

CYBER
THREAT
ANALYSIS

Recorded Future®

By Insikt Group®

May 26, 2022



VULNERABILITY SPOTLIGHT: Dirty Pipe

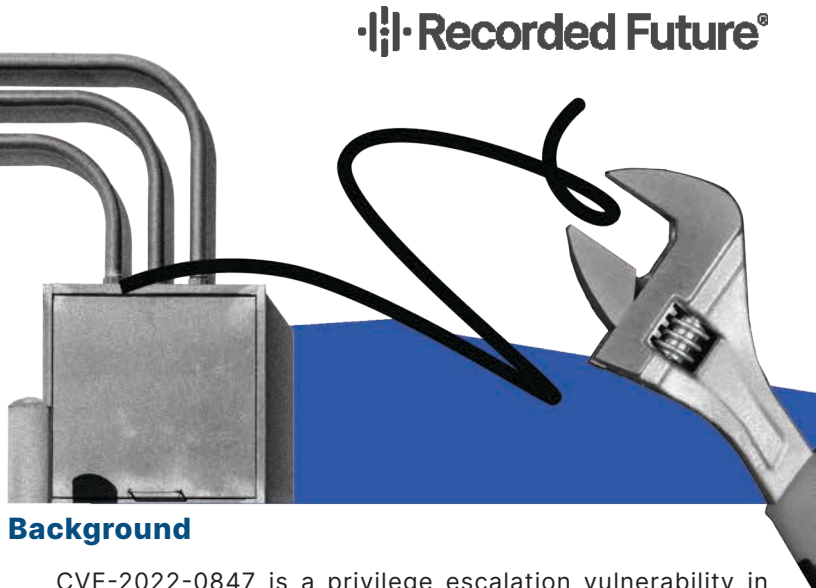
This report provides an overview, technical analysis, and mitigations for CVE-2022-0847. Sources include the Recorded Future® Platform, GitHub, and open-source reporting. The intended audience for this report is defenders and analysts who are interested in how CVE-2022-0847 exploits work, as well as current mitigations that can be employed.

Executive Summary

CVE-2022-0847 (Dirty Pipe) is a Linux kernel vulnerability that was disclosed in early March 2022. The vulnerability was introduced in Linux kernel version 5.8 and allows for local privilege escalation via arbitrary file overwrites. An example proof-of-concept (POC) exploit was released with the disclosure, and since then several other POCs have been published on GitHub. The public exploits are reliable and only require a small number of prerequisites to work, such as having read permissions to a targeted file. Given the nature of this vulnerability, there are many different files that can be targeted for privilege escalation; therefore, this report highlights the techniques used by existing POC exploits. CVE-2022-0847 was patched in Linux kernel versions 5.16.11, 5.15.25, and 5.10.102, and all major Linux-based distributions have incorporated patches into their package repositories. Organizations should apply the recommended patches as soon as possible.

Key Observations

- CVE-2022-0847 existed in the wild for roughly 2 years, although there is no evidence that it was exploited prior to its public disclosure.
- Multiple POC exploits are publicly available, making this vulnerability easy to exploit and accessible to unsophisticated attackers.
- Exploits for CVE-2022-0847 are reliable and allow an attacker to gain root access when run on a vulnerable system. The root access enables the threat actor to perform administrative tasks such as reading sensitive files, installing malicious software, impersonating users, and potentially moving laterally throughout the network.
- The only mitigation for CVE-2022-0847 is to apply security patches, which are available for all major Linux distributions.
- Recorded Future has observed over 90 underground forum references to CVE-2022-0847 since it was disclosed, illustrating a general interest and potential intent to exploit the vulnerability in future campaigns.



Background

CVE-2022-0847 is a privilege escalation vulnerability in the Linux kernel that allows arbitrary files to be overwritten if the attacker has read access to the file. The vulnerability was introduced into the Linux kernel in version 5.8 and existed for roughly 2 years before being discovered and patched. It was discovered by Max Kellermann, who gave it the nickname “Dirty Pipe” due to its similarities with CVE-2016-5195 (aka “Dirty Cow”). Kellermann identified the vulnerability on February 19, 2022, and initiated a coordinated vulnerability disclosure the following day by submitting a bug report, POC exploit, and patch to the Linux kernel security team. Once patches were in place, the vulnerability was publicly disclosed on March 7, 2022. At this time, there is no evidence that CVE-2022-0847 was exploited in the wild prior to its disclosure.

