



SA32005 / CVE-2008-3862

Generated by Secunia

22 October, 2008

9 pages

Table of Contents

Introduction	2
Technical Details	2
Exploitation	7
Characteristics	8
Tested Versions	9
Fixed Versions	9
References	9

Introduction:

=====

A vulnerability in Trend Micro OfficeScan during processing of form data in CGI requests can be exploited by malicious people to compromise a user's system.

Technical Details:

=====

Trend Micro OfficeScan includes numerous CGI executables, which are used by OfficeScan client agents and the web management console and are accessible by default via e.g. "http://<host>:8080/officescan[/console]/cgi" or "https://<host>:4343/officescan[/console]/cgi (for version 7.3). The HTTP interface is exposed by either IIS or Apache (included in the installer).

NOTE: This analysis is performed using the included Apache HTTP server.

Most of the CGI executables use a generic CGI parameter parsing implementation, which contains a vulnerability. Insufficient bounds checking results in a stack-based buffer overflow when parsing the contents of HTTP "multipart/form-data" that can be leveraged for arbitrary code execution.

When an OfficeScan CGI executable runs, it calls the "CGetParameter" constructor to create an object containing all CGI variables.

This example will follow the case of cgiRqOPP.exe.

```
.text:00401000 _main          proc near          ; CODE XREF: start+C0
.text:00401000
.text:00401000 pp_oppActivate = dword ptr -224h
.text:00401000 var_220         = dword ptr -220h
.text:00401000 p_release       = dword ptr -21Ch
.text:00401000 pp_uid          = dword ptr -218h
.text:00401000 CGetParameterObj = dword ptr -214h
.text:00401000 p_chkDatabase    = dword ptr -210h
.text:00401000 WideCharStr     = word ptr -20Ch
.text:00401000 var_C           = dword ptr -0Ch
.text:00401000 var_4           = dword ptr -4
.text:00401000 argc            = dword ptr 4
.text:00401000 argv            = dword ptr 8
.text:00401000 envp            = dword ptr 0Ch
.text:00401000
.text:00401000          push     0FFFFFFFh
.text:00401002          push     offset unknown_libname_64 ; Microsoft VisualC 2-9/net runtime
.text:00401007          mov     eax, large fs:0
.text:0040100D          push     eax
.text:0040100E          mov     large fs:0, esp
.text:00401015          sub     esp, 218h
.text:0040101B          push     ebx
.text:0040101C          push     esi
.text:0040101D          lea    ecx, [esp+22Ch+CGetParameterObj]
.text:00401021          call   ??0CGetParameter@@QAE@XZ ; CGetParameter::CGetParameter(void)
```

Eventually sub_402EB5() is called to parse the HTTP request.

CGI executables receive parameters via environment variables, hence _getenv() is used to retrieve the HTTP request method.

```

.text:00402EB5 sub_402EB5      proc near          ; CODE XREF: sub_402CAE+29
.text:00402EB5
.text:00402EB5 THIS          = dword ptr -40h
.text:00402EB5 var_3C        = dword ptr -3Ch
.text:00402EB5 var_38        = dword ptr -38h
.text:00402EB5 var_34        = byte ptr -34h
.text:00402EB5 var_24        = dword ptr -24h
.text:00402EB5 params        = PARAMS ptr -20h
.text:00402EB5 lpzRequestMethod= dword ptr -10h
.text:00402EB5 var_C         = dword ptr -0Ch
.text:00402EB5 var_4         = dword ptr -4
.text:00402EB5 arg_0         = dword ptr 8
.text:00402EB5
.text:00402EB5      push    ebp
.text:00402EB6      mov     ebp, esp
.text:00402EB8      push    0FFFFFFFh
.text:00402EBA      push    offset unknown_libname_69 ; Microsoft VisualC 2-9/net runtime
.text:00402EBF      mov     eax, large fs:0
.text:00402EC5      push    eax
.text:00402EC6      mov     large fs:0, esp
.text:00402ECD      sub     esp, 34h
.text:00402ED0      mov     [ebp+THIS], ecx
.text:00402ED3      mov     eax, [ebp+arg_0] ; always zero
.text:00402ED6      and     eax, 0FFh
.text:00402EDB      test    eax, eax
.text:00402EDD      jz     short loc_402EEC ; jump always taken
...
.text:00402EEC loc_402EEC      ; CODE XREF: sub_402EB5+28
.text:00402EEC      push    offset aRequest_method ; "REQUEST_METHOD"
.text:00402EF1      call   wrap_getenv
.text:00402EF6      add     esp, 4
.text:00402EF9      mov     [ebp+lpzRequestMethod], eax
.text:00402EFC      cmp     [ebp+lpzRequestMethod], 0
.text:00402F00      jnz    short loc_402F0C ; got request method?

