RustOps

Malware Development Using the Rust Programming Language

whoami

Jose Plascencia

- 34 years old
- Security Consultant @ DirectDefense
- Mentor
- Marathon runner
- Red and Blue Teaming, Windows Internals, Rust dev, Memory Safety, Reverse Engineering

Notes:
Show of hands

- First timers?
- Students?
- Red Teamers?
- Blue Teamers?

Disclaimer

> ⚠ **Warning**
>
> 1. Opinions expressed are solely my own and do **not** express the views or opinions of my employer.
> 2. This presentation has **not** been reviewed or approved by the Rust Foundation.
> 3. I am **not** in a position to endorse any software vendor.
> 4. The following content **is** for personal development.

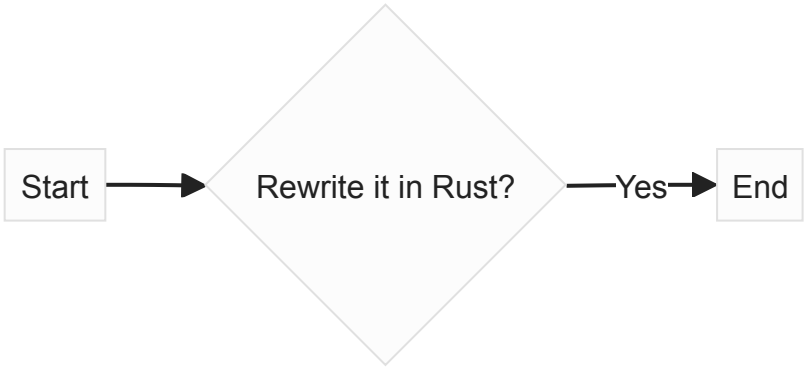
What?

Goal: To arm you with enough knowledge to use Rust in your Red Team engagements and exercises.

RustOps Starterpack:
Obsidian notebook, slides and other workshop content will be made available after the presentation.

Why Rust?

Memory Safety, Security & Reliability, No GC, Concurrent and Parallel Programming, Cross-Platform Support, Crustacean Community, Modern Language Features, ...



Note:

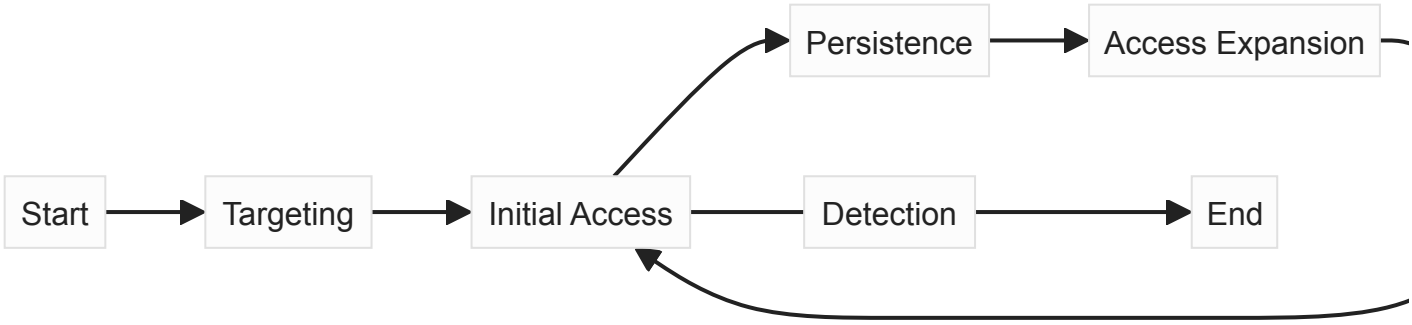
- In the context of malware dev. there are detection challenges...more on that later
- CVE-2024-24576 - CVSS 10
  - "The Rust Security Response WG was notified that the **Rust standard library did not properly escape arguments when invoking batch files (with the bat and cmd extensions)** on Windows using the Command API,"

