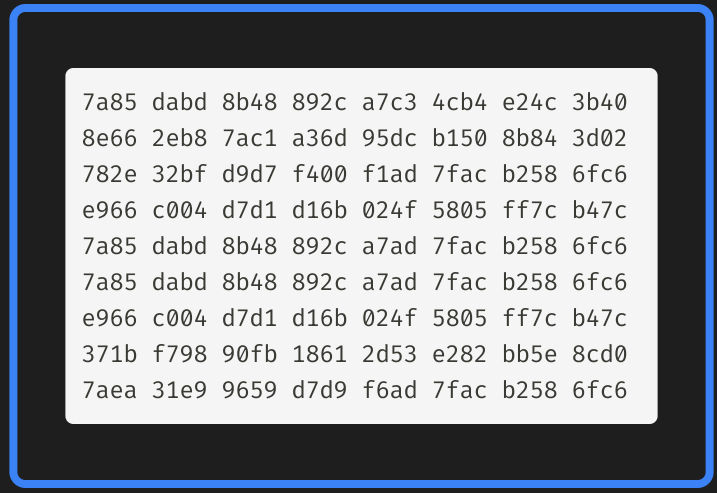
Malware I

Understanding Windows architecture, PE files, and malware tradecraft

Start Week 6 →



Malware Objectives

It’s the job of the reverse engineer to discover the objectives of a malware sample. One common objective is for financial gain.

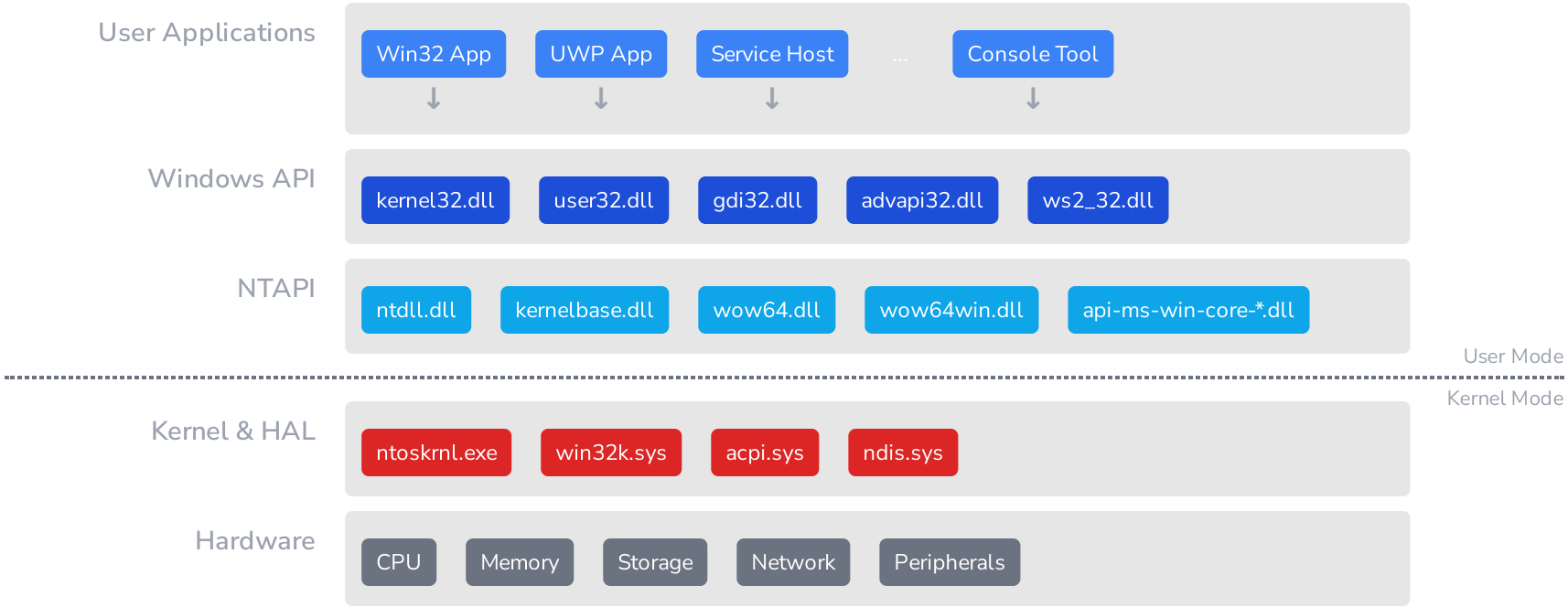
- Ransomware that encrypts data and demands a ransom payment
- Information stealers that harvest credentials, cookies, or crypto walets
- Banking trojeans that inject in to browsers to hijack transactions
- Espionage by states of political, military, and other information
- Cyberattacks by states on critical infrastructure and military capabilities

Malware Platforms

- Malware can target many platforms
- Some sources report >95% of malware targets windows
- Next few weeks we'll be focused on windows

Operating System Architecture

Most concepts from Linux carry over, but Windows layers them differently.