```

The request method is checked to see whether it is "GET" or "POST".

```

.text:00402F0C loc_402F0C      ; CODE XREF: sub_402EB5+4B
.text:00402F0C      push    offset aGet          ; "GET"
.text:00402F11      mov     ecx, [ebp+lpzRequestMethod]
.text:00402F14      push    ecx                  ; Str1
.text:00402F15      call   _strcmp
.text:00402F1A      add     esp, 8
.text:00402F1D      test    eax, eax
.text:00402F1F      jnz    short loc_402F82 ; not "GET"?
...
... "GET" handling
...
.text:00402F82 loc_402F82      ; CODE XREF: sub_402EB5+6A
.text:00402F82      push    offset aPost         ; "POST"
.text:00402F87      mov     edx, [ebp+lpzRequestMethod]
.text:00402F8A      push    edx                  ; Str1
.text:00402F8B      call   _strcmp
.text:00402F90      add     esp, 8
.text:00402F93      test    eax, eax
.text:00402F95      jnz    loc_403083           ; not "POST"?

```

If the request method is "POST" and the "Content-Type" begins with "multipart/form-data", sub_4032B9() is called to parse the content.

```

.text:00402F9B      push    offset aContent_type_1 ; "CONTENT_TYPE"
.text:00402FA0      call   wrap_getenv
.text:00402FA5      add     esp, 4
.text:00402FA8      mov     [ebp+lpzContentType], eax
.text:00402FAB      cmp     [ebp+lpzContentType], 0
.text:00402FAF      jz     short loc_402FE1
.text:00402FB1      push    offset aMultipartFormD ; "multipart/form-data"
.text:00402FB6      call   _strlen
.text:00402FBB      add     esp, 4
.text:00402FBE      push    eax                  ; MaxCount
.text:00402FBF      push    offset aMultipartFor_0 ; "multipart/form-data"
.text:00402FC4      mov     eax, [ebp+lpzContentType]
.text:00402FC7      push    eax                  ; Str1
.text:00402FC8      call   __strnicmp
.text:00402FCD      add     esp, 0Ch
.text:00402FD0      test    eax, eax
.text:00402FD2      jnz    short loc_402FE1 ; not "multipart/form-data"?
.text:00402FD4      mov     ecx, [ebp+THIS]
.text:00402FD7      call   sub_4032B9           ; parse multipart/form-data

```

In sub_4032B9(), after some insignificant processing (not shown in disassembly), the "Content-Length" header is retrieved and converted to an integer and saved.

```
.text:00403321      push   offset aContent_leng_6 ; "CONTENT_LENGTH"
.text:00403326      call   wrap_getenv
.text:0040332B      add    esp, 4
.text:0040332E      test   eax, eax
.text:00403330      jz     short loc_403350 ; no "content-length" header?
.text:00403332      push   offset aContent_leng_7 ; "CONTENT_LENGTH"
.text:00403337      call   wrap_getenv
.text:0040333C      add    esp, 4
.text:0040333F      push   eax                ; Str
.text:00403340      call   _atoi
.text:00403345      add    esp, 4
.text:00403348      mov    [ebp+contentLength], eax
```

A 4096-byte temporary buffer is allocated and the pointer saved in the containing object.

```
.text:004033EF loc_4033EF:                ; CODE XREF: sub_4032B9+10D
.text:004033EF      mov    ecx, [ebp+THIS]
.text:004033F5      mov    [ecx+TempBuffer.bufferSize], 1000h
.text:004033FC      mov    edx, [ebp+THIS]
.text:00403402      mov    [edx+TempBuffer.bufferOffset], 0
.text:00403409      mov    eax, [ebp+THIS]
.text:0040340F      mov    ecx, [eax+TempBuffer.bufferSize]
.text:00403412      push   ecx                ; Size
.text:00403413      call   _malloc            ; allocate 4096 byte buffer
.text:00403418      add    esp, 4
.text:0040341B      mov    edx, [ebp+THIS]
.text:00403421      mov    [edx+TempBuffer.lpBuffer], eax
.text:00403424      mov    eax, [ebp+THIS]
.text:0040342A      mov    ecx, [eax+TempBuffer.bufferSize]
.text:0040342D      push   ecx                ; Size
.text:0040342E      push   0                  ; Val
.text:00403430      mov    edx, [ebp+THIS]
.text:00403436      mov    eax, [edx+TempBuffer.lpBuffer]
.text:00403439      push   eax                ; Dst
.text:0040343A      call   _memset            ; zero out buffer
```

