



**APPLICATION
SECURITY, INC.**

Hacking Windows Internals

Cesar Cerrudo
Argeniss

Hacking Shared Sections

- Shared Section definition
- Using Shared Sections
- Tools
- Problems
- Searching for holes
- Exploitation
- Conclusions
- References

Shared Section

- Basically a Shared Section is a portion of memory shared by a process, mostly used as an IPC (Inter Process Communication) mechanism.
 - Shared Memory.
 - File Mapping.
 - Named or Unnamed.

Using Shared Sections

- Loading binary images by OS.
 - Process creation.
 - DLL loading.
- Mapping kernel mode memory into user address space !?.
 - Used to avoid kernel transitions.
- Sharing data between processes.
 - GDI and GUI data, pointers !?, counters, any data.

Using Shared Sections

- Creating a shared section

```
HANDLE CreateFileMapping(  
    HANDLE hFile,                // handle to file (file mapping)  
                                //or 0xFFFFFFFF (shared memory)  
    LPSECURITY_ATTRIBUTES lpAttributes,    // security  
    DWORD flProtect,             // protection  
    DWORD dwMaximumSizeHigh,     // high-order DWORD of size  
    DWORD dwMaximumSizeLow,     // low-order DWORD of size  
    LPCTSTR lpName               // object name (named)  
                                //or NULL (unnamed)  
); //returns a shared section handle
```

Using Shared Sections

- Opening an existing shared section

```
HANDLE OpenFileMapping(  
    DWORD dwDesiredAccess,    // access mode (FILE_MAP_WRITE  
                               // FILE_MAP_READ, etc.)  
    BOOL bInheritHandle,      // inherit flag  
    LPCTSTR lpName             // shared section name  
); //returns a shared section handle
```

Using Shared Sections

- Mapping a shared section

```
LPVOID MapViewOfFile(  
    HANDLE hFileMappingObject,           // handle to created/opened  
                                           // shared section  
    DWORD dwDesiredAccess,               // access mode(FILE_MAP_WRITE  
                                           // FILE_MAP_READ, etc.)  
    DWORD dwFileOffsetHigh,              // high-order DWORD of offset  
    DWORD dwFileOffsetLow,               // low-order DWORD of offset  
    SIZE_T dwNumberOfBytesToMap // number of bytes to map  
); //returns a pointer to begining of shared section memory
```

Using Shared Sections

- Ntdll.dll Native API

- NtCreateSection() Creates a new section
- NtOpenSection() Opens an existing section
- NtMapViewOfSection() Map a section on memory
- NtUnmapViewOfSection() Unmap a section from memory
- NtQuerySection() Returns section size
- NtExtendSection() Change section size

Using Shared Sections

- Mapping unnamed Shared Sections.

OpenProcess(PROCESS_DUP_HANDLE,...)

DuplicateHandle(...)

MapViewOfFile(...)

- Need permissions on target process

Using Shared Sections

- Demo

Tools

- Process Explorer
 - Shows information about processes (dlls, handles, etc.).
- WinObj
 - Shows Object Manager Namespace information (objects info, permissions, etc.)
- ListSS
 - Lists Shared Sections names (local and TS sessions).
- DumpSS
 - Dumps Shared Section data.
- TestSS
 - Overwrites Shared Section data (to detect bugs)

Problems

- Input validation
- Weak permissions
- Synchronization

Problems

- Input validation
 - Applications don't perform data validation before using the data.
 - Processes trust data on shared sections.

Problems

- Weak permissions
 - Low privileged users can access (read/write/change permissions) shared sections on high privileged processes (services).
 - Terminal Services (maybe Citrix) users can access (read/write/change permissions) shared sections on local logged on user processes, services and other user sessions.