The bug itself is rated high (7.8) on the CVSS 3.0 scale. Given the widespread nature of Linux-based operating systems (OS), the vulnerability affects many devices other than just desktops and servers running Linux-based OSES. Vulnerable devices include a wide range of internet of things (IoT) devices, routers, and Android tablets and phones. It is not possible to remotely exploit the vulnerability; however, it could be chained with a remote code execution (RCE) vulnerability to be used without local access. Additionally, not all devices will be vulnerable as Linux kernel versions prior to 5.8 are not affected, as well as any kernel versions that have been updated to the patched versions.

In a search of underground and dark web sources, we identified numerous discussions pertaining to CVE-2022-0847. While it is uncommon for threat actors to publicly disclose their intentions of targeting specific organizations using a particular CVE, a general interest shared by dark web forum members indicates threat actors’ intent to use CVE-2022-0847 in malicious campaigns. We identified 120 references to CVE-2022-0847 across multiple dark web forums over the past 2 months, as shown in Figure 1 below.

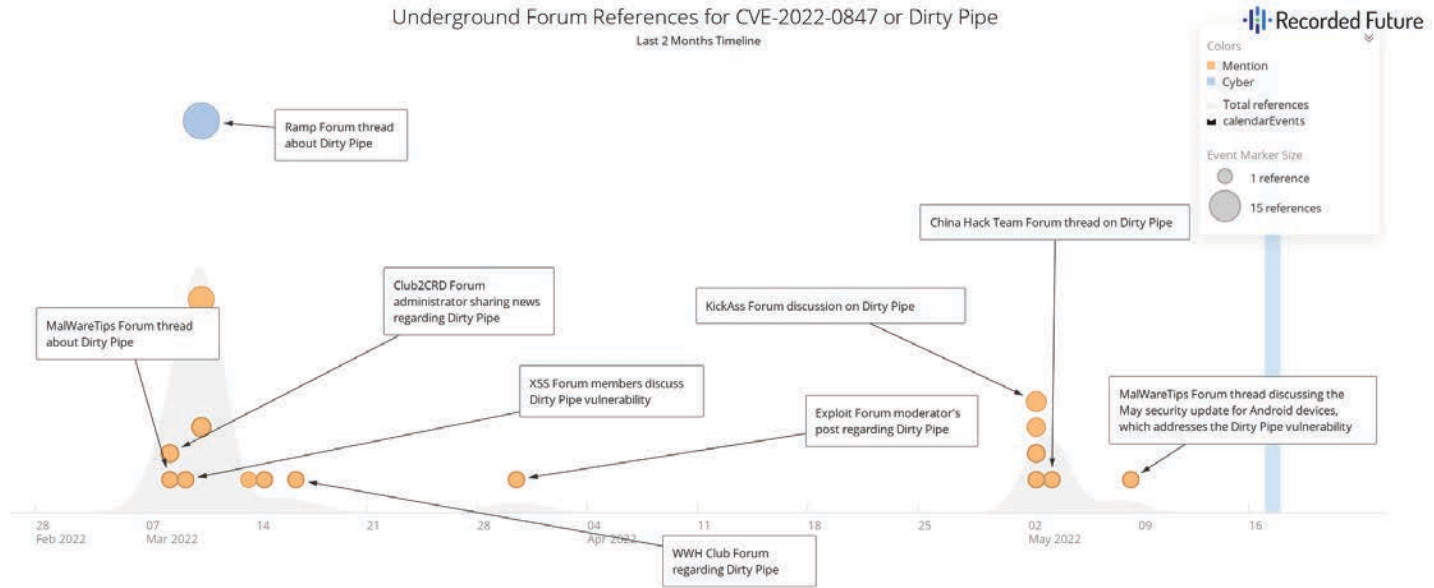


Figure 1: References for CVE-2022-0847 or Dirty Pipe on dark web forums (Source: Recorded Future)

```
$ ip addr show | grep inet
```