The boundary value is parsed from the "Content-Type" header and up to 260 characters are copied into a stack buffer.

```
.text:0040347A      push   104h              ; Count
.text:0040347F      push   '='                ; Val
.text:00403481      push   offset aContent_type_3 ; "CONTENT_TYPE"
.text:00403486      call   wrap_getenv
.text:0040348B      add    esp, 4
.text:0040348E      push   eax                ; Str
.text:0040348F      call   _strchr            ; find '=' indicating boundary in content-type header
.text:00403494      add    esp, 8
.text:00403497      add    eax, 1
.text:0040349A      push   eax                ; Source
.text:0040349B      lea   ecx, [ebp+szBoundary] ; char[260]
.text:004034A1      push   ecx                ; Dest
.text:004034A2      call   _strncpy
.text:004034A7      add    esp, 0Ch
.text:004034AA      lea   edx, [ebp+szBoundary] ; char[260]
.text:004034B0      push   edx                ; Str
.text:004034B1      call   _strlen
.text:004034B6      add    esp, 4
.text:004034B9      mov    [ebp+len_boundary], eax
```

A loop is entered that iterates over each entry in the form data. The content of the HTTP request is available on stdin for the CGI executable.

The temporary buffer is filled up (or until all bytes have been read as specified by the "Content-Length" header) by a call to fread().

```
.text:004034BF      mov    [ebp+szName], 0 ; char[260]
.text:004034C6      mov    [ebp+szDestBuffer], 0 ; char[260]
.text:004034CD      mov    [ebp+szFileName], 0 ; char[260]
.text:004034D4
.text:004034D4 @loop_items:                ; CODE XREF: sub_4032B9+625
.text:004034D4      mov    eax, [ebp+contentLength_]
.text:004034DA      sub    eax, [ebp+bytesRead]
.text:004034E0      mov    [ebp+Count], eax
.text:004034E3      mov    ecx, [ebp+THIS]
.text:004034E9      mov    edx, [ebp+THIS]
```

```

.text:004034EF      mov     eax, [ecx+TempBuffer.bufferSize]
.text:004034F2      sub     eax, [edx+TempBuffer.bufferOffset]
.text:004034F5      cmp     [ebp+Count], eax
.text:004034F8      jle     short loc_40350F ; can fit all data in buffer?
.text:004034FA      mov     ecx, [ebp+THIS]
.text:00403500      mov     edx, [ebp+THIS]
.text:00403506      mov     eax, [ecx+TempBuffer.bufferSize]
.text:00403509      sub     eax, [edx+TempBuffer.bufferOffset]
.text:0040350C      mov     [ebp+Count], eax
.text:0040350F      loc_40350F:
.text:0040350F      ; CODE XREF: sub_4032B9+23F
                cmp     [ebp+Count], 0
.text:00403513      jle     short loc_403562 ; no data to read?
.text:00403515      push   offset File ; File
.text:0040351A      mov     ecx, [ebp+Count]
.text:0040351D      push   ecx ; Count
.text:0040351E      push   1 ; ElementSize
.text:00403520      mov     edx, [ebp+THIS]
.text:00403526      mov     eax, [edx+TempBuffer.lpBuffer]
.text:00403529      mov     ecx, [ebp+THIS]
.text:0040352F      add     eax, [ecx+TempBuffer.bufferOffset]
.text:00403532      push   eax ; DstBuf
.text:00403533      call   wrap_fread

```

The buffer is then tokenised by splitting on newlines ('\r' or '\n'), colons (':'), semi-colons (';'), and equal signs ('=').

```

.text:00403562 loc_403562:
                ; CODE XREF: sub_4032B9+25A
.text:00403562      lea     eax, [ebp+szDelim] ; ":\r\n"
.text:00403568      push   eax ; Delim
.text:00403569      mov     ecx, [ebp+THIS]
.text:0040356F      mov     edx, [ecx+TempBuffer.lpBuffer]
.text:00403572      push   edx ; Str
.text:00403573      call   _strtok
.text:00403578      add     esp, 8
.text:0040357B      mov     [ebp+lpszToken], eax
.text:00403581      cmp     [ebp+lpszToken], 0
.text:00403588      jnz     short @loop_parse_headers ; found a token?

