022, August 23–26, 2022, Vienna, Austria

ACM ISBN 978-1-4503-9670-7/22/08.
https://doi.org/10.1145/3538969.3544439

technical setup, whereas sections 4 and 5 present specific findings in regard to WSL detection as well as acquisition, analysis, and revirtualization. Finally, Section 6 discusses related work, followed by concluding remarks.

2 BACKGROUND

This chapter discusses the two major version of WSL as well as its currently employed Linux Kernel developed by Microsoft. Lastly, we briefly talk about the artifact categories investigated for WSL2 detection.

2.1 Windows Subsystem for Linux

2.1.1 Version 1. The first version of the Windows Subsystem for Linux was integrated into Windows 10 by Microsoft in March 2016 [3]. The software was created primarily for developers and allowed Linux programs to be run within Windows. This eliminated the need to use resource-heavy virtual machines on the Windows host. In order for Linux syscalls to function, an intermediate layer, called LXSS, was developed. This layer translates syscalls from the Linux environment into syscalls that can be interpreted by the Windows kernel [7]. This was realized through pico processes, which are started whenever a Linux program is executed.

WSL primarily consists of the following components:

- A session manager executed in user mode,
- the pico process providers `lxss.sys` and `lxcore.sys`, which emulate the Linux kernel and translate syscalls, and
- the pico processes that host the native Linux apps.

A restriction which exists with version 1 of WSL is the missing compatibility with a number of aforementioned syscalls; not all Linux syscalls can be compiled with the pico process providers. Also, syscalls in WSL may exhibit different behavior than they would inside a native Linux environment [11].

2.1.2 Version 2. WSL2, like version 1, is available in Windows 10 and up. Contrary to WSL1, version 2 (released in May 2019) virtualizes a Linux kernel using Hyper-V [17]. It is built upon a real Linux kernel (see below), which improves file system performance and eliminates syscall incompatibilities as well as suspicious deviations in behavior [17]. At the same time, the list of available Linux operating systems has been expanded significantly, ranging from Debian and Ubuntu to openSUSE, Kali, and others.

Like other virtualization technologies, Hyper-V is used to run multiple (operating) systems on one hardware device. To accomplish this, the software has the ability to pass hardware resources to virtual systems or to share them between them. Hyper-V is available in Windows Server, Windows 10 Pro, Enterprise, and Education and can simply be toggled on though the OS's features configuration tool. Note that some differences in capabilities exist between the Server and Windows 10 versions of Hyper-V [6]. It is possible to freely upgrade or downgrade existing WSL instances [16].

2.2 Microsoft's Linux Kernel

Microsoft obtained its WSL kernel directly from kernel.org and configured it to include a number of optimizations that focus on reducing startup time and memory consumption. Only necessary functions are used for a minimal set of supported devices. This results in a compact, lightweight kernel that is optimized for WSL2. As mentioned before, this kernel is a replacement for the original WSL1 emulation architecture.

Another peculiarity of the kernel is the way it is deployed. Unlike an upgrade process on a Linux system, where updates are applied via the package manager [29], a kernel update of WSL2 instances is possible through Windows Update [9].

The developed kernel is available in a GitHub repository and is actively developed by Microsoft[1]. For expert users, it is possible to customize the kernel for a custom configuration of their WSL environment [15]. In our experiments, we utilized the default kernel (version 5.10.16) without any modifications.

2.3 Plan 9 File Protocol

The 9P protocol was developed by Bell Labs for the Plan 9 operating system and comprises a traditional client-server architecture. A Plan 9 server can provide multiple hierarchical file systems that can be used by Plan 9 processes [1]. WSL2 includes a 9P server that is started with the `init` process when the distribution is run. Using services and drivers, Windows can establish a client connection to the 9P server to enable file sharing. The interface used by Windows to access Linux data is implemented via a file share named `wsl$`. This share can be used from the file explorer to directly access Linux file system. Linux metadata and file permissions are applied [18].