Goals of this talk

- [ ] Malware Fundamentals
- [ ] Rust Fundamentals
- [ ] Guidance
  - [ ] Network Communication
  - [ ] Bad OpSec
  - [ ] OpSec
- [ ] Call to action

Malware Fundamentals

- Malware is used to support the Computer Network Exploitation (CNE) life cycle.
  - Ideal operational CNE life cycle from *Network Attacks & Exploitation: A Framework* by Matthew Monte

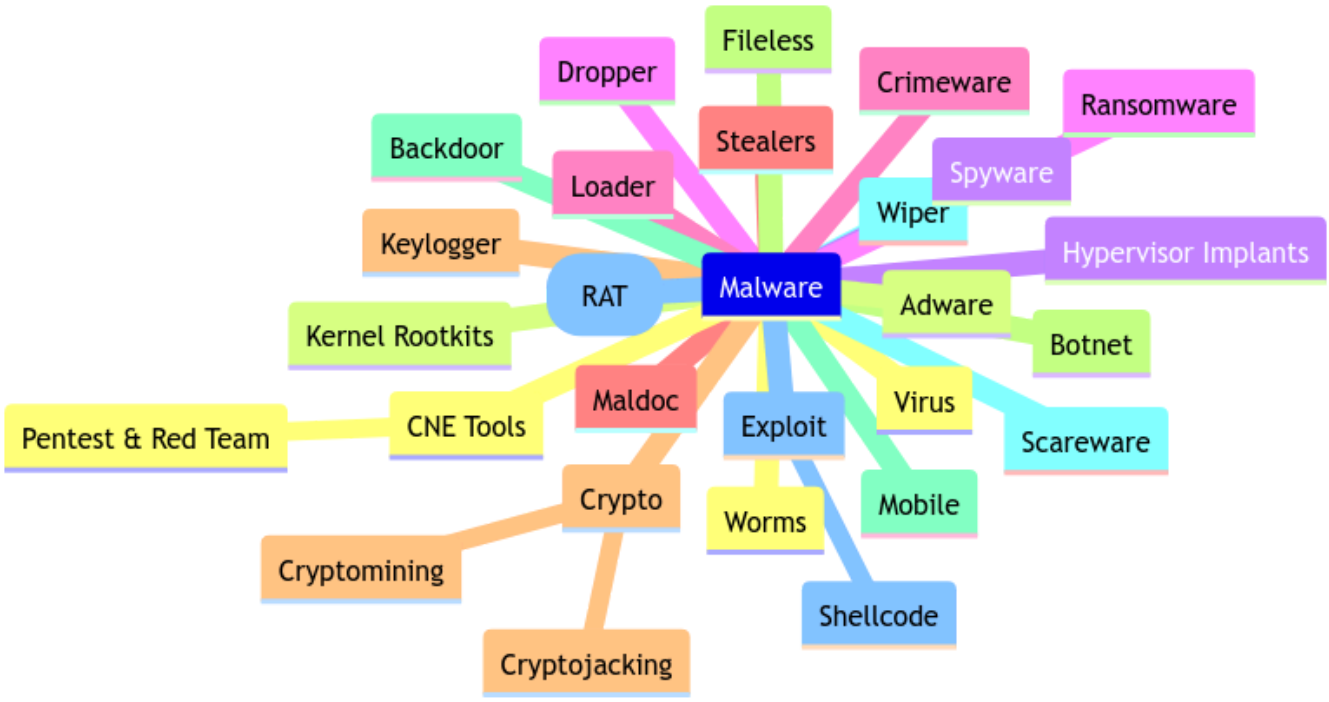


- MITRE ATT&CK® Matrix for Enterprise The "Why" and "How" of CNE through documented real-world Tactics, techniques, and procedures (TTPs).

Note:

- Start, Targeting, IA, Persistence, Access Expansion, Exfil, Detection, End
- Decoupling
- Common C2 frameworks
- Motives: Destruction, Sabotage, Financial gain
- As Red Teamers we want to test the Blue Team.