Problems

- Weak permissions
 - Demo

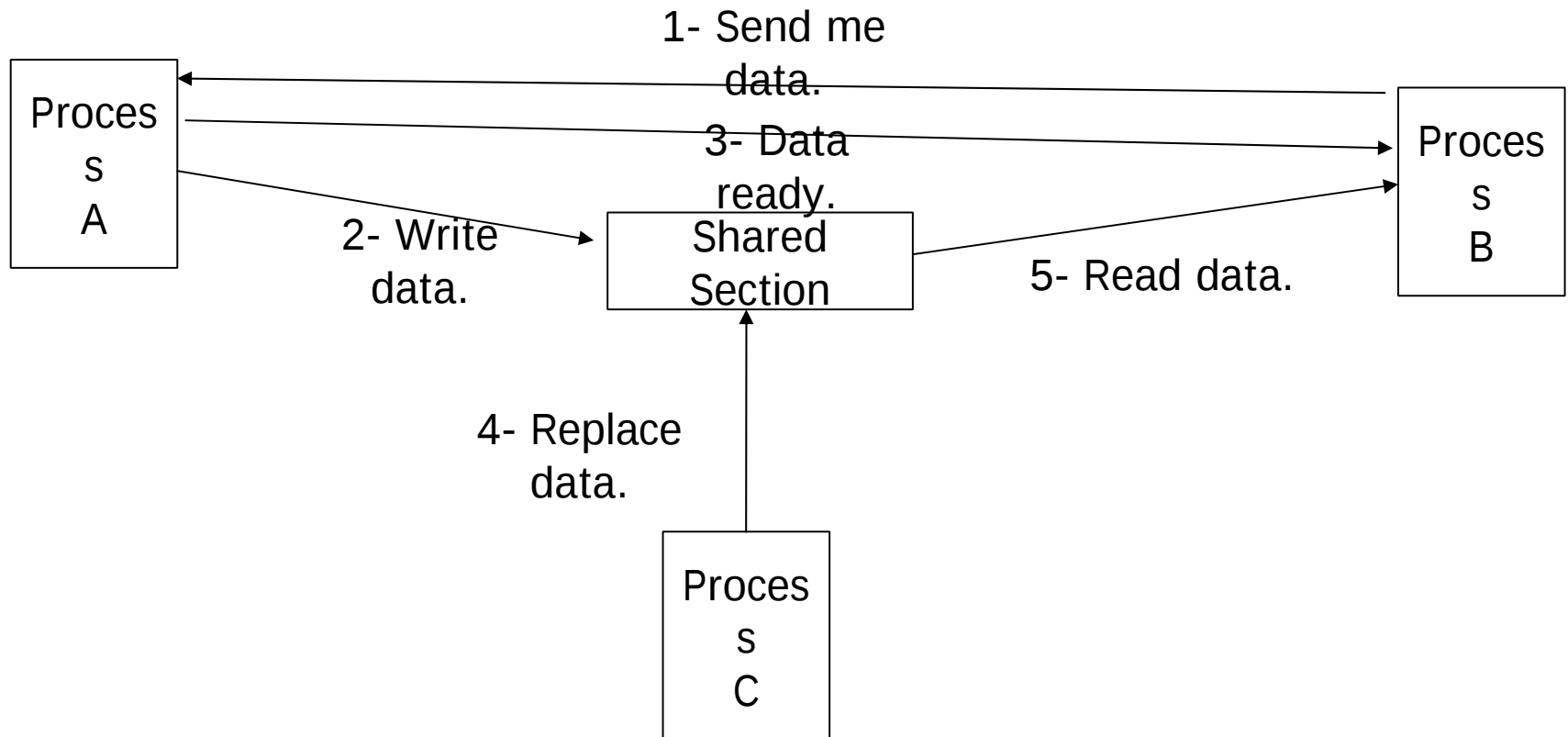
Problems

- Synchronization
 - Not built-in synchronization.
 - Synchronization must be done by processes in order to not corrupt data.
 - There isn't a mechanism to force processes to synchronize or to block shared section access.
 - Any process (with proper rights) can alter a shared section data while another process is using it.

Problems

- Synchronization

- Communication between Process A and B



Searching for holes

- Look for shared sections using Process Explorer or ListSS.
- Attach a process using the shared section to a debugger.
- Run TestSS on shared section.
- Interact with process in order to make it use (read/write) the shared section.
- Look at debugger for crashes :).

Searching for holes

- Demo

Exploitation

- Elevating privileges.
 - Reading data.
 - Altering data.
 - Shared section exploits.
- Using shared sections on virus/rootkits/ etc.

Exploitation

- Reading data.
 - From high privileged processes (services).
 - From local logged on user processes, services and other sessions on Terminal Services.
 - This leads to unauthorized access to data.

Exploitation

- Reading data.
 - Reading Internet Explorer cookies and history information.
(Demo)

Exploitation

- Altering data.
 - On high privileged processes (services).
 - On local logged on user processes, services and other sessions on Terminal Services.
 - This leads to arbitrary code execution, unauthorized access, processes or kernel crashing (DOS).

Exploitation

- Altering data.
 - IIS 5 DOS. (Demo)

Exploitation

- Shared section exploits.
 - When overwriting shared section data allow us to take control of code execution.
 - Some shared sections start addresses are pretty static on same OS and Service Pack.
 - Put shellcode on shared section.
 - Build exploit to jump to shellcode on shared section at static location.

Exploitation

- Shared section exploits.
 - MS05-012 - COM Structured Storage Vulnerability
 - Exploit (Demo)

Exploitation

- Using shared sections on virus/rootkits/etc.
 - Some shared sections are used by many processes (InternatSHData used for Language Settings) others are used by all processes :).
 - Write code to shared section and the code will be instantly mapped on processes memory and also on new created processes.
 - Use SetThreadContext() or CreateRemoteThread() to start executing code.
 - Similar to WriteProcessMemory() - SetThreadContext() technique or DLL Injection.

Exploitation

- Using shared sections on virus/rootkits/etc.
 - Some shared sections have execute access.
 - It would be possible to avoid WinXP sp2 NX .

Conclusions

- Windows and 3rd. Party applications have a bunch of Shared Section related holes.
- These kind of holes will lead to new kind of attacks “SSAtacks” (Shared Section Attacks) ;)
- Microsoft forgot to include a Shared Sections audit on the trustworthy computing initiative :).
- Windows guts are rotten:).

References

- MSDN
- Programming Applications for MS Windows - Fourth Edition
- Process Explorer (www.sysinternals.com)
- WinObj (www.sysinternals.com)
- Rattle - Using Process Infection to Bypass Windows Software Firewalls (PHRACK #62)
- Crazylord - Playing with Windows /dev/(k)mem (PHRACK #59)



FIN

**APPLICATION
SECURITY, INC.**

- Questions?
- Thanks.
- Contact: sqlsec>at<yahoo>dot<com
- Argeniss – <http://www.argenis.com>