2.4 Artifact Categories

Following our hypotheses for detecting WSL2 instances, we focus on a number of Windows artifact categories, which are briefly introduced below. Refer to Section 4 to see which of them are particularly useful for detecting WSL2 Linux distributions.

2.4.1 Windows Registry. The Windows Registry is the central configuration store of the Windows OS and one of the most important data sources for Windows forensics. It contains system settings as well as application and user data [4]. This data can be modified by applications using the Windows API. The Registry contains different keys, subkeys and values. Contentual separation is realized though so-called hives [27], which are stored in files that can be found in the `%SystemRoot%\System32\Config` folder. Examples for Registry artifacts include the UserAssist key [5] and various application installation paths.

2.4.2 Prefetch. The Windows Prefetch mechanism increases the speed with which programs are started by storing the files necessary for their launch. If an application that has already been executed is called again, the data in the respective Prefetch file is accessed. The files are located in the `C:\Windows\Prefetch` folder. The name of the Prefetch file is composed of original program's name, a hash of the path and the `.pf` extension. The information included in the entry includes full filename, libraries loaded, various 'last used' timestamps, and the execution count of the program [20].

2.4.3 Jumplists. Windows jumplists make it easier to open regularly used programs and their associated files. There are two types of jumplists: Automatic and custom ones. They are located in the `%userprofile%` folder and the file name is composed of

[1] https://github.com/microsoft/WSL2-Linux-Kernel

an AppID and the file extension `automaticDestinations-ms` or `customDestinations-ms` [23].

2.4.4 AmCache. The AmCache [13] stores metadata related to Windows program execution and installation on Windows 7 and above. It succeeded the Application Compatibility Cache (shimcache) first implemented in Windows XP and utilizes two files: a `.bcf` file, called `RecentFileCache.bcf`, and a Registry hive called `AmCache.hve`, both of which are found in found in `%WinDir%\AppCompat\Programs\`. The information stored about shimmed applications is similar to what is found in the registry.

3 METHODOLOGY

This work follows a constructive, experiment-driven approach according to Vaishnavi and Kuechler [26]. First, we identify the artifact categories that provide information about current WSL installations. Since our review of related work (see Section 6) has shown that no automated tools yet exist, we provide a plugin to the tool RegRipper in addition to a guideline for manual WSL2 artifact detection. All data acquisition processes were tested on a virtual machine using several WSL2 distributions and compared to a machine without WSL2 installed to highlight the differences.

For acquisition and analysis, both stages of the forensic investigation were conducted within the testing environment detailed below. Several user artifacts were placed within the Linux file system to be acquired in order to challenge current forensics suites. The experiments included the revirtualization of an extracted virtual hard drive for applied live forensics.

The experiments aimed at learning more about the overall process of investigating WSL2 virtual machines and were guided at all times by the research questions and hypotheses presented in Section 1. Please refer to sections 4 and 5 for specific information about the technical procedure.

3.1 Lab Setup

3.1.1 Virtual Machines. The experimental setup was composed of several virtual machines running on a Manjaro Linux host. A common 'shared folder' was used to collect the generated data, reports, and images of the machines. Specifically, we used the following infrastructure:

- Win10_WSL: This machine contained a current Windows 10 installation with WSL2. It served as a data source for all imaging and analysis tasks.
- Win10_NoWSL: This baseline machine ran the same version of Windows 10 as Win10_WSL, but did not contain the subsystem for comparison purposes.
- Win7_Analysis: Here, the forensic investigation of the acquired images was conducted. This machine contained various tools such as RegRipper, WinPrefetchView, JumpListsView, and Registry Explorer for WSL artifact analysis, FTK Imager for extraction purposes, and Autopsy and FTK for user data analysis of the WSL images themselves. Refer to Section 4 for more information.
- Win10_ReVirt: This machine, which contained an initially 'empty' WSL2 environment, was used as host for revirtualizing the extracted WSL image taken from Win10_WSL.

