

# [Full-disclosure] Microsoft Windows Vista/2003/XP/2000 file management security issues

---

*Source:* <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2007-03/msg00102.html>

---

- *From:* 3APA3A <[3APA3A@xxxxxxxxxxxxxxxxxxxx](mailto:3APA3A@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 8 Mar 2007 22:58:37 +0300
- 

This is an article I promised to publish after Windows ReadDirectoryChangesW (CVE-2007-0843) [1] issue. It should explain why you must never place secure data inside insecure directory.

Title: Microsoft Windows Vista/2003/XP/2000 file management security issues

Author: 3APA3A, <http://securityvulns.com/>

Vendor: Microsoft (and potentially another vendors)

Products: Microsoft Windows Vista/2003/XP/2000, Microsoft resource kit for Windows 2000 and different utilities.

Access Vector: Local

Type: multiple/complex (weak design, insecure file operations, etc)

Original advisory: <http://securityvulns.com/advisories/winfiles.asp>

Securityvulns.com news: <http://security.nnov.ru/news/Microsoft/Windows/files.html>

## 0. Intro

This article contains a set of attack scenarios to demonstrate security weakness in few very common Windows management practices. Neither of the problem explained is critical, yet combined together they should force you to review your security practices. I can't even say "vulnerabilities" because there is no something you can call "vulnerability". It's just something you believe is secure and it's not.

### 1.1 Problem: inability to create secured file / folder in public one.

Attack: folder hijack attack

First, it's simply impossible with standard Windows interface to create something secured in insecure folder.

#### Scenario 1.1:

Bob wishes to create "Bob private data" folder in "Public" folder to place few private files. "Public" has at least "Write" permissions for

"User" group. Bob:

- I Creates "Bob private data" folder
- II Sets permission for folder to only allow access to folder himself
- III Copies private files into folder

Alice wants to get access to folder Bob created. She

- Ia Immediately after folder is created, deletes "Bob private data" folder and creates "Bob private data" folder again (or simply takes ownership under "Bob private data" folder if permissions allow). It makes Alice folder owner.
- Iia Immediately after Bob sets permissions, she grants herself full control under folder. She can do it as a folder owner.
- IIIa Reads Bob's private files, because files permissions are inherited from folder

Alice can use "Spydir" (<http://securityvulns.com/soft/>) tool to monitor files access and automate this process. As you can see, [1] elevates this problem significantly.

This is not new attack. Unix has "umask" command to protect administrators and users. Currently, Windows has nothing similar.

CreateFile() API supports setting file ACL on file creation (just like open() allows to set mode on POSIX systems). ACL can be securely set only on newly created files. This raises a problem of secure file creation.

1.2 Problem: Inability to lock / securely change permissions of already created file

Attack: pre-open file/directory attack.

There are few classes of insecure file creation attack (attempt to open existing file), exploitable under Unix with hardlinks or symlinks. It's believed Windows is not vulnerable to this attacks because

- I. There is no symlinks under Windows. Symlink attacks are not possible.
- II. Security information in NTFS is not stored as a part of directory entry, it's a part of file data. Hard link attacks are not possible.
- III. File locks in Windows are mandatory. It means, if one application locks the file, another application can not open this file, if user doesn't have backup privileges. It mitigate different file-based attacks.

There is at least one scenario, attacker can succeed without symbolic link: to steal data written to file created without check for file existence regardless of file locks and permissions.

Attack description: if attacker can predict filename to be written, he can create file, open it and share this file for all types of access. Because locking and permissions are only checked on file open, attacker retain access to the file even if it's locked and it's permissions are changed to deny file access to attacker.

Exploit (or useful tool): <http://securityvulns.com/files/spyfile.c>

Opens file, shares it for different types of access and logs changes, keeping the file open.

Compiled version is available from <http://securityvulns.com/soft/>

Scenario 1.2.1:

Bob is now aware about folder hijack attack. He use xcopy /O /U /S to synchronize his files to newly created folder. xcopy /O copies security information (ownership and permissions) before writing data to file.

