Multiple Ways To
Defeat Office Genuine Advantage (OGA)

29th Jan, 2007

Straight from WiKi (http://en.wikipedia.org/wiki/Office_Genuine_Advantage) :

Office Genuine Advantage (OGA) is a program currently (as of 2006-05-03) being piloted by Microsoft similar to Windows Genuine Advantage in that it requires users of the Microsoft Office software to validate their copy of Microsoft Office.

Currently, Microsoft Office software is validated by using something called the Office Validation Assistant (OVA) which is available on Microsoft's web site. Validation using the OVA is not presently required. Use of Office Genuine Advantage to obtain Office updates and downloads, as of 2006-10-27 is mandatory in most countries.

Starting in January 2007, users of Office Update will have to validate the legitimacy of their Office software before they can use the service. Pirated copies (that is, copies that won't pass product activation) of Microsoft Office 2007 will work in a "reduced functionality mode" (in which you can view but can't edit any document)

Whilst there may be multiple ways to defeat such software piracy control, in this document I will explain two simple methods to defeat Office Genuine Advantage validation check.

Method 1: Bypass OGA by using Google search

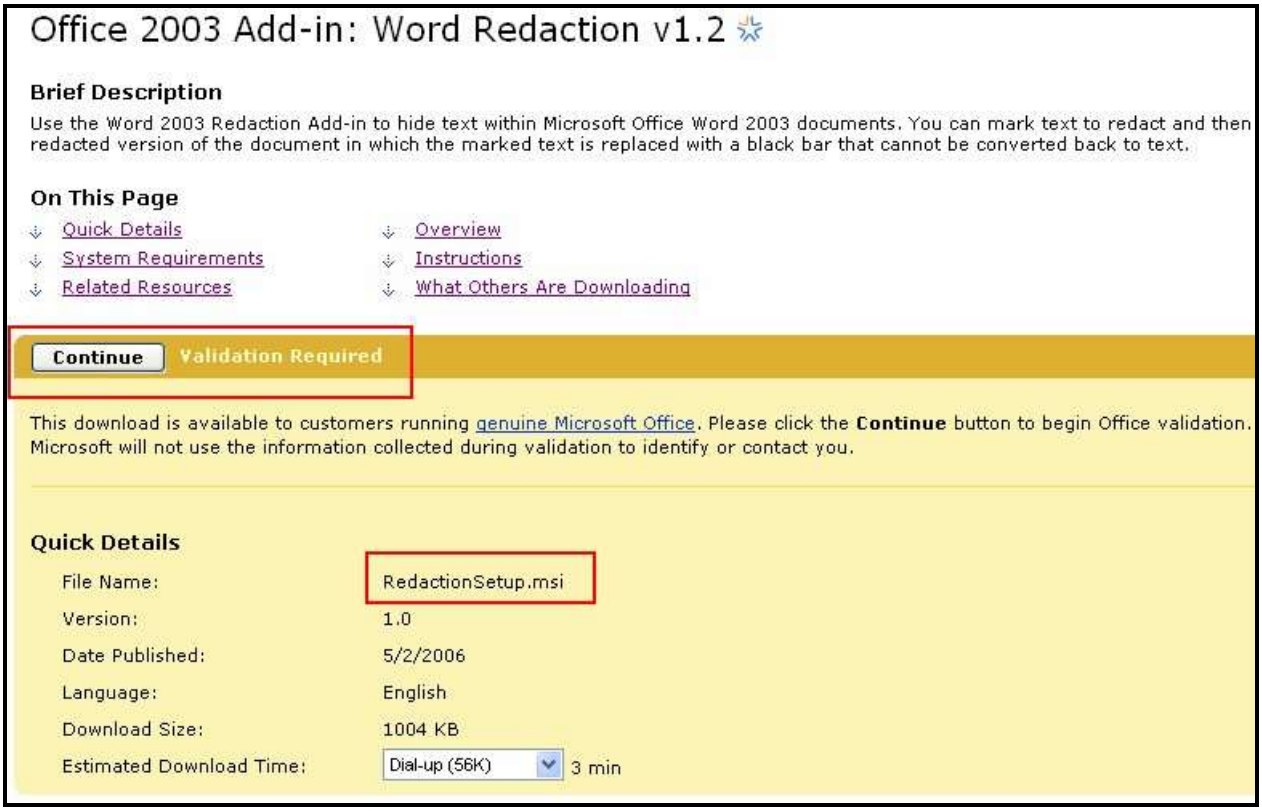
This a very simple bypass method. Check out this download link for Office related updates or add-ins http://office.microsoft.com/en-us/downloads/default.aspx.

Select the Office version from the above link and try downloading any updates/add-ins related to that specific version. It will lead to a validation link (Refer the screenshot below) which will check whether the office installed in the system is genuine or not.

