Disk Forensics II

1.1 Windows Artifacts Analysis

The Windows Registry is a crucial hierarchical, centralized, and scalable database that stores the configuration settings and system information for the Windows Operating System and its installed applications. Its design enables fast access to configuration data and is fundamental for system restore and backup operations. Forensically, the Registry is invaluable as it often contains traces related to malicious or suspicious activities on the system. It helps analysts identify malware persistence mechanisms and provides crucial user and program execution information. The Registry stores a wide range of data, including:

- Configuration settings and system information.
- User profiles and application-specific settings.
- Start-up programs and services.
- Network configurations and security policies.

Structure. The representation of the Registry is simplified, where keys are analogous to folders, which can be further organized into sub-keys. The structure and storage mechanisms have evolved significantly across Windows versions:

- *Windows 3.0*: Configuration data was decentralized, with the `WIN.INI` file using sections like "Extension" to store File Manager and Program Manager data.
- *Windows 3.1*: This introduced `REG.DAT` , the first single, non-text file that contained the entire system registry.
- *Windows 95*: The system adopted a split structure with the creation of `USER.DAT` and `SYSTEM.DAT` to store user-specific and machine-specific data separately.
- *Windows NT/2000/XP and Later*: The registry was reorganized into multiple files (Hives), and security was enhanced, allowing every node (key) in the Registry to have its own access control list.

The logical organization of the Registry is divided into main root keys, known as Hives (represented with the prefix `HKEY_`):

- `HKEY_CLASSES_ROOT` : Stores data about registered applications, primarily focusing on file associations (which program opens which file type) and OLE (Object Linking and Embedding) Object Classes.
- `HKEY_CURRENT_USER` : Contains the settings and configuration data specific to the currently logged-on user.
- `HKEY_LOCAL_MACHINE` : Holds configuration data and hardware profiles used by the Local computer at startup, including device driver settings and system services.

- `HKEY_USER` : Stores configuration data for all users on the local computer, with sub-keys corresponding to each user-s Security Identifier (SID).
- `HKEY_CURRENT_CONFIG` : Stores configuration data that is specific to the hardware profile currently being used by the local computer.

These logical hives are stored physically on the disk as files, typically found in the system's configuration directories. `NTUSER.DAT` physically stores the bulk of the data for the `HKEY_CURRENT_USER` hive, encompassing user-specific settings like desktop layout, file associations, and installed software settings. `SYSTEM` contains critical information about the computer's hardware, device drivers, and system services. Lastly, `SOFTWARE` holds software-related settings for both the system and applications.

For opening and analyzing external registry files, tools like FTK Registry Viewer are highly valued in the forensic community because of their robust capability to process and open even corrupted or non-standard registry files.

Authentication and Artifacts. Windows employs several mechanisms for user authentication and authorization, the evolution of which has led to significant changes in forensic artifact collection. Early Windows versions, up to Windows XP, utilized the LM Hash (LAN Manager Hash). This mechanism is considered very weak and can be easily cracked due to poor cryptographic design. The NTLM Hash (NT LAN Manager Hash) was introduced with Windows NT and ran alongisde LM Hash in subsequent versions. NTLM is a substantial improvement as it uses the MD4 hashing mechanism and does not rely on truncation. Since Windows Vista/7, credential security has continually improved, notably with the adoption of Kerberos as the default network authentication protocol. Further advancements in Windows 10 and later introduced Virtualization-Based Security (VBS), which isolates and protects sensitive system processes, including credential storage, making modern systems significantly harder to compromise.

Windows Active Directory (AD) is a centralized, hierarchical service designed to manage network resources and user access. The structure is based on logical containers:

(1) *Organizational Units* (OUs): Logical containers that organize objects (such as users, groups, or computers) within a domain based on specific administrative or organizational criteria.

(2) *Domains*: Collections of Organizational Units that share a common security boundary and administration.