Malware Types



Rust Malware History

Threat Actors:

- RustDown
- BlueNoroff, RustBucket
- COLDRIVER
- WildCard
- BlackCat
- RansomExx

[Awesome Rust Malware Gallery]

Note:
Notable cases where Rust was used for cyberattacks

- Agenda
- BlackCat
- Buer loader
- Hive
- SSLoad
- We'll cover bad opsec later

Rust Fundamentals

Rust Development Environment

- VMware Workstation Pro: Now Available Free for Personal Use
- Windows Dev Instance
- VS Code
- Set up your dev environment on Windows for Rust
  - Visual C++ Build Tools
  - Rustup: The Rust Installer

```
rustc --version ; cargo --version
```

Ready? LFG!

Notes:

- rust-analyzer for VS Code
- CodeLLDB

Rust Terminology

- Crate
- Executable Crate
- Library Crate
- External Crate
- Packages

- Modules
- Cargo.toml
- Cargo.lock
- '.rs' Extension
  Notes:

- [ ] Cargo.toml is the manifest and config file
- [ ] Cargo.lock is a record of all dependencies
- [ ] Rust source file

Cargo & Crates

- Cargo: The Rust package manager
- RTFM (Yes, there is a book on Cargo.)
- Crates.io: The Rust community's crate registry

New project:
`cargo new {{project_name}}`

Create a new directory under the project:

```
mdkir examples ; touch examples/demo.rs
```

Run example:
`cargo run --example demo`

Build project:
`cargo build`

Run project within the directory :
`cargo run`

Generate HTML documentation from comments:

```
cargo doc
```

Alt. generate an executable from a single .rs file using the Rust compiler:

```
rustc examples/demo.rs
```

Notes:

- [ ] Note there are several flags under the run command

Rust Project Structure

```
❯ ls *
Cargo.lock      Cargo.toml
examples:
demo.rs
src:
main.rs
target:
CACHEDIR.TAG    debug
❯ cargo run
    ... `removed for brevity`
Hello, world!
❯ cargo run --example demo
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.06s
     Running `target/debug/examples/demo`
Hello, from the example dir!
```

Notes:

- [ ] Explain target debug
- [ ] Cargo.toml is the manifest and config file
- [ ] Cargo.lock is a record of all dependencies
- [ ] Rust source file

Rust Basics

```rust
// <--crate imports
//entry point function
//`fn` denotes a function and the curly braces demarcates a function block.
// V important concept for lifetimes
fn main() {
//`println!` is a macro.
println!("Hello, from the example dir!");

let message = "Hello";
let name = "🦀";
println!("{:?}, {:?}",message,name);

}
```

Notes:

- [ ] Macros provides the compatibility of a variadic function.

- [ ] The `println!` macro prints to standard output and appends a linefeed to the formatted output.
- [ ] Check out the `asm!`
- [ ] The `:?` placeholder renders the dev view of the value. Great for debugging.
  - [ ] Requires the debug trait `#[derive(Debug)]` over a struct, `println!` will pretty print the contents of the struct making it easier for the dev.
- [ ] There are format specifier as well. e.g., {:p} for a pointer address

---

Important Concepts 🥲

Closures, Collections, Control Flow, Enums, Functions, Generics, Macros, **Memory**, Modules, Patterns, Structures, Threads, Traits, Variables, **Lifetimes**, **Ownership**, **References**

The Rust Book

Offline copy:
`rustup doc --book`

The Rust Programming Language, 2nd Edition

Notes:

- [ ] The ones in bold are imperative to understand.

