

HISPASEC SISTEMAS

SEGURIDAD Y TECNOLOGÍAS
DE LA INFORMACIÓN

**Documento técnico:
Análisis de la vulnerabilidad de cargador de
punteros de ratón animados
en Microsoft Windows Vista**

abril 2007

1. Resumen

Microsoft Window Vista, al igual que las versiones anteriores, soporta los punteros de ratón animados. Los punteros animados de ratón son cargados desde ficheros .ANI a través de las funciones en USER32.DLL llamadas LoadCursorIconFromResource y LoadCursor. Estas funciones utilizan una función interna llamada _LoadCursorIconFromFileMap la que, bajo ciertas condiciones, se presta a sufrir un desbordamiento de memoria intermedia.

2. Descripción de la vulnerabilidad

Para comprender la vulnerabilidad, se requiere un conocimiento básico sobre el formato de ficheros .ANI. El fichero .ANI comienza con el “magic value” “RIFF”, seguido de un DWORD que contiene la longitud de los datos seguidos del “magic value” “ACON”. El resto del fichero está basado en fragmentos (“chunks”). Cada bloque o fragmento comienza con un “word” identificativo de cuatro bytes (“anih” para cabeceras ANI, “fram” para frames, etc) y un DWORD que contiene la longitud del bloque. Por ejemplo, la longitud de la cabecera ANI “anih” es siempre de longitud 36 bytes, así que los ocho bytes del fragmento serían “anih\x24\0\0”.

La función interna responsable de la lectura del puntero animado desde el fichero, se llama _LoadCursorIconFromFileMap y está localizada en USER32.DLL. La función intenta determinar si el fichero tiene más de un “frame” y, si es así, llama a otra función interna llamada _LoadAniIcon. El pseudo código para esta función es:

```
riff_magic = Read 4 bytes ; read magic
if riff_magic != "RIFF"
return
Skip 4 bytes ; skip length
acon_magic = Read 4 bytes ; read ACON magic
if acon_magic != "ACON"
return
tag = Read 8 bytes ; tag is a struct (magic, size)
if tag.magic == "anih"
if tag.size != 0x24 ; size check
return
header = Read tag.size bytes ; read the chunk
result = ValidateAnih(header) ; validate the header
if !result
return
if header.cFrames > 1 ; are there two or more frames?
call _LoadAniIcon ; execute the LoadAniIcon function
return
...
```

El tamaño de la cabecera se verifica en esta función en dos partes. Primero en “if tag.size != 0x24”, y luego a través de la función ValidateAnih. Si hay más de un “frame” se llama a _LoadAniIcon. El pseudo código para la función es:

```

Rewind ; rewinds the file
Skip 8 bytes
acon_magic = Read 4 bytes ; read the ACON magic once again
if acon_magic != "ACON"
return
loop:
tag = Read 8 bytes ; tag is a struct (magic, size)
if tag.magic == "seq "
...
else if tag.magic == "LIST"
...
else if tag.magic == "rate"
...
else if tag.magic == "anih" ; is this the header ?
header = Read tag.size bytes ; read the header
result = ValidateAnih(header) ; validate the header
if !result
return
...
else
Skip tag.size bytes
goto loop
  
```

Los fragmentos son leídos en un bucle secuencial y manejados si son soportados. El problema reside en la línea donde se encuentra el valor “anih” y la cabecera que es leída. Puesto que el tamaño no es validado, este puede ser proporcionado de forma incorrecta y provocar que los datos de la pila se sobrescriban. Es necesario notar que el tamaño de la primera cabecera “anih” es validada en la función padre, pero teniendo en cuenta que no hay mecanismo que compruebe si el fragmento “anih” ya fue proporcionado, podría ser proporcionado más de una vez y en segunda instancia no sería verificado antes de ser leído. La función ValidateAnih, llamada después de la lectura, determinaría la cabecera como inválida, puesto que el bloque se diferencia en 0x24 bytes, pero ya sería demasiado tarde.

