

Symlink Races and how to deal with them

SambaXP 2022

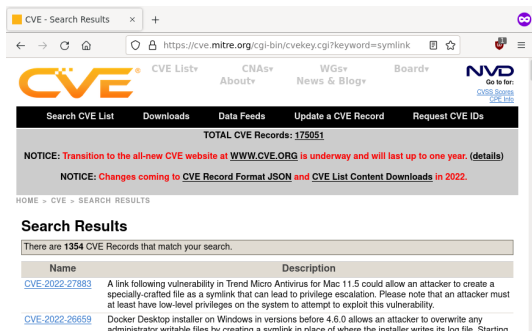
Volker Lendecke

Samba Team / SerNet

2022-06-01

Symlink race – Why care?

- ▶ Search for “symlink” in Common Vulnerabilities and Exposures:



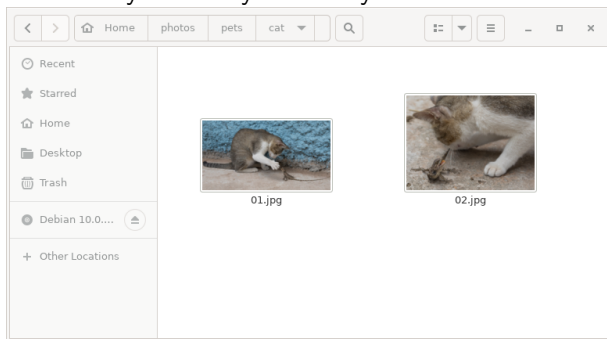
The screenshot shows a web browser window with the URL <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=symlink>. The page header includes the CVE logo and navigation links for CVE List, CNAs, WG's, Board, and NVD. Below the header is a navigation bar with links for Search CVE List, Downloads, Data Feeds, Update a CVE Record, and Request CVE IDs. A black banner displays "TOTAL CVE Records: 175051". Two notices are present: one about the transition to the new CVE website (WWW.CVE.ORG) and another about changes to the CVE Record Format JSON and CVE List Content Downloads in 2022. The search results section shows "There are 1354 CVE Records that match your search." and a table with two entries:

Name	Description
CVE-2022-27883	A link following vulnerability in Trend Micro Antivirus for Mac 11.5 could allow an attacker to create a specially-crafted file as a symlink that can lead to privilege escalation. Please note that an attacker must at least have low-level privileges on the system to attempt to exploit this vulnerability.
CVE-2022-26659	Docker Desktop installer on Windows in versions before 4.6.0 allows an attacker to overwrite any administrator writable files by creating a symlink in place of where the installer writes its log file. Starting

- ▶ Ok, there’s something going on – but what??