Native Windows APIs

Cargo.toml

```
[dependencies]
windows = { version = "0.58.0, features = ["Win32_Foundation","Win32_UI_Shell"
"Win32_UI_WindowsAndMessaging"] }
```

main.rs

```
use windows::{core::*, Win32::UI::Shell::*, Win32::UI::WindowsAndMessaging::*};

fn main() {
        //No memory safety guarantees!
    unsafe {
        MessageBoxA(None, s!("Ansi"), s!("World"), MB_OK);
    }
}
```

Rust for Windows, and the _windows_crate

MSDN

Notes:

- [ ] Note some Windows crates are safe
- [ ] Using unsafe function blocks unlocks super powers or control over raw pointers, access to other unsafe functions, traits, etc.
- [ ] MSDN can be used to learn more about functions and how arguments should be implemented using Rust.

Network Communication Channels

- HTTP/HTTPS
  - Cloud: AWS, GCP, Azure, ...
  - Web services: Social Media Platforms
- DNS
- SMTP/IMAP
- FTP/SFTP
- P2P
- TOR
- TCP
- UDP
- ICMP

HTTP/HTTPS

- Rust
  - minreq
  - reqwest
  - hyper
  - curl-rust
- Native
  - Windows Internet (WinINet)
  - WinHTTP

Notes:

- Mention Headers, hardcoded user agent, for fingerprinting and classifying malware
- Mention URL and API endpoints, e.g. 'api/update', 'api/v1/'

- Static config will result in trivial detection

GET

```
//import reqwest w/ "json" features
//import tokio with "full" features
//include main

let body = reqwest::get("http://httpbin.org/get")
                .await?
                .text()
                .await?;
println!("body = {body:?}");
```

POST

```
let client = reqwest::Client::new();
let res = client.post("http://httpbin.org/post")
        .body("POST Malone")
        .send()
        .await?;
```

Blue Team Challenges & Red Team Considerations

Buer C2 URLs

- "Project 0xA11C" by Nicole Fishbein, Juan Andrés Guerrero-Saade (Presented at BlackHat 2024)
  - Methodology and tools for Rust malware RE

Notes:

- Complexity: Rust does not have a stable Application Binary Interface (ABI)
  - Memory safe operations or anything that could cause the app to panic.
- "A Rust developer can create a program without depending on the Rust standard library and only import system libraries as needed" BinaryDefense
- Strategy

- Detection points...

Bad OpSec

Debugging Information

Strings revealing: Hostnames, URLs, Rust Crates & Modules, PDB File Path

```
user@user-virtual-machine:/dev/shm$ strings 88689636f4b2287701b63f42c12e7e2387bf4c3ecc45eeb8a61ea707126bad9b.exe | grep '^src.*rs$'
src\defence.rs
src\main.rs
```