```

If such a delimiter is found, the form subpart headers are parsed in a loop.

```

.text:0040358F @loop_parse_headers:
                ; CODE XREF: sub_4032B9+2CF
.text:0040358F      mov     [ebp+lpszValue], 0
.text:00403596      push   offset aContentType ; "Content-type"
.text:0040359B      mov     eax, [ebp+lpszToken]
.text:004035A1      push   eax ; Str1
.text:004035A2      call   __strcmpi
.text:004035A7      add     esp, 8
.text:004035AA      test   eax, eax
.text:004035AC      jnz     short loc_4035C7 ; not "content-type"?

```

If e.g. a "filename" header is found, a pointer to its value is saved. Up to 260 characters from the optionally quoted value are copied into a stack buffer by sub_40397C() (not shown in disassembly).

```

.text:004035F8 loc_4035F8:
                ; CODE XREF: sub_4032B9+324
.text:004035F8      push   offset SubStr ; "filename"
.text:004035FD      mov     ecx, [ebp+lpszToken]
.text:00403603      push   ecx ; Str
.text:00403604      call   _strrstr
.text:00403609      add     esp, 8
.text:0040360C      test   eax, eax
.text:0040360E      jz     short loc_403641 ; "filename" not found?
.text:00403610      lea     edx, [ebp+szDelims2] ; ":\r\n"
.text:00403616      push   edx ; Delim
.text:00403617      push   0 ; Str
.text:00403619      call   _strtok ; get "filename" value
.text:0040361E      add     esp, 8
.text:00403621      mov     [ebp+lpszValue], eax
.text:00403624      push   104h ; Count
.text:00403629      lea     eax, [ebp+szFileName] ; char[260]
.text:0040362F      push   eax ; lpszDest
.text:00403630      mov     ecx, [ebp+lpszValue]
.text:00403633      push   ecx ; lpszStr
.text:00403634      mov     ecx, [ebp+THIS]
.text:0040363A      call   sub_40397C ; copy without quotes

```

In addition to the "filename" header, the "content-type", "content-disposition", and "name" headers are searched for (not shown in disassembly).

If a token was found, then it is checked to see if it is followed by two newlines "\r\n\r\n", which signifies the end of the subpart headers.

```
.text:00403688 loc_403688:                ; CODE XREF: sub_4032B9+309
.text:00403688                cmp     [ebp+lpszValue], 0
.text:0040368C                jz     loc_40389C        ; no token?
.text:00403692                mov     eax, [ebp+lpszValue]
.text:00403695                push   eax                ; Str
.text:00403696                call   _strlen
.text:0040369B                add     esp, 4
.text:0040369E                mov     ecx, [ebp+lpszValue]
.text:004036A1                lea    edx, [ecx+eax+1]
.text:004036A5                mov     [ebp+lpszValue], edx
.text:004036A8                mov     eax, [ebp+lpszValue]
.text:004036AB                movsx  ecx, byte ptr [eax]
.text:004036AE                cmp     ecx, 0Ah
.text:004036B1                jnz    loc_40389C        ; not \n?
.text:004036B7                mov     edx, [ebp+lpszValue]
.text:004036BA                movsx  eax, byte ptr [edx+1]
.text:004036BE                cmp     eax, 0Dh
.text:004036C1                jnz    loc_40389C        ; not \r?
.text:004036C7                mov     ecx, [ebp+lpszValue]
.text:004036CA                movsx  edx, byte ptr [ecx+2]
.text:004036CE                cmp     edx, 0Ah
.text:004036D1                jnz    loc_40389C        ; not \r\n\r\n?
```

If the end of the headers is found, data is moved in the temporary file buffer so the next unread bytes appear at the start (not shown in disassembly).