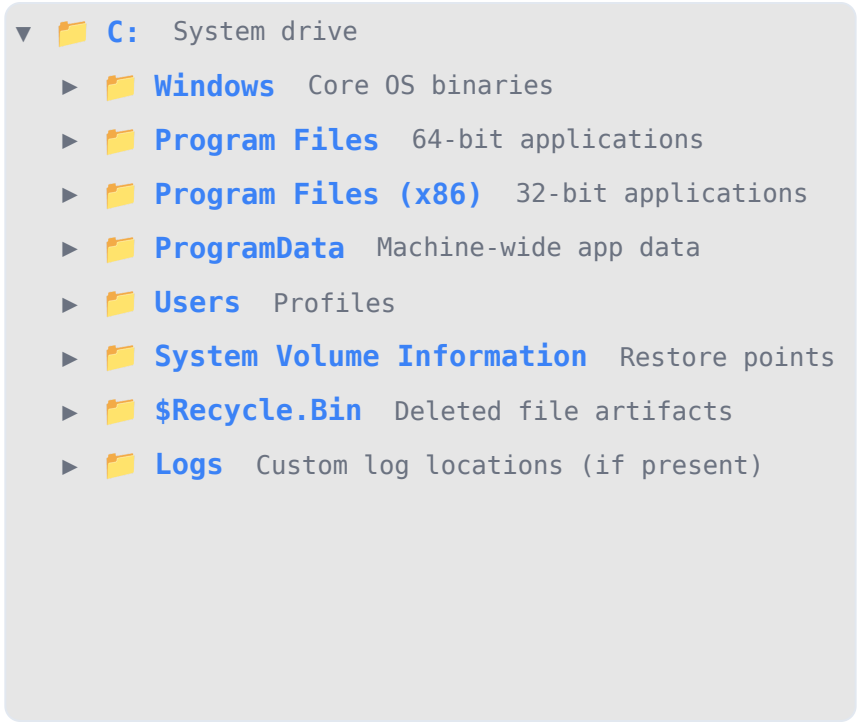
Quick comparison between Windows and Linux

- Dynamically loaded libraries are `.dlls`
- Drivers are usually `.sys`
- Main executable file format is the Portable Executable (`PE`)
- Main scripting language is *powershell*

Windows Directory Structure

Overview of the file system.

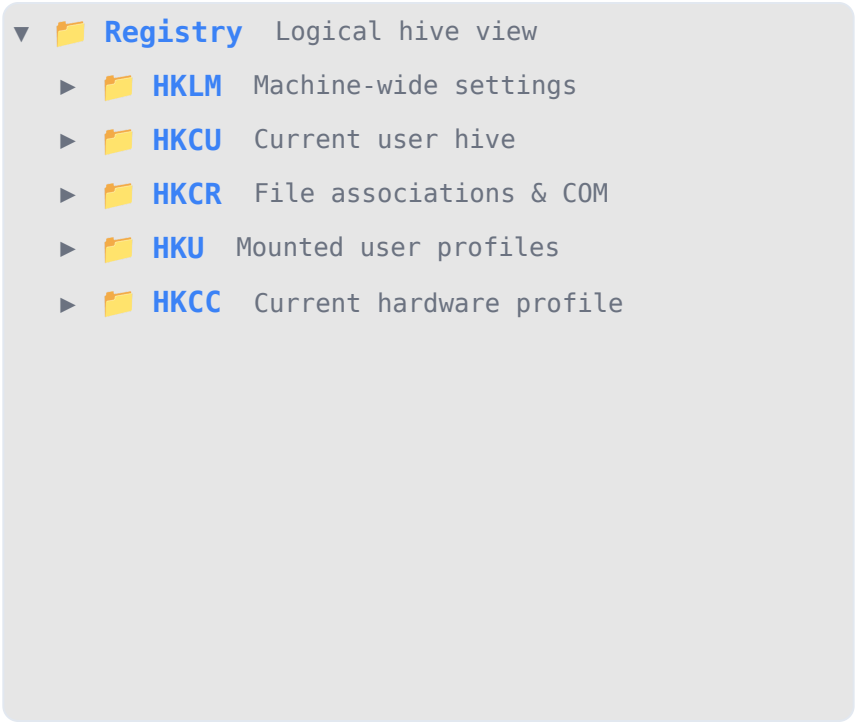
- System binaries in `%SystemRoot%`
- User data and Startup folders per profile
- Artifacts under `ProgramData` and `AppData`



Windows Registry Overview

The registry is a hierarchical database.

