Win32API Interceptor
Monitoring Windows API calls

Kfir Karmon  (skarmont2.technion.ac.il)
Polina Shafran  (shpolina@t2.technion.ac.il)

Supervisors:
Dr. Ilana David  (ilana@ee.technion.ac.il)
Ben Bernstein, Microsoft  (bbern@microsoft.com)
The goal of this project is to build a system that intercepts API calls on a Windows server, and enables a user to monitor the API calls made by a certain process.
The system is built using the Detours technology, that was developed by Microsoft researchers.

- Allows to intercept Win32 API calls without having an access to the application’s source code.

- Interception code is applied dynamically at runtime.

- The target function code is modified in memory and not on the disk.
The Detours Technology (cont.)

- Replaces the first instructions of a *target* function (binary code) with an unconditional jump to a *detour* function.

- *Detour* function consists of some arbitrary instructions and a branch to the *trampoline* function.

- The *target* function contents are copied to the *trampoline* function.

- There is an unconditional jump to the remainder of the *target* function in the *trampoline* function.
The Detours Technology (cont.)

Diagram: How Detours change the original functions' calling sequence
The Detours Technology (cont.)

**Before Detours:**

<table>
<thead>
<tr>
<th>Target:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>push ebp</td>
<td>1 byte</td>
</tr>
<tr>
<td>mov ebp, esp</td>
<td>2 bytes</td>
</tr>
<tr>
<td>push ebx</td>
<td>1 byte</td>
</tr>
<tr>
<td>push esi</td>
<td>1 byte</td>
</tr>
<tr>
<td>push edi</td>
<td></td>
</tr>
<tr>
<td></td>
<td>....</td>
</tr>
</tbody>
</table>

**After Detours:**

<table>
<thead>
<tr>
<th>Target:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>jmp Detour</td>
<td>5 bytes</td>
</tr>
<tr>
<td>push edi</td>
<td></td>
</tr>
<tr>
<td></td>
<td>....</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detour:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>...Your code...</td>
<td></td>
</tr>
<tr>
<td>Call Trampoline</td>
<td></td>
</tr>
<tr>
<td>...More of your code...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trampoline:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>push ebp</td>
<td></td>
</tr>
<tr>
<td>mov ebp, esp</td>
<td></td>
</tr>
<tr>
<td>push ebx</td>
<td></td>
</tr>
<tr>
<td>push esi</td>
<td></td>
</tr>
<tr>
<td>jmp Target+5</td>
<td></td>
</tr>
</tbody>
</table>

Diagram: The code beneath the Detours mechanism
You can’t use Detours if:

1. The target function is less then 5 bytes long (sizeof(JMP)).
2. First bytes of the target code contain the destination of a branch instruction.
   
   ```
   Start:
   
   ;;;any code
   JNE  Start + 2
   ```
3. The address of the target code is unknown.
4. The platform is not x86 (no alpha or ia64 support).
5. Using .NET CLR (MSIL) code.
Win32API Interceptor – Architecture

1. Create COM Object
2. Create COM Object
3a. Spawn
3b. Inject
4. Post function call logging data
5. Extract logging data
6. Log function call data

Win32API Interceptor (MS Access Data Base)

DllInjectionAppLoader (COM Object)

InterceptLogger (COM Object)

\pipe\Win32APIInterceptor (OS pipe)

Spawned process (with the TraceAPI.DLL injected)

TraceAPI. DLL (Binary DLL with the Detours functions)

InterceptLogger

3b. Inject

4. Post function call logging data

5. Extract logging data

6. Log function call data

2. Create COM Object

1. Create COM Object
System Requirements

- x86 version of Windows NT/Windows 2000/Windows XP
- Microsoft Office Access 2003
- Administrator permissions
ODBC definition

- Start -> Settings -> Control Panel -> ODBC Data Source Administrator.

- Press Add in the User DSN section.
  Select Driver do Microsoft Access (*.mdb) and press Finish.

- Type Win32APIInterceptor in the Data Source Name field. Press Select button
  Select Win32APIInterceptor.mdb file from Win32APIInterceptor directory. Press OK.
Win32APIInterceptor installation

- Win32APIInterceptor -> Win32APIIntercept. Open Win32APIIntercept.sln with Microsoft Visual C++ .NET

- Select Build Solution from the Build menu and wait until the build process is complete.

- To start a Win32APIInterceptor, open the Win32APIInterceptor.mdb file (from Win32APIInterceptor directory) with Microsoft Access.
The main window

The main window consists of 4 sections:

1. Intercepted Executable section
2. Log section
3. Tools section (has one subsection – Filters)
4. Top Functions
Example
Cmd API calls(1)

1. **Interception:**

- Type `cmd` in *Executable location* field.
- Press *Start Interception* button.
- Wait a few seconds.
- Press *Refresh* button.
- Press *Stop Interception* button.

Note: Function counter stands on 282 now.

- Try to highlight *AutoRefresh* option and change the *timeout* field, during the interception.
Cmd API calls(2)

2. Filters

- Press **Clear** button in the **Filter** section.
- Add the **GetLocaleInfoW** function to the list.
- Highlight **Unwanted** option.
- Repeat through the interception steps.

Function counter stands on 144.

- Change **Unwanted** option to **Wanted**.
- Repeat through the interception steps.

Function counter stands on 138.

Note: $138 + 144 = 282$