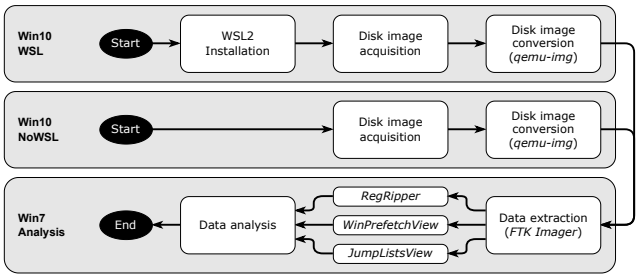

Figure 1: Process for WSL2 artifact discovery. Note that the RegRipper stage also included the investigation of the Windows Registry using Registry Explorer.

In the following, we discuss the specific process of identifying WSL2 instances, our detection plugin, and the process of acquiring, analyzing, and revirtualizing WSL2 images.

4 WSL2 DETECTION

This section describes the experiment conducted to determine WSL2 artifacts and presents the detection tool created to automate the process.

4.1 Procedure

In preparation for the analysis of the "Win10_WSL" image, some initial preparation was required. First, WSL2 was installed using the respective PowerShell command, `wsl --install`. We primarily used Debian for our experiments, which was set up via the `-d debian` argument.

To create an image of the machine for analysis on "Win7_Analysis", the virtual machine was shut down and its disk file (`.vmdk`) was located and converted to a raw image using `qemu-img`. This image was then imported into the various tools mentioned above and discussed below. The image of "Win10_NoWSL" was created the same way to serve as a comparison baseline.

In the following, we introduce the tools used for WSL2 detection and summarize the results. Refer to Figure 1 for a visual overview.

4.1.1 RegRipper. The first part of the artifact discovery process was conducted using RegRipper[2]. RegRipper is an open source tool for extracting and parsing Windows Registry data. It is written in Perl and can extract and display the keys, values and data of the Registry in a user-friendly manner. In some cases it was necessary to open the Registry files to directly access its hierarchical database. For this purpose, Registry Explorer[3] was used.

4.1.2 WinPrefetchView. WinPrefetchView by NirSoft[4] is a tool to read Windows Prefetch data, which includes all information listed in Section 2. It supports exporting its findings into text files for easier processing.

4.1.3 JumpListsView. Similar to WinPrefetchView, JumpListsView by NirSoft is a tool that processes jumplist entries and displays them

[2]https://github.com/keydet89/RegRipper3.0
[3]https://ericzimmerman.github.io/
[4]https://www.nirsoft.net/utils/win_prefetch_view.html

in tabular fashion. It extracts and exports – among other things – file names, paths, various timestamps and file attributes.

4.1.4 Event Viewer. To read the Windows Event logs, the internal viewer of Windows 10 was used. For easier parsing, command line tools like EvtxECmd[5] could be utilized instead.

4.2 Results

This subsection presents all our findings in regard to WSL detection. Artifact categories discussed in other works (e.g., AmCache) are summarized for completeness's sake.

4.2.1 Windows Registry. We identified the RegRipper plugins `apppaths`, `clsid`, `msis`, `shimcache`, `appcompatcache_tln`, `services`, and `svc_plus` as most relevant in terms of WSL artifact detection. While their outputs are rarely indisputable – which promoted us to create our own plugin to easily identify which WSL instances are set up on the machine in question – the sum of results speaks a clear language.

For example, `apppaths` is able to locate the paths to the distribution executable, which may look like this:

```
|REG|||App Paths - debian.exe - C:\Program Files\
WindowsApps\TheDebianProject.DebianGNULinux_(...)_
x64__76v4gfsz19hv4\debian.exe
```

Through `cslid`, which extracts the unique identifiers that belong to COM class objects [28], we can identify certain WSL2 components:

```
{615a13be-241d-48b1-89b0-8e1d40ffd287}  WslClient
LastWrite: Thu Jan  1 00:00:00 1970 Z
InprocServer32: C:\WINDOWS\System32\lxss\wslclient.dll
```