3. Análisis del malware

Un malware que ha sido analizado y usaba esta vulnerabilidad ha sido Trojan-Downloader.Win32.Ani.g. Después de desbordar la memoria intermedia (buffer overflow), ejecuta LoadLibraryA para cargar URLMON.DLL. Posteriormente descarga un ejecutable especificado (usando la función URLDownloadToCacheFileA), y lo ejecuta usando CreateProcessA(“cmd”, “/c <nombre>”,...). Los offsets de las funciones

usadas no están incrustadas (“hardcoded”) sino que el malware tiene una función similar a GetProcAddress propia incluida que usa un algoritmo de hash simple para localizar la función correcta. Por ejemplo, la función URLDownloadToCacheFileA se adquiere usando el siguiente código:

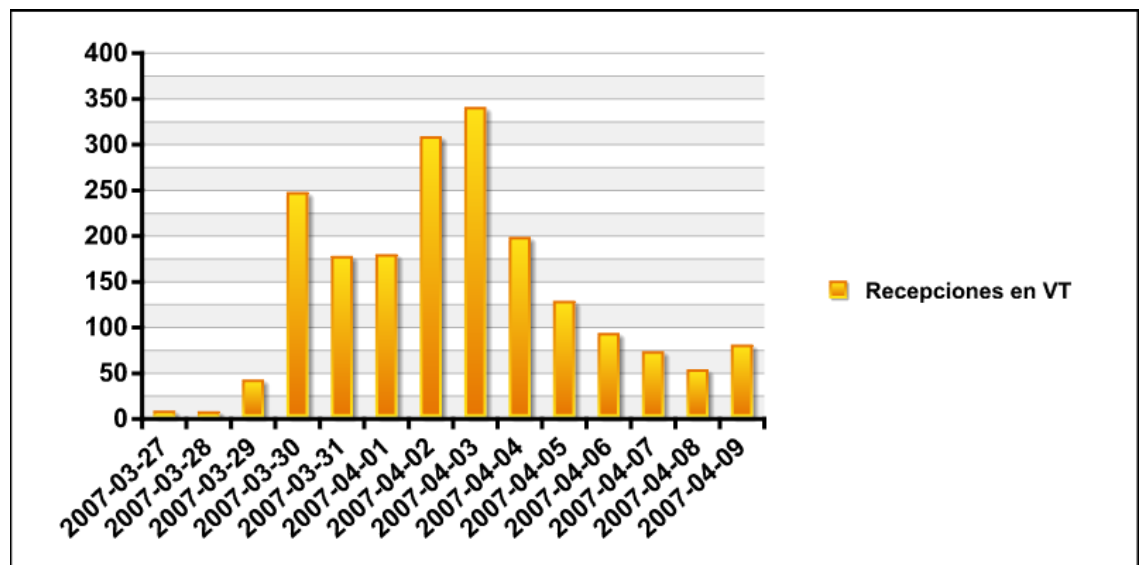
```
call sub_1A0 ; this acquires the handle to URLMON.DLL
push 54FEF4Fh ; the hash of the sought function
push eax ; the handle to URLMON.DLL
call get_func_by_hash ; the right call
```

```
get_func_by_hash:
...
try_next_func:
...
calc_hash: ; this loop calculates the hash
lodsb
test al,al
jz hash_done
ror edi, 0x0D ; the heart of the hash
add edi, eax ;
jmp calc_hash
cmp edi, [arg_sought_hash]
jnz try_next_func
...
```

El uso de estas funciones lo hace un poco más independiente de la versión del sistema.

4. Recepción en VirusTotal

La gráfica ofrece el número de muestras que aprovechan la vulnerabilidad ANI recibidas en VirusTotal del 27-03-2007 al 09-04-2007. El número de muestras únicas recibidas en total es de 1037 (según hash MD5) en este periodo.



5. Resumen

Es realmente sencillo crear un exploit para esta vulnerabilidad. Se recomienda actualizar Microsoft Windows con el correspondiente parche que soluciona este problema.

El análisis ha sido realizado sobre Microsoft Windows Vista versión 6.0.6000.16386.