In short, the OGA validation generates a hash out of information gathered from the installed MS Office and passes it on to the server for verification. The user will be directed to the download link only if the installed office version is genuine.

To bypass such validation, follow these simple steps:

- Select the add-ins / updates that you require for office
- In the Validation required page, copy the file name (Refer to the following screenshot)



- Use google search for the filename using the following search keyword combination:
  site:download.microsoft.com/download <File Name>

  For Example: To search for the file name “RedactionSetup.msi” use the following search key combination site:download.microsoft.com/download RedactionSetup.msi

- The above search result provides the direct link to the file available on the Microsoft download server which can be used to directly download the file without any validation check (Refer to the following screenshot).

Method 2: Bypass OGA by re-using the product hash

It is possible to re-use the hash generated in any machine having a genuine copy of MS Office and can be distributed over the internet.

For example – The table below has hashes for downloading rhdtool.exe and RedactionSetup.msi which can be re-used by any user having pirated copy of Office to defeat OGA validation check.

http://www.microsoft.com/downloads/details.aspx?FamilyID=144E54ED-D43E-42CA-BC7B5446D34E5360&displaylang=en&Hash=[Put the hash here]

Office 2003/XP Add-in:	Remove Hidden Data [rhdtool.exe]
FamilyID	144E54ED-D43E-42CA-BC7B-5446D34E5360
Hash	GV44ca[redacted]ST43JYu2tbJlUyCgXIkWLgWm5GVMtHsO61Dnfq Qr[redacted]==

Office 2003/XP Add-in:	Word Redaction v1.2 [RedactionSetup.msi]
FamilyID	144E54ED-D43E-42CA-BC7B-5446D34E5360
Hash	[redacted]+ow+1+ZIEDGnzI+W4yRMKLNw7zwHNJZ8vtrYFeoBYkG++lz7b FByo[redacted]

Method 3: A Quick Analysis of OGACheckControl.dll

Last but not the least is the reverse analysis of OGACheckControl.dll and creating a patched version to bypass OGA check.

I did a quick reverse analysis and found few interesting sections which confirmed that creating a patch, in this case, is going to be much simpler than I thought it to be. However, I do not intend to release any binary patch to bypass the OGA check in the wild at this stage and get my name tagged as a software pirate. Therefore, sharing these details only for reference only.

Regardless, those who know how to create a binary patch or have minimal knowledge of reversing will easily figure out themselves.

- **Call to Hash creation routine**

```
:00427A72 loc_427A72:                             ; CODE XREF: sub_42784F+1F2↑j
:00427A72                 lea     eax, [ebp+hHash]
:00427A75                 push    eax             ; phHash
:00427A76                 push    ebx             ; dwFlags
:00427A77                 push    ebx             ; hKey
:00427A78                 push    8003h           ; Algid
:00427A7D                 push    [ebp+hProv]     ; hProv
:00427A80                 call    ds:CryptCreateHash
:00427A86                 test    eax, eax
:00427A88                 jnz     short loc_427AB1
:00427A8A                 call    ds:__imp_GetLastError
:00427A90                 cmp     eax, ebx
:00427A92                 jle     short loc_427A9E
:00427A94                 and     eax, 0FFFFh
:00427A99                 or      eax, 80070000h
```

```
:00427AB1 loc_427AB1:                             ; CODE XREF: sub_42784F+239↑j
:00427AB1                 cmp     esi, ebx
:00427AB3                 jl      loc_427BA0
:00427AB9                 push    ebx             ; dwFlags
:00427ABA                 push    6Dh             ; dwDataLen
:00427ABC                 lea     eax, [ebp+pbData]
:00427ABF                 push    eax             ; pbData
:00427AC0                 push    [ebp+hHash]     ; hHash
:00427AC3                 call    ds:CryptHashData
:00427AC9                 test    eax, eax
:00427ACB                 jnz     short loc_427AF4
:00427ACD                 call    ds:__imp_GetLastError
:00427AD3                 cmp     eax, ebx
:00427AD5                 jle     short loc_427AE1
:00427AD7                 and     eax, 0FFFFh
:00427ADC                 or      eax, 80070000h
```

```
:00427A57 loc_427A57:                             ; CODE XREF: sub_42784F+1FC↑j
:00427A57                 push    ebx
:00427A58                 mov     esi, eax
:00427A5A                 push    esi
:00427A5B                 push    122h
:00427A60                 push    offset aD4ab90a5 ; "D4AB90A5"
:00427A65                 call    sub_421227
:00427A6A                 cmp     esi, ebx
:00427A6C                 jl      loc_427BA0
```