Within the application compatibility cache (extracted using the `shimcache` and `appcompatcache_tln` plugins), another relevant artifact can be found. This is related to the aforementioned AmCache. Note that the entry for `wsl.exe` is not unique to machines with WSL2 installed; it was also found on the baseline machine.

```
1637265328|REG|||M... AppCompatCache - C:\WINDOWS\
system32\lxss\wslhost.exe
```

Through the `services` plugin, it is possible to locate LXSS artifacts (abbreviated):

```
Name      = LxssManagerUser
Display   = (...)\lxss\wslclient.dll,-102
ImagePath = (...)\svchost.exe -k LxssManagerUser -p
Type      = 0x0
Start     = Manual
```

While `scv` and `scvdll` do not produce results unique to the WSL machine, `svc_plus` extracts LXSS manager user artifacts related to the above.

Other, manually extractable Registry artifacts e.g., include the WSL bash environment variable `BASHENV` found in `HKCU\Environment`, and the `UserAssist` key that can be extracted and decoded using UserAssistView[6].

[5]https://ericzimmerman.github.io/
[6]https://www.nirsoft.net/utils/userassist_view.html

4.2.2 Prefetch. WSL2 instances are optimized through Prefetch like any other Windows application, provided it has been started at least once. The entry for a Debian installation extracted by `WinPrefetchView` may look like this:

```
Filename          : DEBIAN.EXE-1BE9E921.pf
Created Time      : 17.01.2022 16:56:17
Modified Time     : 27.01.2022 12:27:32
File Size         : 6.045
Process EXE       : DEBIAN.EXE
Process Path      : C:\(...)\debian.exe
Run Counter       : 10
Last Run Time     : 27.01.2022 12:27:31
                    (...)
Missing Process   : No
```

4.2.3 Jumplists. Contrary to our expectations, the investigated jumplists did not contain artifacts pointing at WSL or our Debian installation.

4.2.4 Event Log. Numerous WSL2 artifacts can potentially be found in the event log. These refer to the setup process and updates (e.g., event ID 16, 1033, 1035, 1040, 1042, 11707), service interaction (e.g., ID 7, 9, 13), auditing (e.g., ID 4907), or Hyper-V networking events with IDs 5, 7, 9, 62, , 232, 233, and 264.

A link to example outputs for many of these events can be found in the appendix.

4.2.5 Other artifacts. Additional WSL2 indicators can be found in the AmCache (extractable with the e.g., AmcacheParser[5] by Eric Zimmerman), and in the NTFS Master File Table (MFT) directly [19].

4.3 Implementation

Since only the identified Registry artifacts can be considered mostly deterministic, we have focused on them for our implementation of a dedicated parser that collects information from across the Registry. Our objective was to easily identify which WSL2 distributions are installed and summarize all relevant information in one place. For this purpose, the RegRipper plugin `wsldetect` was created.

The tool retrieves the following information from respective Registry sources:

- Kernel version (LXSS)
- Kernel installer information
- Application paths (App Paths)
- WSL2 software packages

The data is processed and formatted for user-friendly output, which may look like this (abbreviated; limited to one Registry key each):

```
wsldetect v.20220210
(Software) Detector Plugin for WSL2 on Win10
=============
SOFTWARE: get wsl kernel version:
Registry path: Microsoft\Windows\CurrentVersion\Lxss
KernelVersion: 5.10.16
=============
SOFTWARE: get installer of the Linux kernel:
Registry path: (...)
Installer Name: Windows Subsystem for Linux
```

```
 Update by Microsoft Corporation
 Installer ID: E752FE635D127F4448157029A3C864E5
 Version: 5.10.16
 Installed on: 20211005
=============
SOFTWARE: get distro in appaths:
Registry path: (...)
Key name: debian.exe
 Key value:(...)\debian.exe
 Last modified: 2022-01-13 16:37:08
=============
SOFTWARE: get distro in software packages:
Registry path: (...)\PackageRepository\Packages
Keyname: 46932SUSE.openSUSELeap(...)__022rs5jcyhyac
 Key value: (...)
 Last modified: 2022-03-07 15:37:33
====================================
Result Summary:
---------------
Kernel Version: 5.10.16
Kernel Install Date: 20211005
Indicators in Appaths: 8
Indicators in Software Packages: 26
------------------------------------
Total Indicators of WSL: 34
------------------------------------
Following WSL distros were detected:
 debian.exe
 kali.exe
 openSUSE-42.exe
 (...)
```

