Shellcode & Process Injection

25-29 JAN 21

Tasking

Getting familiar with native encryption and in-memory execution techniques.

Detourer

Massive updates to Detourer, which we created and detailed in last writeup. This week's changes:

1. Added support for dynamic module inclusion
    - Project now automatically import payloads that are simply present in the designated build directory
        - Allows for hot-swapping and rapid/automated deployment
    - Loads these modules at runtime to be procedurally injected into (and later removed from) the target
2. Fixed bugs
    - Some payload shells would kill host if used
        - Removed until ready to customize
    - Some pointers would get mixed up and result in looping hooks and corrupted data
        - Fixed in the course of QoL changes below
3. QoL & Honorable mentions
    - Moved most .vcxproj items to property sheets
    - Removed junk & restructured
        - Moved Detourer to Includes, adding room for new projects to use it
        - Reduced reliance on macro magic significantly—nearly eliminated
    - Optimized code size, PE size, PE speed, and build time significantly
        - x64 Release (speed optimized) DLL is only 76KB and rebuilds in under 7s, despite bundling MS Detours + the current payload modules

I also tested the new BCryptEncrypt module on a sample target process and was able to extract the key and plaintext trivially and reliably from intercepted calls. This is promising for the future.

Future Plans

1. Explore potential for global hooks
2. Complete and further modularize payloads
    - Configure payloads to interact well with each other
        - Don’t hook a WriteFile call when we’re trying to exfil data…
    - Add generic payload wrappers to exfil to file/IPC, modify args, etc.
    - Add class to generate hook function based on mix/matched choice of:
        - Target function, e.g. VirtualAlloc(…)
        - Payload type (the generic helper wrapper)
        - (optional) Streamlined custom payload function(s) to extend the above
    - One-file plug & play payload modules, only if possible without magic
    - Add an attacker project template that allows for zero-knowledge configuration and deployment
3. Slim down project and PE even more
    - Fix the #includes
        - Not stomping on others but are often redundant. They’re mostly optimized out of the PE, but slow the build and add clutter
    - Improve backwards compatibility
        - Ideally, pure C eventually
        - Conform to standard, don’t rely on MS stuff
        - Minimize needed Windows SDK version and inclusion/usage
    - Compare runtimes and avoid building from multiple stacks
    - Remove unnecessary items from MS Detours, e.g. sample code, other archs