If the "filename" was not specified (but there was at least one other header), sub_403A38() is called as a method of the current object and is passed a 260-byte destination buffer along with the boundary string and the value of the "Content-Length" header.

```
.text:00403733                lea    eax, [ebp+szFileName] ; char[260]
.text:00403739                push   eax                ; Str
.text:0040373A                call   _strlen
.text:0040373F                add     esp, 4
.text:00403742                test   eax, eax
.text:00403744                jbe    loc_403830        ; filename not specified?
...
.text:00403830 loc_403830:                ; CODE XREF: sub_4032B9+48B
.text:00403830                lea    eax, [ebp+szDestBuffer] ; char[260]
.text:00403836                push   eax                ; lpDestBuffer
.text:00403837                lea    ecx, [ebp+szBoundary] ; char[260]
.text:0040383D                push   ecx                ; lpszBoundary
.text:0040383E                push   0                ; lpFileName
.text:00403840                lea    edx, [ebp+contentLength_]
.text:00403846                push   edx                ; int
.text:00403847                lea    eax, [ebp+bytesRead]
.text:0040384D                push   eax                ; lpBytesRead
.text:0040384E                mov     ecx, [ebp+THIS]
.text:00403854                call   sub_403A38
```

In sub_403A38(), after some initial setup (not shown in disassembly), if a destination buffer pointer was passed in (and in this case it was), then the first byte is set to zero.

```
.text:00403A56                cmp     [ebp+lpDestBuffer], 0
.text:00403A5A                jz     short loc_403A64 ; destination buffer not supplied?
.text:00403A5C                mov     eax, [ebp+lpDestBuffer]
.text:00403A5F                mov     byte ptr [eax], 0 ; set destination to empty string
.text:00403A62                jmp     short loc_403A90
```

A loop is entered that begins by filling up the temporary buffer (until the buffer is full or the number of bytes read equals the "Content-Length" header).

```
.text:00403AA6 @loop_read_data:                ; CODE XREF: sub_403A38:loc_403CBF
.text:00403AA6     mov     eax, [ebp+THIS]
.text:00403AA9     mov     ecx, [ebp+THIS]
.text:00403AAC     mov     edx, [eax+TempBuffer.bufferSize]
.text:00403AAF     sub     edx, [ecx+TempBuffer.bufferOffset]
.text:00403AB2     mov     [ebp+Count], edx ; unread bytes in buffer
.text:00403AB5     mov     eax, [ebp+lpContentLength]
.text:00403AB8     mov     ecx, [ebp+lpBytesRead]
.text:00403ABB     mov     edx, [eax]
.text:00403ABD     sub     edx, [ecx] ; number of bytes remaining in request
.text:00403ABF     cmp     [ebp+Count], edx
...
... read data into buffer
...
```

The supplied boundary string is searched for in the buffer by sub_402A22() (not shown in disassembly).

```
.text:00403B16 loc_403B16:                    ; CODE XREF: sub_403A38+9D
.text:00403B16     mov     [ebp+var_10], 0
.text:00403B1D     push   1 ; int
.text:00403B1F     mov     ecx, [ebp+lenBoundary]
.text:00403B22     push   ecx ; lenNeedle
.text:00403B23     mov     edx, [ebp+lpszBoundary]
.text:00403B26     push   edx ; lpszNeedle
.text:00403B27     mov     eax, [ebp+THIS]
.text:00403B2A     mov     ecx, [eax+TempBuffer.bufferOffset]
.text:00403B2D     push   ecx ; lenHaystack
.text:00403B2E     mov     edx, [ebp+THIS]
.text:00403B31     mov     eax, [edx+TempBuffer.lpBuffer]
.text:00403B34     push   eax ; lpszHaystack
.text:00403B35     call   sub_402A22 ; search for boundary in buffer
.text:00403B3A     add     esp, 14h
.text:00403B3D     mov     [ebp+boundaryLocation], eax
.text:00403B40     cmp     [ebp+boundaryLocation], 0
.text:00403B44     jz     short loc_403BC1 ; not found?
```

Regardless whether the boundary was found, data is copied into the destination buffer if it was supplied. If the boundary was not found, then all the unread data in the temporary buffer is copied. If the boundary was found, only that amount of data is copied. However, there is no verification that the size of the data copied is less than the size of the destination buffer and hence a stack-based buffer overflow may occur.

```
.text:00403C0B loc_403C0B:                    ; CODE XREF: sub_403A38:loc_403BFD
.text:00403C0B     cmp     [ebp+lpDestBuffer], 0
.text:00403C0F     jnz    short loc_403C34 ; have destination buffer?
...
.text:00403C34 loc_403C34:                    ; CODE XREF: sub_403A38+1D7
.text:00403C34     mov     eax, [ebp+Size]
.text:00403C37     push   eax ; Size
.text:00403C38     mov     ecx, [ebp+THIS]
.text:00403C3B     mov     edx, [ecx+TempBuffer.lpBuffer]
.text:00403C3E     push   edx ; Src
.text:00403C3F     mov     eax, [ebp+lpDestBuffer]
.text:00403C42     add     eax, [ebp+bufferOffset]
.text:00403C45     push   eax ; Dst
.text:00403C46     call   _memcpy_0
.text:00403C4B     add     esp, 0Ch
.text:00403C4E     mov     ecx, [ebp+bufferOffset]
.text:00403C51     add     ecx, [ebp+Size]
.text:00403C54     mov     [ebp+bufferOffset], ecx
```

Exploitation:

=====

Exploitation can be achieved by sending a POST request containing "multipart/form-data" that does not specify a filename and is longer than 260 characters. The return address can be overwritten, but due to continuation of execution problems, execution has been proven by overwriting a structured exception handler.