The tool was tested on an extracted WSL2 disk image and launched both through RegRipper's CLI and graphical interface for images of the "Win10_WSL" and "Win10_NoWSL" machines. While some few artifacts can be found on Windows 10 even without WSL2 installed, the results were still unambiguous and sped up the detection process significantly.

The download link of the `wsldetect` plugin can be found in the appendix.

4.4 Discussion

Summarizing the results of the investigation with RegRipper, we could see that the detection of WSL does not always yield unequivocal results, since even machines without WSL2 installed will contain numerous artifacts pointing at its possible existence. Our dedicated plugin helped close that gap and provides clearer output that can be used to identify WSL2 with little to no room of error, thereby addressing and answering research question 1. The four registry sources discussed in Section 4.3 have proven to be most accurate in regard to WSL2 detection.

No traces of WSL instances were found in the Windows jumplists data, refuting that hypothesis. On the other hand, Prefetch data can be used as an indicator of WSL installations. Besides the presence of files about the subsystem, information about the time of execution can be extracted. Note that these artifacts may not be present for e.g., newly installed distributions that were never launched.

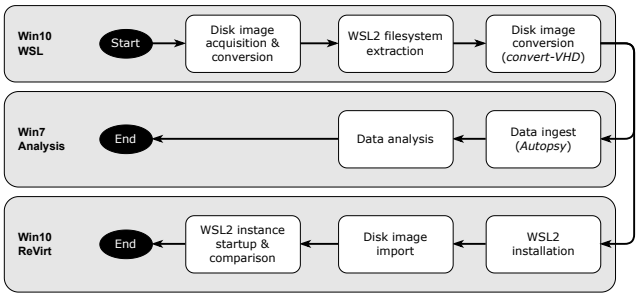

Figure 2: Process for WSL2 image acquisition, analysis, and revirtualization.

It has been shown that traces of WSL can also be detected in the Windows event logs. For this purpose, refer to the event log IDs listed above for easy filtering. Indicators are found in the application, security, installation, and system logs.

The presence of artifacts outside the Windows Registry offers room for improvement in regard to the implementation of a multi-source WSL2 detection and identification tool that also considers recent app use and Hyper-V-specific artifacts like network interface control events. The detection of currently unsupported or custom distributions will have to be investigated as well. For distributions available through the Windows marketplace at the time of writing, however, our approach provides reliable results.

5 FILE SYSTEM ACQUISITION & ANALYSIS

In this section, we discuss the acquisition, analysis, and revirtualization of WSL2 Linux disk images and evaluate the compatibility to the wide-used open source forensics suite TSK/Autopsy and the commercial Forensic Toolkit (FTK).

5.1 Procedure

In this experiment, we stepped through numerous stages to mirror a forensically sound investigation process. The process overview is depicted in Figure 2.

Prior to imaging, we placed numerous evidence items within the Linux file system, which we later endeavored to locate. This e.g. included picture files, documents, as well as text and archive files. In addition, we used the terminal to execute numerous commands, install package updates, ping a website, mount devices, and more.

In stage 2, we located the virtual hard disk file associated to the WSL2 instance. Since the subsystem follows the Hyper-V default, the disk image is of the format `.vhdx` with an initial size of 256 GiB and a maximum capacity of 64 TiB [25]. In our experiment, the hard drive in question held 699 MiB of data.

Next, we shut down the "Win10_WSL" host machine to acquire the image like in experiment 1, before accessing it in FTK Imager to extract the Debian disk file. The respective files for the ext4 file system are named `ext4.vhdx` and located in `%LOCALAPPDATA%\Packages\<Distribution>\LocalState`.

