Shellcode & Process Injection

14-18 DEC 20

Tasking

I was tasked with getting familiar with process injection and hooking.

HookVirtualAlloc

Expanding on previous work, I realized that my injection method was flawed. After reading a bunch of x86 documentation, I found and patched a different part of the process that allowed main to call my function directly instead of reaching into the loaded kernel32.dll.

In the end, my "VirtualAlloc" got called, dumped the caller's arguments as a POC, called the real VirtualAlloc, and transparently returned the result to the caller. One could supply malicious arguments instead to retain access to the contents of that memory. Callers will almost certainly not VirtualQuery their newly allocated memory. In case they do, just hook that too.

Extending this to remote processes would give a significant amount of opportunity. One could spawn a threat or all-access handle, monitor activity, beacon data, etc.

TLDR

1. Fixed + improved local process injection
2. Better understanding of process/-access mechanisms
3. Better understanding of shellcode writing

Future Plans

- Push most of the code to a library + object for cleanliness and extensibility
- Enable remote process injection
- Dynamically scan target process for entry points