Figure 2: Example pipe command (Source: Recorded Future)

Technical Analysis

The Dirty Pipe vulnerability relies on the way memory is managed in the Linux kernel. When it comes to memory management, the smallest unit of memory managed by the kernel is a page, which is typically 4 KB. When memory is needed by a process, pages are allocated by the kernel based on the size of the memory requested. For example, when a process reads a file from disk, the kernel performs an I/O operation to read the contents of the file into one or more pages (depending on the file size) and stores them in a subsystem known as the page cache. From there, the data can be copied into a process's memory for use. As the name implies, pages remain in the page cache until they are needed again or the kernel reclaims their memory; this efficiency allows the kernel to potentially avoid repeating expensive disk I/O operations. In addition to caching data that is read, the page cache also stores data that is ready to be written to a file. When a page has been updated in memory but the change is not reflected in the underlying physical media, it is said to be "dirty"; hence the name Dirty Pipe.

In Linux, a pipe is used for unidirectional interprocess communication. One common use case for a pipe is to pass output from one command to another on the command line using the | symbol. For example, the following command takes the output from the ip command and provides it as input to the grep command so the lines containing "inet" can be filtered out. The pipe symbol essentially wires the ip command's stdout output stream to the grep command's stdin input stream.

Linux supports named pipes, also known as First In First Out (FIFO). The only difference between named pipes and the aforementioned "unnamed pipe" is how they are created. A named pipe is created using the [mkfifo](#) command and exists on the filesystem as a file that can be written to or read from. As with all pipes, data is read from the pipe in the same order that it was written.

In the kernel, a pipe is represented by the [pipe_inode_info](#) structure. This structure stores pipe data in a circular array of [pipe_buffer](#) structures. Among other data, the pipe_buffer structure stores a pointer to a page that contains the buffered data as well as flags to describe the buffer's characteristics.

The Vulnerability

The initial bug that preceded CVE-2022-0847 was introduced into the Linux kernel in 2016 with commit [241699cd7](#), which added 2 functions that create page_buffer structs but do not initialize their flags. At the time, this was not an issue as it was not exploitable; however, in mid-2020 commit [f6dd97558](#) introduced a new flag for pipe buffers known as PIPE_BUF_FLAG_CAN_MERGE that is used to indicate when a page is allowed to be written back to its original source file. Combining this new flag with the original bug from 2016, a pipe created in a specific manner could now write data back to a page located in the page cache, regardless of the permissions on the original file. To better understand the vulnerability, we can analyze Kellermann's original POC exploit code. Figure 3 shows an abbreviated version of the main function for the POC. It begins by opening the target file in read-only mode.

```

/* open the input file and validate the specified offset */
const int fd = open(path, O_RDONLY); // yes, read-
only! :-)

[...]

prepare_pipe(p);

/* splice one byte from before the specified offset
into the pipe; this will add a reference to the page
cache, but since copy_page_to_iter_pipe() does not
initialize the "flags", PIPE_BUF_FLAG_CAN_MERGE is still set */
--offset;
ssize_t nbytes = splice(fd, &offset, p[1], NULL,
1, 0);

[...]

/* the following write will not create a new pipe_
buffer, but will instead write into the page cache, because
of the PIPE_BUF_FLAG_CAN_MERGE flag */
nbytes = write(p[1], data, data_size);