Since many forensic tools do not yet support the `.vhdx` format, we converted the file to the legacy `.vhd` format using `Convert-VHD`. The responsible PowerShell tool is available on Windows 10+ and requires the installation of the optional feature 'Hyper-V'.

After this step, we loaded the converted image into The Sleuth Kit (TSK) via its graphical 4.19.3 version of the Autopsy interface[7], processed it using all available ingest modules, and looked for the previously placed artifacts.

In preparation for revirtualization, it was necessary to install WSL and the same Linux distribution on the "Win10_ReVirt" machine. That process, as well as the results of our forensic investigation, is discussed in the next subsection.

5.2 Results

Fortunately from an investigator's perspective, our experiments did not surface any incompatibilities and largely confirmed our hypotheses. After conversion, Hyper-V's disk image files could be imported into Autopsy and FTK without issue. All previously placed user artifacts could easily be extracted in accordance with the tool's general capabilities in regard to ext4 images.

Revirtualization proved to be straightforward as well: Provided the same distribution and version of Linux was installed on the investigator's machine, `ext4.vhdx` could simply be placed in the aforementioned folder, replacing any existing disk images within. There were no subsequent issues starting up the copied WSL instances. On the contrary, we found that after copying the image onto `WSL_ReVirt`, no login prompt was displayed for the Linux user configured on the other machine upon startup, allowing us to access the data and interact with the virtual system (sans root privileges) without knowing any credentials. This auto-login of the designated main user is the set default behavior of WSL2 and cannot currently be changed easily.

5.3 Discussion

Extracting and revirtualizing WSL2 images can be done without expert knowledge and requires only a minimal set of tools, most of which are already used in conventional Linux forensics. Forensic integrity can be maintained by imaging the host computer's disk (either by shutting down the VM, or through a write blocker and COTS imaging software for native machines) and then extracting the Hyper-V image files stored for each WSL2 distribution. Thanks to the hassle-free process demonstrated in our experiments, we can answer research question 2 in the affirmative.

Analysis of WSL2 images can be performed out-of-the-box with both commercial and open-source tools such as Autopsy/TSK and Exterro's Forensic Toolkit (FTK). This answers research question 3: Forensics tools are able to effectively analyze WSL2 images. They face the same constraints and potential compatibility issues as conventional virtual machines or bare metal installations of the same distribution/file system.

Future work will include forensically investigating Microsoft's support for launching graphical applications through WSL2, which was recently incorporated into Windows 11. It stands to reason that a number of host artifacts may be created in the process that could be assessed outside of the virtual Linux instance.

[7] https://www.autopsy.com/

6 RELATED WORK

In the following, we chronologically present a number of works that either provide useful background information or highlight investigative processes and tools related to WSL or similar technologies. At the time of writing, there existed little scientific literature on WSL2 forensics outside of specialist talks or online discussion snippets.

The fundamental work by Carvey [4] revolves around the Windows Registry as a valuable resource for forensic investigations. In his work, Carvey regards autostart paths and artifacts pertaining to USB devices, just to name a few. Recently opened files are also analyzed. While this article and later book [5] provides many interesting insights into the topic, WSL was not yet in its scope.

In their book, Barrett and Kipper [2] discuss the impact of virtualization technology on digital forensics investigations. Their work provides a valuable overview of various tools and techniques, but is limited to systems released prior to 2010. Generally, research on Hyper-V forensics mostly focuses on memory analysis [8] and virtual machine introspection [21, 30], which is particularly relevant in cloud environments.

Lewis et al. [14] focus on memory analysis for WSL1. The researchers used reverse engineering to analyze its userland and kernel components to study the effect of WSL on existing Volatility plugins. Their goal was to ensure full support for memory forensics of machines running WSL. To this end, their paper investigates data structures relevant to WSL and presents a new Volatility analysis plugin called 'picoscan', which is able to handle the special pico processes utilized by WSL1 in order to provide workable results.