(3) *Trees*: A group of domains that share a contiguous name space.

(4) *Forests*: The largest structure, consisting of a group of trees that share a common global catalog and security trusts.

Group Policies are the foundational rules applied to manage and enforce configuration settings across these Organizational Units and other objects. Authentication and authorization within the AD environment are primarily managed through the Kerberos protocol.

The *Local Security Authority Subsystem Service* (LSASS) is the core Windows process responsible for managing authentication, defining local security policies, and checking user rights. It is implemented via the executable `lsass.exe` . When a user successfully authenticates, LSASS stores their credential material, including clear-text passwords, NTLM hashes, and Kerberos tickers, in memory. Forensic analysis of the LSASS process memory dump is therefore a

highly effective method for harvesting active user credentials. LSASS also manages critical system secrets stored in the Registry. These special secrets, which include credentials related to Active Directory and the Data Protection API (DPAPI), are stored within the protected key: `HKEY_LOCAL_MACHINE/SECURITY/Policy/Secrets` . Due to the high sensitivity of this data, accessing these secrets often requires specialized tools like the `lsadump` module of Mimikatz, which bypasses standard security protections to extract the hidden registry key data.

Windows manages its system and application history through *Event Logs*, which are accessible locally via the Event Viewer (`eventvwr.msc`). These logs are stored in files with the `.evtx` extension and are categorized by function:

- *Security Logs*: Record events related to security, such as login attempts (success and failure), permission changes, and object access.
- *Application Logs*: Monitor possible crashes or failures of applications and other related software events.
- *System Logs*: Track core system activities, including device driver loading, hardware failures, and other Operating System related events.
- *Setup Logs*: Relate specifically to the Windows installation and update process.

For visualizing and analyzing external `.evtx` files during forensics, tools such as FullEventLogView are commonly employed.

Prefetch Files are an artifact created by Windows to improve system performance. Their primary purpose is to reduce the boot time of applications by caching information about the data and code resources that were accessed during the application'f first few launches. These files have the `.pf` extension and are physically located in the `C:/Windows/Prefetch` directory. Forensically, prefetch files are critical because they provide undeniable evidence of a program's execution and the number of times it was run. They can be analyzed using specialized tools like WinPrefetchView.

Scripting Languages and *macro* functionalities are powerful tools for automation but are frequently abused by attackers to deliver and conceal malicious code.

- *PowerShell*: This is a powerful, object-oriented scripting language native to Windows, primarily used for automating system operations and task management. Its flexibility and deep integration with the OS make it a common choice for adversaries to hide malicious code due to its ability to run complex commands from memory. For forensics and malware analysis, de-obfuscating complex PowerShell scripts is a necessity, often accomplished with tools like PowerDecode.
- *Visual Basic (VB) Macros*: These are embedded code snippets often found in document files (e.g., Microsoft Office). Macros offer extensive functionality, including the ability to access Windows APIs and interact directly with memory management, which enables powerful and stealthy malware execution. As with PowerShell, VB macros found in malicious documents are critical artifacts that must be extracted and analyzed.

1.2 Linux

Partitions. The organizational approach for storage in Linux, while sharing common standards like MBR and GPT with Windows, employs a distinct hierarchical structure and naming

convention. Linux supports both the older Master Boot Record (MBR) and the modern GUID Partition Table (GPT) partitioning schemes. However, the logical role and mounting points of the partitions are different from Windows:

- *EFI/BIOS Partitions*: These are required for booting the operating system. They contain the necessary bootloader code, often including an `.efi` binary, and are typcially mounted to the `/boot` directory.
- *Root Partition*: This partition contains the core operating system files and directories, forming the root of the entire file system hierarchy. It is always mounted as `/` .
- *Swap Partition*: This space is dedicated for use as virtual memory when the system's physical RAM is completely utilized, preventing system crashes under high memory load.