A large number of CGI executables contain the vulnerable code or call it via CGICommon.DLL and are therefore all vulnerable. Some of the CGI executables require an authenticated request to reach the vulnerable code, but most can be exploited without authentication.

Secunia Research has developed both a PoC and working exploit for the vulnerability. These are available to customers on Secunia Proof of Concept and Exploit Code Services.

Characteristics:

=====

Detection:

Look for an HTTP POST request to an OfficeScan CGI script with "multipart/form-data" where at least one field does not specify a "filename" and contains more than 260 characters of data. Note that due to the large number of CGI executables they are all potentially vulnerable.

Verification:

Make an HTTP POST request to the "cgiRqOPP.exe" CGI executable with "multipart/form-data" containing a field that specifies the name and is followed by 10000 "A" characters after the "\r\n\r\n". Ensure that the "Content-Length" header is correct. A vulnerable CGI process will terminate prematurely and return a "500 Internal Server Error".

Identification:

Please refer to the table below for confirmed vulnerable files. The default CGI folders in OfficeScan 7.3 with Apache are "%ProgramFiles%\Trend Micro\OfficeScan\PCCSRV\Web\CGI\" and "%ProgramFiles%\Trend Micro\OfficeScan\PCCSRV\Web_console\CGI\"

Notes:

- 1) Some files do not include version information.
- 2) This list is not exhaustive and more files may in fact be vulnerable.

Vulnerable files in "%ProgramFiles%\Trend Micro\OfficeScan\PCCSRV\Web\CGI":

Executable	Version
cgiCAV.exe	7.3.0.1020
cgiCheckIP.exe	no version info
cgiGetClient.exe	7.3.0.1343
cgiGetDomain.exe	7.3.0.1343
cgiLog.exe	7.3.0.1343
cgiMovClient.exe	7.3.0.1343
CGIOCommon.dll	no version info
cgiOnClientCfg.exe	7.3.0.1343
cgiOnClose.exe	7.3.0.1343
cgiOnMSCfg.exe	7.3.0.1343
cgiOnPSCfg.exe	7.3.0.1343
cgiOnRTCfg.exe	7.3.0.1343
cgiOnScan.exe	7.3.0.1343
cgiOnStart.exe	7.3.0.1343
cgiOnUpdate.exe	7.3.0.1343
cgiRqAlertMsg.exe	7.3.0.1343
cgiRqCfg.exe	7.3.0.1343
cgiRqINI.exe	7.3.0.1343
cgiRqOPP.exe	7.3.0.1343
cgiRqUnInst.exe	7.3.0.1343
jdkGetNotifyClient.exe	7.3.0.1343
jdkNotify.exe	7.3.0.1343

Vulnerable files in "%ProgramFiles%\Trend Micro\OfficeScan\PCOSRV\Web_console\CGI":

Executable	Version
cgiFindClient.exe	7.3.0.1343
CGIOCommon.dll	no version info
cgiRenDomain.exe	7.3.0.1343
cgiShowAntiBody.exe	no version info
cgiTurnLog.exe	no version info

Tested Versions:

=====

The vulnerability was analysed on Windows XP SP3 with Trend Micro OfficeScan Server 7.3 patch 4 build 1343 with critical patches 1355, 1362, and 1367 applied.

Fixed Versions:

=====

The vulnerability is patched in Trend Micro OfficeScan Server 7.3 patch 4 by Critical Patch Build 1374 by passing the size of the destination buffer to the vulnerable function and correctly limiting the size of the write.

References:

=====

SA32005:

<http://secunia.com/advisories/32005/>

CVE-2008-3862:

http://secunia.com/cve_reference/CVE-2008-3862/

Secunia Research:

http://secunia.com/secunia_research/2008-40/

Trend Micro:

http://www.trendmicro.com/ftp/documentation/readme/OSCE_7.3_CriticalPatch_B1374_readme.txt