Riaz and Tahir [22] investigate the analysis of virtual machines in general. These VMs were populated with test data and analyzed using various tools. In addition to the images themselves, host artifacts such as the 'VMware Log File' are also highlighted, as are folders particularly relevant for forensic analysts. Similar to our work, Riaz and Tahir look for indicators of virtual environments. WSL, however, is not considered.

Ionescu et al. [10] explore different ways to run Linux programs on Windows. Different virtualization technologies are compared, including Virtualbox, Docker, and Cygwin in addition to WSL. Each solution is presented and subsequently tested for performance. While the authors' work provides an interesting overview of Linux virtualization, it does not include information about forensic artifacts hinting at their presence on a host.

In [11], Kochberger et al. examine the special features of the WSL system (version 1). Since the Windows Subsystem for Linux differs in architecture from a conventional Linux system, anomalous responses can occur when programs are executed. The researchers created a program that facilitates the detection of WSL1 on a Windows host. Unlike our work, the Kochberger et al. [11] only consider WSL1 and look at the processes during runtime, as opposed to post-mortem forensics.

In 2020, Matadar [19] presented his talk on WSL2 forensics at OSDFCon. He focused primarily on attacks on WSL2 instances and depicted a number of attack scenarios including reverse shells, remote file copy, living-off-the-land attacks, and command & control. Artifacts hinting at WSL installations were briefly touched and included e.g., UserAssist and the AmCache.

7 CONCLUSION

Thanks to Microsoft's changes to the WSL platform and the move to Hyper-V virtual machines, there are few inhibiting factors to performing a digital forensics investigation on machines hosting one or several WSL2 instances. We have presented an overview of artifacts that facilitate the detection of said guests and proposed a tool that helps make the process more accurate and speedy.

The feasibility of the extraction and analysis of WSL2 disk image files was shown, as was the ease of revirtualizing an extracted image on another machine. The fact that this process even allowed us to log in automatically to the virtualized Linux instance without knowledge of the main user's credentials, is bound to aid investigators in their endeavors.

With post-mortem analysis largely covered, future efforts may shift towards memory analysis and attacks on the guest operating systems themselves, as well as the impact of such threats on the WSL host.

REFERENCES

[1] Manual Page Archive. [n.d.]. Plan 9 Manual. http://man.cat-v.org/plan_9/5/intro

[2] Diane Barrett and Gregory Kipper. 2010. *Virtualization and forensics: A digital forensic investigator's guide to virtual environments.* Syngress.

[3] Windows Developer Blog. 2016. Run Bash on Ubuntu on Windows. https://blogs.windows.com/windowsdeveloper/2016/03/30/run-bash-on-ubuntu-on-windows/

[4] Harlan Carvey. 2005. The Windows Registry as a forensic resource. *Digital Investigation* 2, 3 (Sept. 2005), 201–205. https://doi.org/10.1016/j.diin.2005.07.003

[5] Harlan Carvey. 2016. *Windows Registry Forensics: Advanced Digital Forensic Analysis of the Windows Registry.* Syngress.

[6] Sarah Cooley. [n.d.]. Introduction to Hyper-V on Windows 10. https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/

[7] Thomas Deepu. [n.d.]. Windows Subsystem for Linux Overview. https://docs.microsoft.com/en-us/archive/blogs/wsl/windows-subsystem-for-linux-overview

[8] Mariano Graziano, Andrea Lanzi, and Davide Balzarotti. 2013. Hypervisor memory forensics. In *International Workshop on Recent Advances in Intrusion Detection.* Springer, 21–40.

[9] Jack Hammons. 2019. Shipping a Linux Kernel with Windows. https://devblogs.microsoft.com/commandline/shipping-a-linux-kernel-with-windows/

[10] Valeriu Manuel Ionescu, Manan Patel, and Drashti Hindocha. 2019. Alternatives for Running Linux Applications in Windows. In *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI).* 1–4. https://doi.org/10.1109/ECAI46879.2019.9042127

[11] Patrick Kochberger, Alexander Tauber, and Sebastian Schrittwieser. 2019. Assessment of the Transparency of the Windows Subsystem for Linux (WSL). In *2019 International Conference on Software Security and Assurance (ICSSA).* 60–69. https://doi.org/10.1109/ICSSA48308.2019.00015

[12] Black Lotus Labs. 2022. Windows Subsystem for Linux (WSL): Threats Still Lurk Below the (Sub)Surface. https://blog.lumen.com/windows-subsystem-for-linux-wsl-threats/

[13] Blanche Lagny. 2019. Analysis of the amcache. *ANSSI-DFIRSummit* (2019).