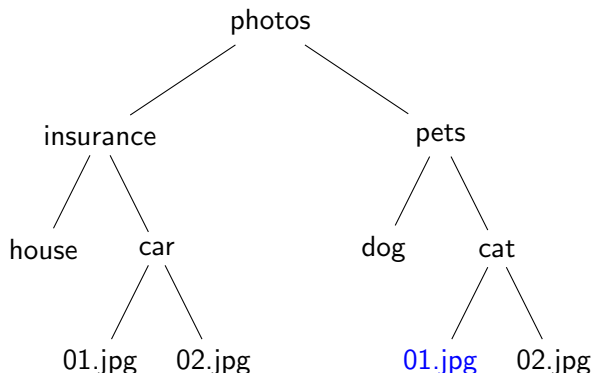
Publish cat photos to my blog

► Browse my directory hierarchy



- Copy file path `/photos/pets/cat/01.jpg` into Browser
- Press upload

My directory hierarchy

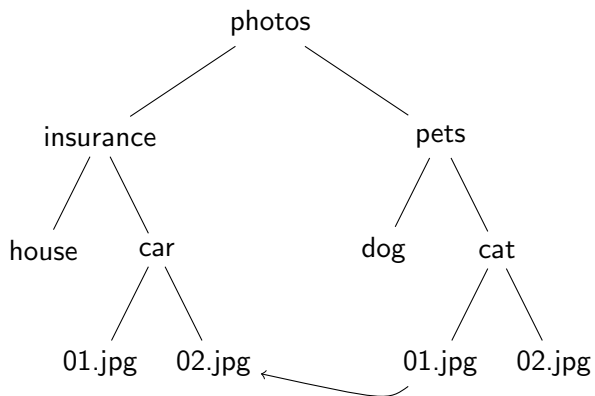


- ▶ Cute cat under `/photos/pets/cat/01.jpg`

Symlink Race

- ▶ Publish cat photo
 - ▶ Browse my directory hierarchy
 - ▶ Copy file path `/photos/pets/cat/01.jpg` into Browser
 - ▶ A few seconds for an **ATTACK**
 - ▶ Press upload
- ▶ The attacker replaces “01.jpg” with a symbolic link to the latest communication with my insurance
- ▶ In `-sf /photos/insurance/car/02.jpg /photos/pets/cat/01.jpg`

My new directory hierarchy



What gets published?

- ▶ /photos/pets/cat/01.jpg gets uploaded, but... oops



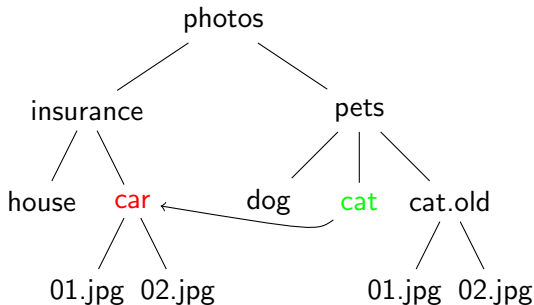
- ▶ This could have been your Passwords.DOCX

O_NOFOLLOW

- ▶ Posix/Linux can protect against this kind of attack
- ▶ Opening a file in Posix via C function `open()`
- ▶ Adding the flag `O_NOFOLLOW` to `open()` ensures “01.jpg” is not a symlink:
 - ▶ If the trailing component (i.e., `basename`) of `pathname` is a symbolic link, then the `open` fails, with the error `ELOOP`
- ▶ When using `O_NOFOLLOW`, the upload function will get an error
- ▶ However:
 - ▶ Symbolic links in earlier components of the `pathname` will still be followed.

My latest directory hierarchy

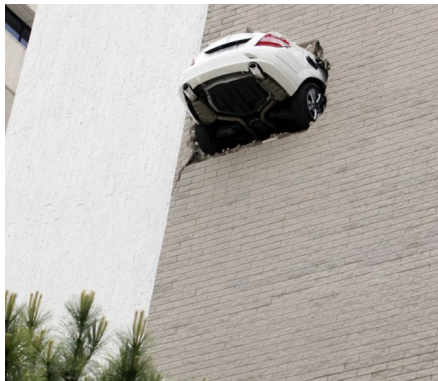
- ▶ Another **attack**



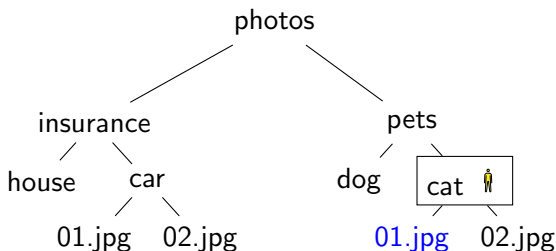
- ▶ `/photos/pets/cat` → `/photos/insurance/car`
- ▶ `/photos/pets/cat/01.jpg` → `/photos/insurance/car/01.jpg`

What gets published?

- ▶ /photos/pets/cat/01.jpg gets uploaded, but... ouch!