```

Figure 3: Abbreviated main function for Kellermann's POC exploit

Next, the `prepare_pipe` function (shown in detail in Figure 4 below) is called. This function sets the `PIPE_BUF_FLAG_CAN_MERGE` flag on each of the pipe's `pipe_buffer` structures by completely filling the pipe with data. Then, it drains the pipe to empty out the data while leaving the flags intact.

```

/**
 * Create a pipe where all "bufs" on the pipe_inode_info
ring have the
 * PIPE_BUF_FLAG_CAN_MERGE flag set.
 */
static void prepare_pipe(int p[2])
{
    if (pipe(p)) abort();

    const unsigned pipe_size = fcntl(p[1], F_GETPIPE_
SZ);
    static char buffer[4096];

    /* fill the pipe completely; each pipe_buffer will
now have
    the PIPE_BUF_FLAG_CAN_MERGE flag */
    for (unsigned r = pipe_size; r > 0;) {
        unsigned n = r > sizeof(buffer) ?
sizeof(buffer) : r;
        write(p[1], buffer, n);
        r -= n;
    }

    /* drain the pipe, freeing all pipe_buffer
instances (but
    leaving the flags initialized) */
    for (unsigned r = pipe_size; r > 0;) {
        unsigned n = r > sizeof(buffer) ?
sizeof(buffer) : r;
        read(p[0], buffer, n);
        r -= n;
    }

    /* the pipe is now empty, and if somebody adds a
new
    pipe_buffer without initializing its "flags",
the buffer
    will be mergeable */
}

```

Figure 4: POC exploit to set the `PIPE_BUF_FLAG_CAN_MERGE` flag on a pipe's buffer
(Source: Max Kellermann)

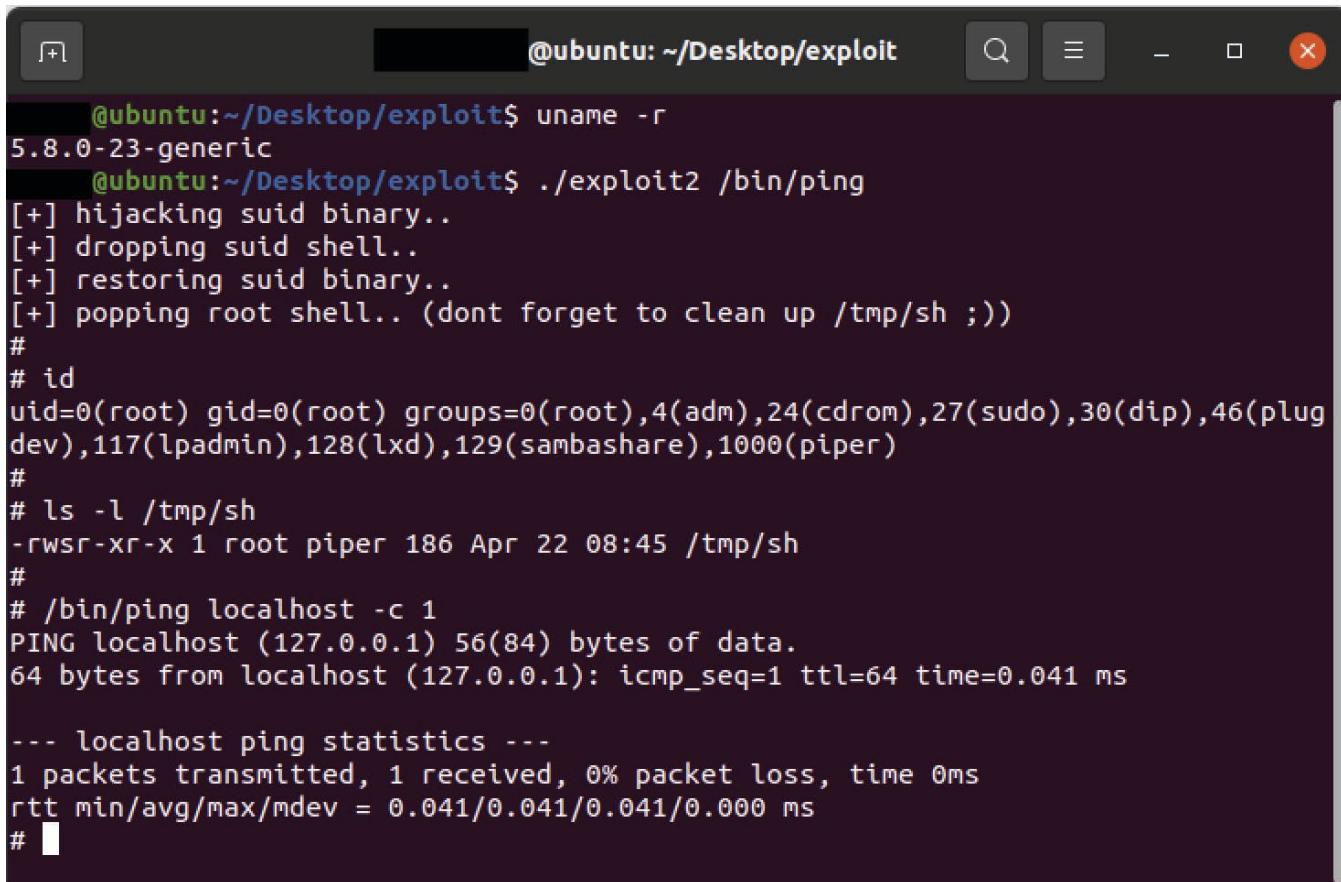
After preparing the pipe, a `splice` system call is made to write 1 byte from the file into the pipe. This call is required because it allows the file to be read into a page stored in the page cache and then copies a reference to the page cache into the pipe's buffer. The underlying function that performs the copy fails to properly initialize the flags, thus leaving the `PIPE_BUF_FLAG_CAN_MERGE` flag set. Subsequent writes to the pipe are then written back into the page cache, allowing the file contents to be overwritten even though it was opened in a read-only manner.