- Keys and subkeys act like folders; values store data such as `REG_SZ`, `REG_DWORD`, `REG_BINARY`
- Backed by hive files in `%SystemRoot%\System32\Config` plus per-user `NTUSER.DAT` / `USRCLASS.DAT`



Registry Entries to Watch

Registry Path	Analyst Focus
`HKLM\SYSTEM\CurrentControlSet\Services`	Service/driver installs, malicious autostarts
`HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon`	Shell, Userinit, GINA persistence
`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`	Machine-wide startup programs
`HKCU\Software\Microsoft\Windows\CurrentVersion\Run`	User startup (per-profile persistence)
`HKCU\Software\Classes\CLSID`	COM hijacking, shell extension abuse
`HKCU\Software\Microsoft\Office\*`	Macro trust, add-in persistence

Portable Executable Overview

PE files start with a DOS header that points to the NT headers containing loader metadata.

```
typedef struct _IMAGE_DOS_HEADER {
  WORD   e_magic;    // "MZ"
  WORD   e_cblp;
  WORD   e_cp;
  WORD   e_crlc;
  WORD   e_cparhdr;
  WORD   e_minalloc;
  WORD   e_maxalloc;
  WORD   e_ss;
  WORD   e_sp;
  WORD   e_csum;
  WORD   e_ip;
```

```
typedef struct _IMAGE_NT_HEADERS64 {
  DWORD Signature;          // "PE\0\0"
  IMAGE_FILE_HEADER FileHeader;
  IMAGE_OPTIONAL_HEADER64 OptionalHeader;
} IMAGE_NT_HEADERS64;

typedef struct _IMAGE_FILE_HEADER {
  WORD  Machine;
  WORD  NumberOfSections;
  DWORD TimeDateStamp;
  DWORD PointerToSymbolTable;
  DWORD NumberOfSymbols;
```

PE Sections & Directories

Common Sections

- `.text` – executable code
- `.rdata` – imports, constants
- `.data` – initialized globals
- `.pdata` – exception handling (x64)
- `.reloc` – base relocations
- `.rsrc` – resources (icons, dialogs)

Data Directories

- Import Table – DLL dependencies, function thunks
- Export Table – provided APIs
- Resource Table – embedded payloads
- Relocation Table – ASLR support
- TLSTable – thread local storage callbacks
- Load Config – Control Flow Guard, SEH

Windows API Reference

Provides useful information for the reverse engineer when these functions are imported.

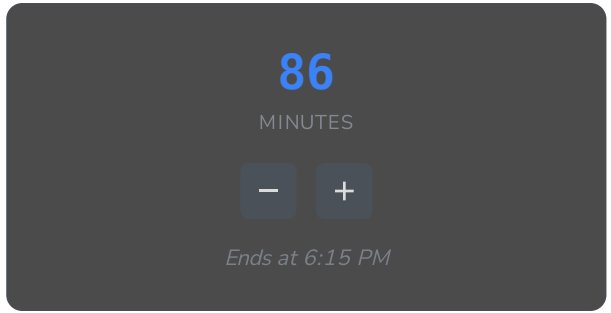
Windows API Example

```
#include <windows.h>
#include <stdio.h>

int wmain() {
  const wchar_t *path = L"C:\\\\Temp\\\\example.txt";

  HANDLE file = CreateFileW(
      path,
      GENERIC_READ | GENERIC_WRITE,
      FILE_SHARE_READ,
      nullptr,
      OPEN_ALWAYS,
      FILE_ATTRIBUTE_NORMAL,
      nullptr);

  if (file == INVALID_HANDLE_VALUE) {
    DWORD err = GetLastError();
    wprintf(L"CreateFile failed: %lu\n", err);
    return 1;
  }

  const char payload[] = "Hello Windows API!\\r\\n";
  DWORD bytesWritten = 0;
```

Lab 1

Portable Executable triage and header parsing.

https://hacs408e.umd.edu/labs/week-06/lab-1/



Malware Analyst Focus Areas

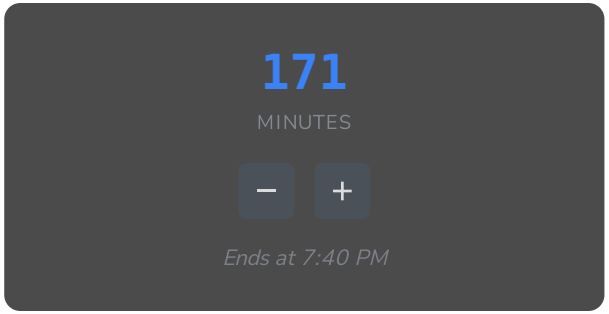
The rapid pace of cyberattacks makes quick malware triage useful. We can discover a lot about a sample just by running it and monitoring for file system changes, network activity, and registry changes.

- Payloads
- System modifications
- Persistence
- Command and control
- Evasion
- Origins
- Signatures

Lab 2

Windows malware behavior analysis.

https://hacs408e.umd.edu/labs/week-06/lab-2/



Homework

- Homework due next week
- Quiz on networking basics next week