[14] Nathan Lewis, Andrew Case, Aisha Ali-Gombe, and Golden G. Richard. 2018. Memory forensics and the Windows Subsystem for Linux. *Digital Investigation* 26 (July 2018), S3–S11. https://doi.org/10.1016/j.diin.2018.04.018

[15] Craig Loewen. [n.d.]. Advanced settings configuration in WSL. https://docs.microsoft.com/en-us/windows/wsl/wsl-config

[16] Craig Loewen. [n.d.]. Häufig gestellte Fragen zu Windows-Subsystem für Linux. https://docs.microsoft.com/de-de/windows/wsl/faq

[17] Craig Loewen. [n.d.]. What is Windows Subsystem for Linux. https://docs.microsoft.com/en-us/windows/wsl/about

[18] Craig Loewen. 2019. What's new for WSL in Windows 10 version 1903? https://devblogs.microsoft.com/commandline/whats-new-for-wsl-in-windows-10-version-1903/

[19] Asif Matadar. 2020. Investigating WSL Endpoints. https://www.osdfcon.org/presentations/2020/Asif-Matadar_Investigating-WSL-Endpoints.pdf

[20] Joachim Metz. [n.d.]. Prefetch - Forensics Wiki. https://forensicswiki.xyz/wiki/index.php?title=Prefetch

[21] Rainer Poisel, Erich Malzer, and Simon Tjoa. 2013. Evidence and Cloud Computing: The Virtual Machine Introspection Approach. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 4, 1 (2013), 135–152.

[22] Hammad Riaz and Mohammad Ashraf Tahir. 2018. Analysis of VMware virtual machine in forensics and anti-forensics paradigm. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS).* 1–6. https://doi.org/10.1109/ISDFS.2018.8355375

[23] St191404. [n.d.]. Jump Lists – IT-Forensik Wiki. https://it-forensik.fiw.hs-wismar.de/index.php/Jump_Lists

[24] StatCounter. [n.d.]. Desktop Operating System Market Share Worldwide. https://gs.statcounter.com/os-market-share/desktop/worldwide

[25] Philip Steele. [n.d.]. Hyper-V Storage I/O Performance. https://docs.microsoft.com/en-us/windows-server/administration/performance-tuning/role/hyper-v-server/storage-io-performance

[26] V.K. Vaishnavi and W. Kuechler. 2015. *Design Science Research Methods and Patterns: Innovating Information and Communication Technology, 2nd Edition.* CRC Press. https://books.google.at/books?id=OOE_CQAAQBAJ

[27] Steven White. [n.d.]. Registry Hives - Win32 apps. https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry-hives

[28] Steven White. 2020. COM Class Objects and CLSIDs. https://docs.microsoft.com/en-us/windows/win32/com/com-class-objects-and-clsids

[29] Paul Wise. [n.d.]. HowToUpgradeKernel - Debian Wiki. https://wiki.debian.org/HowToUpgradeKernel

[30] Jidong Xiao, Lei Lu, Haining Wang, and Xiaoyun Zhu. 2016. HyperLink: Virtual machine introspection and memory forensic analysis without kernel source code. In *2016 IEEE international conference on autonomic computing (ICAC).* IEEE, 127–136.

A ONLINE RESOURCES

The RegRipper plugin discussed in this paper is available for download here:
https://github.com/Phil-31/additional_files/blob/main/wsldetect.pl.

Additional event log example outputs referenced in Section 4 can be downloaded from here:
https://github.com/Phil-31/additional_files/tree/main/Eventlog