Notes: Remove debugging information and symbol tables

- Alt. https://github.com/BinaryDefense/GhidraRustDependenciesExtractor

---

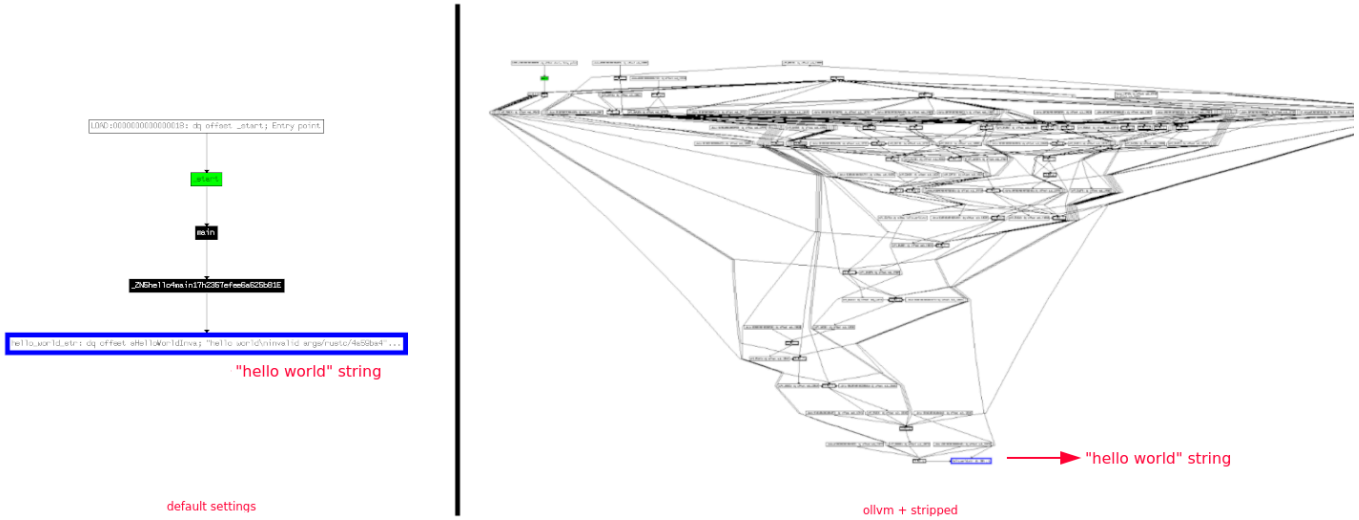
Windows API Calls || Injection Techniques

```
 NtAllocateVirtualMemory(NtCurrentProcess,&mut base_address,0, &mut shellcode.len(), 0x00003000, 0x40);
#[cfg(feature = "console_mode")]
println!("[*] Calling NtWriteVirtualMemory");
NtWriteVirtualMemory(NtCurrentProcess,base_address,shellcode.as_ptr() as _,shellcode.len() as usize,null_mu
#[cfg(feature = "console_mode")]
println!("[*] Calling NtProtectVirtualMemory");
NtProtectVirtualMemory(NtCurrentProcess, &mut base_address, &mut sellcode_length,  0x20, &mut temp);
let mut thread_handle : *mut c_void = std::ptr::null_mut();
NtCreateThreadEx(&mut thread_handle, MAXIMUM_ALLOWED, std::ptr::null_mut(), NtCurrentProcess, base_address
NtWaitForSingleObject(thread_handle, 0, std::ptr::null_mut());
```

```
mov     dword ptr [rsp-8+arg_20], 40h ; '@'
mov     dword ptr [rsp-8+lpNumberOfBytesWritten], 3000h
lea     rdx, [rbp+1B0h+var_C0]
mov     r9, rbp
mov     rcx, 0FFFFFFFFFFFFFFFFh
xor     r8d, r8d
call    cs:NtAllocateVirtualMemory
mov     rdx, qword ptr [rbp+1B0h+var_C0]
mov     r8, qword ptr [rbp+1B0h+var_90+8] ; shellcode
mov     r9, qword ptr [rbp+1B0h+var_80]
mov     [rsp-8+lpNumberOfBytesWritten], 0
mov     rcx, 0FFFFFFFFFFFFFFFFh
call    cs:NtWriteVirtualMemory
lea     rax, [rbp+1B0h+var_B0]
mov     [rsp-8+lpNumberOfBytesWritten], rax
lea     rdx, [rbp+1B0h+var_C0]
lea     r8, [rbp+1B0h+var_A8]
mov     rcx, 0FFFFFFFFFFFFFFFFh
mov     r9d, 20h ; ' '
call    cs:NtProtectVirtualMemory
mov     qword ptr [rbp+1B0h+StartupInfo.cb], 0
mov     rax, qword ptr [rbp+1B0h+var_C0]
xorps   xmm0, xmm0
movups  [rsp-8+arg_40], xmm0
movups  [rsp-8+arg_30], xmm0
mov     [rsp-8+lpNumberOfBytesWritten], rax
mov     [rsp-8+arg_28], 0
mov     [rsp-8+arg_20], 0
mov     rcx, rbp
mov     edx, 2000000h
xor     r8d, r8d
mov     r9, 0FFFFFFFFFFFFFFFFh
call    cs:NtCreateThreadEx
mov     rcx, qword ptr [rbp+1B0h+StartupInfo.cb]
xor     edx, edx
xor     r8d, r8d
call    cs:NtWaitForSingleObject
```