When exploited, this vulnerability permits writing to files that would normally not be writable even by a root user, such as `btrfs` snapshots, read-only mounts (for example, CD-ROMs), and immutable files. This is possible because of the non-conventional way the writes are being performed by the kernel via the page cache, and pipes do not check for permissions. As a result, attackers can exploit Dirty Pipe to gain root access in non-traditional environments that are mounted in read-only mode, such as mobile and IoT device file systems.

The only requirements for successful exploitation of the vulnerability are:

- The attacker must have read-only data to the file they wish to modify. This is because the file must be read into a page so that it can later be spliced into the pipe.
- The data written into the file cannot start on a page boundary, otherwise a reference to the page cache would not be copied into the pipe's buffer.
- The data written cannot cross page boundaries because this would create a new page rather than overwriting the page in the page cache.
- The file cannot be resized because the pipe is unable to tell the page cache how much data was appended.

In practice, these limitations provide numerous avenues for threat actors to gain escalated privileges. Unprivileged users have read access permissions to a number of sensitive files on a file system for legitimate reasons, such as the `/etc/passwd` file and SUID binaries. Furthermore, the restrictions on page boundary writing and size are also easy to circumvent as the injected content does not need to be large for an exploit to be effective. In the following section, we highlight some of the techniques observed in existing publicly available POC exploits for Dirty Pipe.



```
@ubuntu: ~/Desktop/exploit
@ubuntu:~/Desktop/exploit$ uname -r
5.8.0-23-generic
@ubuntu:~/Desktop/exploit$ ./exploit2 /bin/ping
[+] hijacking suid binary..
[+] dropping suid shell..
[+] restoring suid binary..
[+] popping root shell.. (dont forget to clean up /tmp/sh ;)
#
# id
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),117(lpadmin),128(lxd),129(sambashare),1000(piper)
#
# ls -l /tmp/sh
-rwsr-xr-x 1 root piper 186 Apr 22 08:45 /tmp/sh
#
# /bin/ping localhost -c 1
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.041 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.041/0.041/0.041/0.000 ms
#
```

Figure 5: Running SUID binary POC, showing shell running with root privileges, created `/tmp/sh` file with SUID bit, and restored functionality of overwritten SUID binary (Source: Recorded Future)

POC Exploit Techniques

Several POC exploits have been released since the vulnerability was publicly disclosed. Each is based on Kellermann's original POC code, but they take different approaches in how they escalate privileges. While there are many possible ways to abuse CVE-2022-0847, we have observed 3 common techniques in POC exploits:

