Talos Vulnerability Report

TALOS-2021-1358

Garrett Metal Detectors iC Module CMA CLI getenv command directory traversal vulnerability

DECEMBER 20, 2021

CVE NUMBER

CVE-2021-21907

Summary

A directory traversal vulnerability exists in the CMA CLI getenv command functionality of Garrett Metal Detectors' iC Module CMA Version 5.0. A specially-crafted command line argument can lead to local file inclusion. An attacker can provide malicious input to trigger this vulnerability.

Tested Versions

Garrett Metal Detectors iC Module CMA Version 5.0

Product URLs

https://garrett.com/security/walk-through/accessories

CVSSv3 Score

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

CWE

CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Details

The Garrett iC Module provides network connectivity to either the Garrett PD 6500i or Garrett MZ 6100 models of walk-through metal detectors. This module enables a remote user to monitor statistics such as alarm and visitor counts in real time as well as make configuration changes to metal detectors.

The Garrett iC Module exposes an authenticated CLI over TCP port 6877. This interface is used by a secondary GUI client, "CMA Connect", to interact with the iC Module on behalf of the user. After a client successfully authenticates they may send plaintext commands to interact with the device. This CLI is how the remote software invokes the majority of its functionality when getting and setting various device configurations.

One of the commands exposed by this service allows an authenticated user to read what the application refers to as "environment variables". These "environment variables" are tracked as key/value pairs where the value is stored as the contents of a file named after the key. This command, `getenv [key]`, will result in a file being read and the contents returned to the user. This filename is crafted by concatenating the string "`/ltrx_user/env/`" with the user-supplied `key`.

For reference, an approximate decompilation of the `setEnv` handler function is included below. For brevity, functionality that was not relevant to the vulnerability (such as logging, error handling and remote client interaction) has been excluded.

```
void __cdecl getEnv(unsigned __int8 *key, unsigned __int8 *dest, uint8_t len)
{
  size_t v3; // r0
  unsigned __int8 envBuf[128]; // [sp+1Ch] [bp-190h] BYREF
  unsigned __int8 tmpBuf[128]; // [sp+9Ch] [bp-110h] BYREF
  unsigned __int8 filename[128]; // [sp+11Ch] [bp-90h] BYREF
  dirent *dir; // [sp+19Ch] [bp-10h]
  DIR *d; // [sp+1A0h] [bp-Ch]
  uint8_t i; // [sp+1A7h] [bp-5h]

  i = 1;
  if ( strlen((const char *)key) > 1 && *key )
  {
    strcpy((char *)filename, "/ltrx_user/env/");
    strcat((char *)filename, (const char *)key);
    if ( file_exists(filename) )
    {
      readfile(filename, envBuf);
      strcpy((char *)dest, (const char *)envBuf);
    }
    else
    {
      strcat((char *)dest, "variable not found\r\n");
    }
  } else {
    ...
   }
}
```

The `getEnv` function does not attempt to sanitize or otherwise validate the contents of the `key` parameter, allowing an authenticated attacker to supply directory traversal primitives and read semi-arbitrary files. However, attempting to read the majority of files will cause a `SEGFAULT` as the implementation of `readfile` does not limit the length of the file contents being copied into the fixed-size buffer provided by `getEnv`. As seen above, the contents of the supplied file are read in to `envBuf` which is 128 bytes long. If the contents of the file are longer than 128 bytes the buffer will overflow the `envBuf` buffer and begin corrupting the stack. Given how the function is structured, the function will execute as expected up until the entire stack frame is corrupted, requiring 396 bytes of data to corrupt the stack frame (`sizeof(envBuf[128])+sizeof(tmpBuf[128])+sizeof(filename[128])+sizeof(dirent *) + sizeof(DIR *) + sizeof(uint8_t) + align(3) = 396`). This restricts the maximum size of a disclosable file to no more than 396 bytes.

At this time, I have been unable to find a way to craft a file large enough to trigger this buffer overflow and gain code execution. If an attacker were able to place arbitrary files onto the filesystem, this vulnerability could be abused to execute arbitrary code.

Exploit Proof of Concept

```
getenv ../../../etc/version
```

Timeline

2021-08-17 - Vendor Disclosure
2021-11-10 - Talos granted disclosure extension
2021-12-13 - Vendor patched
2021-12-15 - Talos tested patch
2021-12-20 - Public Release

CREDIT

Discovered by Matt Wiseman of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2021-1357

NEXT REPORT TALOS-2021-1426