Freeze.rs

High entropy



Note:

- These images are not my own. Please see referenced article online.
- Shannon Entropy - The measure of randomness

OpSec

- Obfuscation:
  - Obfuscator LLVM (OLLVM)
  - Goldberg
- VEH:
  - Control Flow Manipulation (`RtlAddVectoredExceptionHandler`)
- Direct and Indirect/Dynamic syscalls
  - Evading hooks on *ntdll.dll* functions
- Hooking & Unhooking:
  - API interception and manipulation
  - Trampoline

- Process Injection
  - Process Hollowing
  - DLL injection
  - Module stomping
  - Asynchronous Procedure Calls (APC) Injection
- Patching:
  - ETW
  - AMSI
- Encoding

    - Base64
    - Hex
    - URL
    - ROT13
- Encryption:
    - XOR
    - AES
    - RC4
    - LZMA

Exercises, Resources, Shout-outs

Community Projects

- https://github.com/joaoviictorti/RustRedOps/
- https://github.com/BlackSnufkin/Rusty-Playground
- https://github.com/Whitecat18/Rust-for-Malware-Development
- https://github.com/trickster0/OffensiveRust
- https://github.com/hakaioffsec/coffee
- https://github.com/Nariod/RustPacker
- https://github.com/janoglezcampos/rust_syscalls
- https://github.com/b1nhack/rust-shellcode
- https://github.com/pumpbin/pumpbin
- https://github.com/g0h4n/syscalls-rs

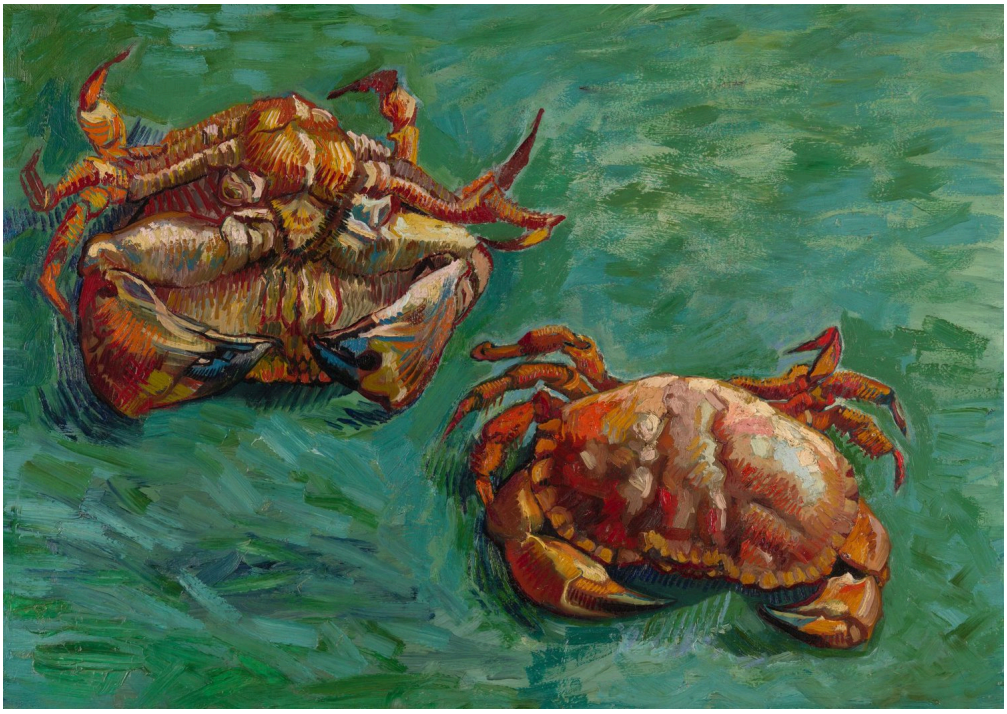
Researchers

- https://github.com/Idov31
- https://github.com/trickster0
- https://github.com/Kudaes
- https://github.com/memN0ps/
- https://github.com/hakaioffsec

Documented Evasion Techniques (Mostly C++)
https://www.unprotect.it

Notes: Use these resources to learn more and implement your capability.