1. [Targeting](#) the `/etc/passwd` file to include a new user, mapped to uid 0 (that is, root), with a known password hash. This is a viable method because most systems will still check the `/etc/passwd` file when authenticating a user, despite modern systems placing their password hashes in `/etc/shadow`.
2. Overwriting a SUID binary, such as `ping`, with a custom executable instead. The purpose of a SUID binary is to allow non-privileged users to perform privileged actions, such as generating packets with the `ping` command. The SUID bit can be set via the `chmod` command to allow a binary to run with the permissions of the file's owner, which in most cases is root. Overwriting one of these files with a custom binary allows an attacker to effectively escalate their privileges to that of the original file's owner. In [this](#) example POC, the target binary is overwritten with a small ELF program that drops another ELF program to `/tmp/sh`, setting the SUID bit and the owner as root. The exploit then runs the newly created file `/tmp/sh`, which sets the UID and GID to 0 (root) and then executes `/bin/sh` to launch an interactive shell. Once in the shell, the attacker has root access to the system. The exploit restores the original content of the overwritten binary and advises the attacker to manually remove the temporary ELF file created as `/tmp/sh`.
3. In Kellermann's [original POC exploit](#), he left a comment showing how to use the exploit to add a new entry to the root user's `authorized_keys` file. The `authorized_keys` file contains a list of public keys that are able to log in as the user (in this case root) provided they have the corresponding private key. Adding a new entry to the `authorized_keys` file allows an attacker to SSH into the victim machine as the root user with their own private key, bypassing any password prompts and providing them with root access to the system. In practice this vector is unlikely to work, as an unprivileged user does not have read access to the root user's `authorized_keys` file by default.

Mitigations

The only viable mitigation for CVE-2022-0847 is to upgrade to a newer version of the Linux kernel. The vulnerability has been patched in kernel versions 5.16.11, 5.15.25, and 5.10.102, and updates are available in all major distributions via their respective package managers. Organizations should check the kernel version of their Linux systems and patch accordingly.

To determine the kernel version a Linux installation is using, the `uname -r` command can be run in a terminal. Additionally, a shell script is available [here](#) to automatically perform a version check against a specific system (or version of the kernel) to determine whether it is vulnerable.

Android and IoT devices are more complicated to patch due to their reliance on device manufacturers for firmware updates. Android users can check their kernel versions by going to Settings → About Phone → Android/Software Version → Kernel Version. Affected users are encouraged to reach out to their device manufacturer for update information. Similarly, due to the nature of IoT devices, it is impossible to provide generic advice for determining if, and what version of, the Linux kernel is being used. Therefore, the device manufacturer should be consulted to determine whether a particular IoT device is vulnerable and the availability of firmware updates.

Outlook

CVE-2022-0847 is an easy and reliable vulnerability that is exploitable on any of the unpatched affected kernel versions. Many POC exploits are publicly available and accessible to unsophisticated threat actors who may otherwise not possess the skill to create an exploit themselves. An attacker exploiting Dirty Pipe on an OS running a vulnerable kernel version will gain root privileges. The root privileges allow the attacker to perform actions on their objectives, such as reading sensitive files, installing malicious software, impersonating users, or potentially moving laterally throughout the network. Organizations running affected versions of the Linux kernel are highly encouraged to follow the steps outlined in the mitigations section and upgrade their systems as soon as possible.

About Insikt Group®

Insikt Group is Recorded Future's threat research division, comprising analysts and security researchers with deep government, law enforcement, military, and intelligence agency experience. Their mission is to produce intelligence on a range of cyber and geopolitical threats that reduces risk for clients, enables tangible outcomes, and prevents business disruption. Coverage areas include research on state-sponsored threat groups; financially-motivated threat actors on the darknet and criminal underground; newly emerging malware and attacker infrastructure; strategic geopolitics; and influence operations.

About Recorded Future®

Recorded Future is the world's largest intelligence company. Recorded Future's cloud-based Intelligence Platform provides the most complete coverage across adversaries, infrastructure, and targets. By combining persistent and pervasive automated data collection and analytics with human analysis, Recorded Future provides real-time visibility into the vast digital landscape and empowers clients to take proactive action to disrupt adversaries and keep their people, systems, and infrastructure safe. Headquartered in Boston with offices and employees around the world, Recorded Future works with more than 1,400 businesses and government organizations across more than 60 countries.

Learn more at recordedfuture.com and follow us on Twitter at @RecordedFuture.