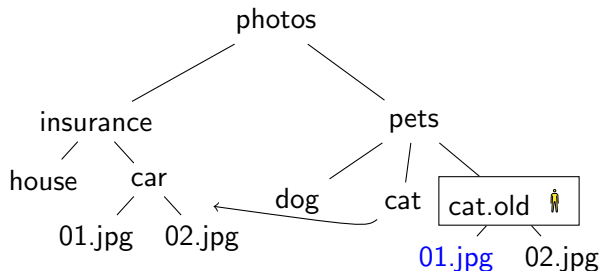
Change into /photos/pets/cat



► Four expensive steps

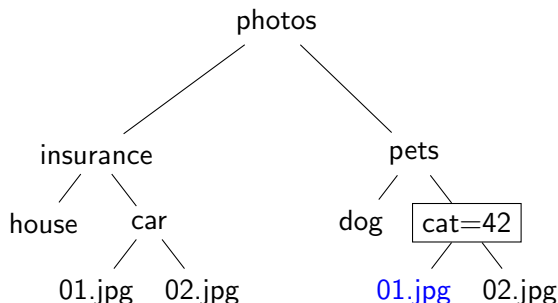
- `chdir /photos/pets/cat`
- check I am really in `/photos/pets/cat`
- `open("01.jpg", O_NOFOLLOW)`
- `chdir /`

Attack mitigated



- ▶ Attacker moves me to cat.old
- ▶ `open("01.jpg", O_NOFOLLOW)` references the correct file
- ▶ Opening files becomes expensive due to the four steps

Openat-Call



- ▶ Four expensive steps done just once
- ▶ Hold and cache reference to /photos/pets/cat (=42)
- ▶ Cheap openat(42, "01.jpg", O_NOFOLLOW)

Symlink races from 30,000ft

- ▶ Paths in Posix are prone to input validation problems
- ▶ Symlink races are a Time-Of-Check Time-Of-Use (TOCTOU) problem
- ▶ Sanitizing paths is possible, but tedious
⇒ Nobody does it
- ▶ Current status in Samba with 4.16:
 - ▶ Many places in Samba chose the `chdir/check/open/chdir` way
 - ▶ Work is ongoing to pass directory handles (the “42”) everywhere
- ▶ Work ongoing in Linux to make sanitizing easier:
 - ▶ `opent2(RESOLVE_NO_SYMLINKS)`
 - ▶ Mount option disallowing symlinks?

Samba 4.16

- ▶ OEMs complain to SerNet that Samba 4.15 is slower than 4.12
 - ▶ Benchmarks like specsfs swbuild do a lot of stat-opens
 - ▶ Just ask for file metadata like timestamps or filesize
- ▶ Number of syscalls is a problem:
- ▶ `strace -p <smbd-pid> of smbclient -c "open a/b/c/d/e/f/g/x.txt"`:
 - ▶ v4-12-strace: 30 lines
 - ▶ v4-16-strace: 55 lines
 - ▶ v4-17-strace: 43 lines
- ▶ v4-17 removes one `parent_pathref()` from `create_file_unixpath()`
 - ▶ `filename_convert_dirfsp()` provides the directory fsp now

Deleting a directory

- ▶ Three steps
 1. Open file with DELETE access permissions
 - ▶ Check parent access for DELETE_CHILD
 2. Set DELETE_ON_CLOSE disposition
 - ▶ Check for empty directory
 3. Close
 - ▶ `parent_pathref()` to prepare for `unlinkat()`
- ▶ In every step there is one or more `non_widelink_open()`

Next steps

- ▶ Replace `filename_convert()` with `filename_convert_dirfsp()`
- ▶ Pass `dirfsp` through as many callers as possible
- ▶ Alternative: Store `dirfsp` on every `fsp`
 - ▶ Required for file deletion
 - ▶ `dirfsp` is rather heavy-weight (344 bytes for `files_struct` alone)
- ▶ Eliminate calls `openat_pathref_fsp`
 - ▶ `synthetic_pathref()`, `parent_pathref()` etc
- ▶ Create a `conn→root_fsp` to avoid calls to `chdir()`

How to speed up filename_convert()?

- ▶ test_chdir: Do what we do today
 - ▶ Get current working directory
 - ▶ Verify full pathname with realpath()
 - ▶ chdir() into target directory
 - ▶ Verify “.” inside current working directory
 - ▶ open(“filename”, O_NOFOLLOW);
 - ▶ chdir back
- ▶ test_openat:
 - ▶ next = openat(dirfd, “single component”, O_PATH);
 - ▶ close(dirfd);
 - ▶ dirfd = next;
 - ▶ This is the path to proper SMB2 server-side symlink handling
- ▶ test_openat2: Just use RESOLVE_NO_SYMLINKS

Thanks for your attention

```
vl@samba.org / vl@sernet.de  
https://www.sernet.de/  
https://www.samba.org/
```