Process Injection and Persistence using Application Shimming

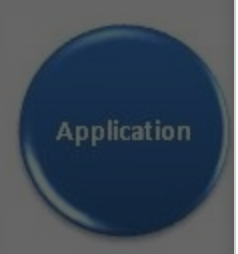
Nov 12, 2018

Microsoft provides Shims to developers mainly for backward compatibility, but malware can take advantage of shims to target an executable for both persistence and injection.

As the Windows operating system evolves from version to version, changes to the implementation of some functions may affect applications that depend on them.
Because of the nature of software, modifying the function again to resolve this compatibility issue could break additional applications or require Windows to remain the same regardless of the improvement that the alternative im

Using the Shim I r a particular application (and d outside the core Windows fu

The Shim Infrast oking: specifically, it lev ernative code (the shim).



from https://docs.r 10)

So, shims are e bles: Windows runs the Shim Er pply the appropriate fixes n persistence and injection.

How can be c

Microsoft allows anyone to create and install Shim database (sdb) files.
These database files contain the specific details on how Windows should manipulate (in other words 'shim') a target program with predefined 'Fixes'.
Microsoft provides also a free tool called the Application Compatibility Administrator which allows users to create and apply specific fixes such as '**DisableNX**','**ModifyShellLinkPath**','**VirtualRegistry**','**DisableAdvancedPCClientHardening**', '**ForceAdminAccess**','**InjectDll**', '**DisableSeh**','**ShellExecuteXP**' and many others.

After, the Application Compatibility Toolkit can be used to create and install a shim by guiding the user through a simple wizard.

The installer will create a GUID, copy the sdb file in to

```
%SystemRoot%\AppPatch\Custom\<GUID>.sdb
```

then add a registry key using an internal database namein the format of

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Custom\<GUID>.sdb
and
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\InstalledSDB\<GUID>.sdb
```

Obviously if a user/malware has administrative access,they could simply add the keys to the registry directly.

After the sdb file installation is complete all processes launched after that point will be subjected to the file matching rules of this shim database.

How shim cache can be useful for a malware?

Nearly every process is vulnerable to shim injection and all modern Windows OS versions support shims and natively ship with the auto-elevated shim database installer **sdbinst.exe**.

Custom fixes can be defined in the form of a user supplied DLL file, furthermore fixes are not considered executable even though they can contain shellcode.

The shim engine (**shimeng.dll**) will not shim certain hard coded modules such as: **NT Symbolic Debugger** (NTSD), **WinDbg** or **Software License Service** (slsvc.exe) and it will intercept **GetProcAddress()** in the event an application attempts to dynamically call a function that the shim engine has manipulated.

How analyze a shim database?

There are a few e … kit of python parsers for Appli …

Examples

`sdb_dump_raw.p…`

This script dump…

```
$python sdb_d…
<INDEXES>
  <INDEX>
    <INDEX_TA…
    <INDEX_KE…
    <INDEX_BI…
  </INDEX>
  <INDEX>
    <INDEX_TA…
    <INDEX_KE…
    <INDEX_FL…
    <INDEX_BI…
  </INDEX>
  <INDEX>
    <INDEX_TAG type='integer'>0x7007</INDEX_TAG>
    <INDEX_KEY type='integer'>0x600b</INDEX_KEY>
    <INDEX_FLAGS type='integer'>0x1</INDEX_FLAGS>
    <INDEX_BITS type='hex'>00000000000000000000000000000000000000000000000000000000000000
  </INDEX>
  <INDEX>
    <INDEX_TAG type='integer'>0x7007</INDEX_TAG>
    <INDEX_KEY type='integer'>0x6020</INDEX_KEY>
    <INDEX_FLAGS type='integer'>0x1</INDEX_FLAGS>
    <INDEX_BITS type='hex'>00000000000000000000000000000000000000000000000000000000000000
  </INDEX>
  <INDEX>
    <INDEX_TAG type='integer'>0x7007</INDEX_TAG>
    <INDEX_KEY type='integer'>0x9004</INDEX_KEY>
    <INDEX_FLAGS type='integer'>0x1</INDEX_FLAGS>
    <INDEX_BITS type='hex'>00000000000000000000000000000000000000000000000000000000000000
  </INDEX>
</INDEXES>
<DATABASE>
  <OS_PLATFORM type='integer'>0x1</OS_PLATFORM>
```