Alice use "Spydir" to monitor newly created folders and files in Bob's directory. She use Spyfile to create spoofed files in target directory and waits for Bob to run xcopy. Now, she has full control under content of Bob's files despite the fact she has no permissions to access these files.

In a same way directory content may be monitored by pre-opening directory.

Scenario 1.2.2:

Enterprise directory structure is replicated every day to another user-writable location in order to allow users to recover suddenly deleted or modified files. xcopy or robocopy (from resource kit) is used for replication. Attacker can hijack content of newly created files in newly created folders.

Same problem may happen on archive extraction or backup restoration.

Vulnerable applications:

- xcopy (from all Windows versions),
- robocopy (Windows 2000 Resource Kit),
- different archivers
- backup restoration utilities

By default, xcopy warns user the file exists, unless /Y or /U key is specified. But

I. /Y is always specified for replication

II. /Y can be specified via COPYCMD environment variable. COPYCMD environment variable can be created in autoexec.bat file.

Different situations are possible, where autoexec.bat is writable by attacker, if:

- Default Windows 2000 permissions are used or applied with domain policy [2].
- One can try to re-create autoexec.bat using POSIX subsystem

III. Neither xcopy nor other utilities warn user on existing directory. Pre-open directory attack will always succeed.

As you can see, [1] again dramatically elevates this problem.

1.3 Problem: user can completely block access to the files

Attack: open file deletion

(including Windows file replication service DoS)

If files is deleted while it's open, it still present in file system under it's old name until close. Any operation on this file (including attributes requests) fails, regardless of application rights and permissions (including backup ones).

Exploit: use spyfile, delete file while it's spied. Now, without closing spyfile, attempt any operation on this file (e.g. try to find it's ownership).

Scenario 1.3.1

Now Bob found an copy application to securely copy files. It deletes old file before creating new one. But it fails if Alice tries to spy on Bob files, because attempt to delete file succeeds, but file still present and is unmanageable.

Scenario 1.3.2

Windows file replication service (FRS) is used to replicate data between 2 public DFS folders to distribute load. Folder has permissions:

Everyone: Add & read

Creator Owner: Full Control

Thouse, Alice has no permissions to delete files created by Bob.

Replicated folder is available as a share on 2 different servers: \\SERVER1\Share and \\SERVER2\Share. Bob is connected to \\SERVER1\Share.

Alice uses "Spydir" to monitor files creation by Bob. Every time Bob creates new file on \\SERVER1\Share, Alice use spyfile to create file with same name on \\SERVER2\Share. It effectively leads to FRS collision. While trying to resolve collision, FRS fails to delete file created by Alice and Bob file is deleted (original file is moved to special hidden folder only accessible by administrator).

Workaround: never try to use creator-owner based permissions in

replicated folders.

Again, [1] seriously escalates this problem.

## 2. Conclusion:

It's simply impossible to securely create something in public folder.

At least DoS conditions are always possible.

Developers should not consider mandatory file locking as a security feature.

Developers should care about secure file creation to store sensitive information. CREATE\_NEW should always be used and ACL should be set with lpSecurityAttributes of CreateFile. No attempt to open existing file should be made.

Never try to create secure folder in public one. If you are forced, disconnect all users before this operation.

Never use replication, archive extraction or backup restore to user-accessible folder.

Bob and Alice should finally marry.

## 3. Vendor:

All timelines are same with [1].

[1]. Microsoft Windows ReadDirectoryChangesW information leak (CVE-2007-0843)

<http://security.nnov.ru/news/Microsoft/Windows/ReadDirector.html>

[2]. Windows 2000 system partition weak default permissions

<http://securityvulns.ru/news2205.html>

--

<http://securityvulns.com/>

^\_^

{ , . } \

+---oQQo-->{ ^ }<-----+ \

| ZARAZA U 3APA3A } You know my name – look up my number (The Beatles)

+-----o66o---+ /

|/

---

Full-Disclosure – We believe in it.

Charter: <http://lists.grok.org.uk/full-disclosure-charter.html>

Hosted and sponsored by Secunia – <http://secunia.com/>