A common practice in Linux systems is Partition Splitting. This means the main file system can be logically divided, where specific subdirectories of the root partition (such as `/home` , `/var` , `/log` , or `/usr`) are mounted as separate, dedicated partitions. This segregation can improve performance, stability, and forensic analysis by isolating different types of data.

Block Devices. In Linux, all storage drives and their partitions are managed as block devices, which are special files located within the `/dev` directory. Data transfer for these devices occurs in fixed-size blocks rather than a single character at a time. Devices are identified by a consistent naming convention:

- `/dev/sdX` : Used to identify older storage devices like HDDs, modern SSDs, and USB drives. The letter 'X' increments for each physical device found (e.g., `/dev/sda` , `/dev/sdb`).
- `/dev/srX` : Used for optical drives such as CD or DVD drives.
- `/dev/nvmeX` : Used specifically for high-speed NVMe SSDs.

It is important to note that not all entries in the `/dev` folder represent physical hardware. Some, such as `/dev/loop0` (for loop devices), can simulate hardware to allow files to be accessed as if they were disk drives. Partitions are derived from the main device name by appending a numerical identifier (e.g., the first partition on the first SATA drive is named `/dev/sda1`). Devices and their partitions can be visualized and managed using tools like `fdisk` and `lsblk` .

File systems. Linux environments utilize several robust file systems, each with characteristics suited to different operational needs, all of which are critical for forensic analysis. Common Linux File Systems:

- *Ext4*: This is the current standard and default for many modern Linux distributions. It is widely adopted due to its superior stability, performance, and solid backward compatibility with its predecessors.
- *Ext3*: While still common on older installations, most systems have migrated to Ext4. It introduced journaling to the Ext family.
- *BTRFS*: This modern file system is popular in environments requiring flexible storage management, as it natively features advanced characteristics such as snapshots and self-healing capabilities.

- *XFS*: Often the default for certain enterprise-grade Linux distributions, XFS is renowned for its scalability and performance, making it popular for systems that handle exceptionally large files or require high levels of Input/Output throughput.

The design of the ext family of file systems is based on organizing the disk space into Block Groups, each composed of sectors (typically multiples of 512 bytes):

- *Superblock*: A central data structure containing critical configuration information about the entire file system, such as its size, and the location of the first Inode table.
- *Inodes*: These are data structures that store the metadata about every file and directory, including permissions, ownership, size, and timestamps. Crucially, Inodes contain pointers to the physical data blocks where the file content resides, often utilizing multiple levels of indirection to locate the data.
- *Data Blocks*: These are the portions of the file system that contain the actual file data.
- *Bitmaps*: Both block and Inode bitmaps are used to track the allocation status (used or free) of all data blocks and Inodes within a block group.

All ext file systems support journaling, which is a record of file system changes used to maintain integrity. Tools available for analyzing this structure and journal data include `tune2fs` , `debug2fs` , and `dumpe2fs` .

The Ext4 file system extends the functionalities of its predecessors by introducing key features:

- *Flex Block Groups*: A more advanced grouping of multiple block groups together, which is particularly useful for efficiency managing very large volumes and optimizing performance.
- *Extents*: These replace the traditional indirect block pointers. An Extent is a contiguous ensemble of blocks, allowing for a single allocation entry to point to a large, continuous chunk of data. This improves performance and is necessary for supporting larger individual files.
- *Advanced Journaling*: Ext4's journaling includes checksums to verify the integrity of the journal itself.

For forensic purposes, the utility `sudo fsstat -t DEVICE` can be used to quickly determine the partition type, and `sudo fsstat -f partition_type PARTITION` can be used to list detailed information about the structure of a specific partition. To recover deleted files from ext3 or ext4 partitions, specialized tools are necessary:

- `ext4undelete` : A tool capable of restoring specific deleted files, directories, or entire partitions.
- `ext4magic` : Another powerful utility used to recover deleted files from ext3 or ext4 file systems, often supporting recovery based on time or Inode number.