```
    <NAME type='stringref'>0x6</NAME>
    <DATABASE_ID type='guid'>XXXXXXX-XX-XXXX-XXXX-XXXX...</DATABASE_ID>
    <LIBRARY>
      <SHIM>
        <NAME type='stringref'>0x30</NAME>
        <DLLFILE type='stringref'>0x52</DLLFILE>
      </SHIM>
    </LIBRARY>
    <EXE>
      <NAME type='stringref'>0x7e</NAME>
      <APP_NAME type='stringref'>0x9c</APP_NAME>
      <EXE_ID type='hex'>YYYYY-YYY-YYY-YYYYY...</EXE_ID>
      <MATCHING_FILE>
        <NAME type='stringref'>0xbe</NAME>
      </MATCHING_FILE>
      <SHIM_REF>
        <NAME type='stringref'>0x30</NAME>
        <SHIM_TAGID type='integer'>0x47c</SHIM_TAGID>
      </SHIM_REF>
    </EXE>
  </DATABASE>
  <STRINGTABLE>
    <STRINGTABL
    <STRINGTABL
    <STRINGTABL
    <STRINGTABL
    <STRINGTABL
    <STRINGTABL
  </STRINGTABLE
```



```
sdb_dump_datab
```

This script dump s.

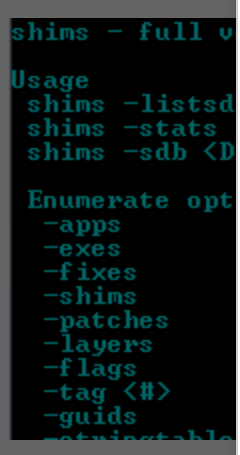
```
$python sdb_d
<DATABASE>
  <OS_PLATFOR
  <NAME type=
  <DATABASE_I
  <LIBRARY>
    <SHIM>
      <NAME t
      <DLLFIL
    </SHIM>
  </LIBRARY>
  <EXE>
    <NAME type='stringref'>calc.exe</NAME>
    <APP_NAME type='stringref'>XXXengine_Apps</APP_NAME>
    <EXE_ID type='hex'>YYYYY-YYY-YYY-YYYYY...</EXE_ID>
    <MATCHING_FILE>
      <NAME type='stringref'>*</NAME>
    </MATCHING_FILE>
    <SHIM_REF>
      <NAME type='stringref'>XXXengine_Shim</NAME>
      <SHIM_TAGID type='integer'>0x47c</SHIM_TAGID>
    </SHIM_REF>
  </EXE>
</DATABASE>
```

`sdb_dump_shims.py`

This script dumps the `DATABASE` element of a shim database, resolves value references, and substitutes complete shim definitions for `SHIM_REF` elements.

```
<DATABASE>
  <OS_PLATFORM type='integer'>0x1</OS_PLATFORM>
  <NAME type='stringref'>XXXengine_Database</NAME>
  <DATABASE_ID type='guid'>XXXXXXX-XXXX-XXX-XXX-XX...</DATABASE_ID>
  <LIBRARY>
    <SHIM>
      <NAME type='stringref'>XXXengine_Shim</NAME>
      <DLLFILE type='stringref'>Custom\xxx.dll</DLLFILE>
    </SHIM>
  </LIBRARY>
  <EXE>
    <NAME type='stringref'>calc.exe</NAME>
    <APP_NAME type='stringref'>XXXengine_Apps</APP_NAME>
    <EXE_ID type='hex'>YYYYY-YYY-YYY-YYYYY...</EXE_ID>
    <MATCHING_FILE>
      <NAME type='stringref'>*</NAME>
    </MATCHING_FILE>
    <SHIM>
      <!-- SHIM_REF name:'XXXengine_Shim' offset:0x47c -->
      <NAME type='stringref'>XXXengine_Shim</NAME>
      <DLLFILE type='stringref'>Custom\xxx.dll</DLLFILE>
    </SHIM>
  </EXE>
</DATABASE>
```

Another useful t



Referenc

- Understandi
- Microsoft Ap
- python-sdb
- Windows Sh

Andrea Fortuna

Andrea Fortuna
andrea@andreafortuna.org

andreafortuna
andrea-fortuna
andrea

Cybersecurity expert, software developer, experienced digital forensic analyst, musician