- **Modification of OGA code and 'subcode' value in registry:** This can be patched to keep the value as code=100 and subcode=200 always

```
:0042AFC0 ; int __stdcall sub_42AFC0(int,BYTE Data)
:0042AFC0 sub_42AFC0      proc near               ; CODE XREF: sub_42B0F8+73↓p
:0042AFC0
:0042AFC0 hKey            = dword ptr -10Ch
:0042AFC0 SubKey          = byte ptr -108h
:0042AFC0 var_4           = dword ptr -4
:0042AFC0 arg_0           = dword ptr  8
:0042AFC0 Data            = byte ptr  0Ch
:0042AFC0
:0042AFC0                 push    ebp
:0042AFC1                 mov     ebp, esp
:0042AFC3                 sub     esp, 10Ch
:0042AFC9                 mov     eax, dword_4975A0
:0042AFCE                 xor     eax, ebp
:0042AFD0                 mov     [ebp+var_4], eax
:0042AFD3                 push    ebx
:0042AFD4                 push    esi
:0042AFD5                 push    edi
:0042AFD6                 xor     ebx, ebx
:0042AFD8                 push    103h
:0042AFDD                 lea     eax, [ebp-107h]
:0042AFE3                 push    ebx
:0042AFE4                 push    eax
:0042AFE5                 mov     esi, ecx
:0042AFE7                 mov     [ebp+hKey], ebx
:0042AFED                 mov     [ebp+SubKey], bl
:0042AFF3                 call    sub_41218F
```

```
:0042AFF8                 add     esp, 0Ch
:0042AFFB                 push    [ebp+arg_0]     ; Data
:0042AFFE                 mov     ecx, esi
:0042B000                 call    sub_421929
:0042B005                 push    ebx
:0042B006                 mov     esi, eax
:0042B008                 push    esi
:0042B009                 push    101h
:0042B00E                 mov     edi, offset a81af8c18 ; "81AF8C18"
:0042B013                 push    edi
:0042B014                 call    sub_421227
:0042B019                 cmp     esi, ebx
:0042B01B                 jl      loc_42B0D1
:0042B021                 push    offset aOfficeGenuineA ; "Office Genuine Advantage"
:0042B026                 push    offset aSoftwareMicros ; "SOFTWARE\\Microsoft\\%s"
:0042B02B                 lea     eax, [ebp+SubKey]
:0042B031                 push    104h
:0042B036                 push    eax
:0042B037                 call    sub_40F560
:0042B03C                 add     esp, 10h
:0042B03F                 push    ebx
:0042B040                 mov     esi, eax
:0042B042                 push    esi
:0042B043                 push    107h
:0042B048                 push    edi
:0042B049                 call    sub_421227
:0042B04E                 cmp     esi, ebx
:0042B050                 jl      short loc_42B0D1
```

```
:0042B052                push    ebx             ; lpdwDisposition
:0042B053                lea     eax, [ebp+hKey]
:0042B059                push    eax             ; phkResult
:0042B05A                push    ebx             ; lpSecurityAttributes
:0042B05B                push    2001Fh          ; samDesired
:0042B060                push    ebx             ; dwOptions
:0042B061                push    ebx             ; lpClass
:0042B062                push    ebx             ; Reserved
:0042B063                lea     eax, [ebp+SubKey]
:0042B069                push    eax             ; lpSubKey
:0042B06A                push    80000001h       ; hKey
:0042B06F                call    ds:RegCreateKeyExA
:0042B075                cmp     eax, ebx
:0042B077                jz      short loc_42B094
:0042B079                jle     short loc_42B085
:0042B07B                and     eax, 0FFFFh
:0042B080                or      eax, 80070000h
```

```
:0042B085
:0042B085 loc_42B085:                            ; CODE XREF: sub_42AFC0+B9↑j
:0042B085                push    ebx
:0042B086                mov     esi, eax
:0042B088                push    esi
:0042B089                push    119h
:0042B08E                push    edi
:0042B08F                call    sub_421227
:0042B094
:0042B094 loc_42B094:                            ; CODE XREF: sub_42AFC0+B7↑j
:0042B094                cmp     esi, ebx
:0042B096                jl      short loc_42B0D1
:0042B098                push    4               ; cbData
:0042B09A                lea     eax, [ebp+Data]
:0042B09D                push    eax             ; lpData
:0042B09E                push    4               ; dwType
:0042B0A0                push    ebx             ; Reserved
:0042B0A1                push    offset aSubcode ; "subcode"
:0042B0A6                push    [ebp+hKey]      ; hKey
:0042B0AC                call    ds:RegSetValueExA
:0042B0B2                cmp     eax, ebx
:0042B0B4                jz      short loc_42B0D1
:0042B0B6                jle     short loc_42B0C2
:0042B0B8                and     eax, 0FFFFh
:0042B0BD                or      eax, 80070000h
:0042B0C2
```