Call to Action



There are a thousand hacking at the branches of evil to one who is striking at the root.

— Henry David Thoreau, Walden, or Life in the Woods

Note:

- I hope you found this useful and fun.
- Why am I sharing this with you?
  - I figured this would be a fun way to learn the language.
  - To gamify it
- Efforts to translate legacy code. DARPA - Project TRACTOR
  - "The TRACTOR **program aims to automate the translation of legacy C code to Rust**. The goal is to achieve the same quality and style that a skilled Rust developer would produce, thereby eliminating the entire class of memory safety security vulnerabilities present in C programs. This program may involve novel combinations of software analysis, such as static analysis and dynamic analysis, and machine learning techniques like large language models."

End

Feel free to contact me via Signal

Github: grim3

Appendix

Detection Tools/ OpSec Validation

- [ ] VirusTotal
- [ ] ThreatCheck
- [ ] PE-sieve
- [ ] mal_unpack
- [ ] Moneta
- [ ] Beaconeye
- [ ] MalMemDetect
- [ ] SignatureScanner
- [ ] YARA
- [ ] Capa-Rules
- [ ] SpeakEasy

- Books
  - [ ] Evading EDR by Matt Hand
  - [ ] Windows Security Internals by James Forshaw
  - [ ] Windows Internals, Sixth Edition, Part 1 By David A. Solomon Mark E. Russinovich and Alex Ionescu
  - [ ] Windows Internals, Part 2, 7th Edition By Andrea Allievi, Alex Ionescu, David A. Solomon, et.al.

Sysmon

```
[Net.ServicePointManager]: :SecurityProtocol = [Net.SecurityProtocolType]:: T1s12
$url = "https://download.sysinternals.com/files/Sysmon.zip"
$dest = "$env:APPDATA\Sysmon.zip"
Invoke-WebRequest -Uri $url -OutFile $dest
Expand-Archive -Path $dest -DestinationPath "$env:APPDATA\"
```

```
sysmon -accepteula -i
```

Procmon

```
[Net.ServicePointManager]: :SecurityProtocol = [Net.SecurityProtocolType]:: T1s12
$url = "https://download.sysinternals.com/files/ProcessMonitor.zip"
```

```
$dest = "$env:APPDATA\ProcessMonitor.zip"
Invoke-WebRequest -Uri $url -OutFile $dest
Expand-Archive -Path $dest -DestinationPath "$env:APPDATA\"
```

Binary Ninja (binja)

```
[Net.ServicePointManager]: :SecurityProtocol = [Net.SecurityProtocolType]:: T1s12
$url = "https://cdn.binary.ninja/installers/binaryninja_free_win64.exe"

$dest = "$env:APPDATA\binaryninja_free_win64.exe"
Invoke-WebRequest -Uri $url -OutFile $dest
```

WinDbg

```
[Net.ServicePointManager]: :SecurityProtocol = [Net.SecurityProtocolType]:: T1s12

$winDbgUrl = "https://windbg.download.prss.microsoft.com/dbazure/prod/1-2306-12001-
0/windbg.msixbundle"
$windbgdirectoryPath = "C:\Users\dev\windbgx"
New-Item -ItemType Directory -Path $windbgdirectoryPath
Invoke-WebRequest -Uri $winDbgUrl -OutFile "$windbgdirectoryPath\windbg.msixbundle"
Add-AppxPackage -Path "$windbgdirectoryPath\windbg.msixbundle"
```

Credits

Obsidian is the private and flexible writing app that adapts to the way you think.

Advanced Slides is an open source plugin for Obsidian that allows you to create reveal.js based presentations in Obsidian. With this tool anyone who is able to create a note in Obsidian can also create a beautiful presentation.

The artists and researchers for their tireless work.
