IOActive Security Advisory

Title	CIRCUTOR SGE PLC1000/PLC50
Severity	1 Critical, 10 High, 3 medium
Discovered by	Gabriel Gonzalez / Sergio Ruiz
Advisory Date	TBD

Affected Product

- CIRCUTOR – SGE-PLC1000/SGE-PLC50 v 0.9.2 / ServicePack 140411

Timeline

- 2025-03-14: Vulnerabilities identified, disclosure process begins.
- 2025-09-30: INCIBE gets ahold of the client to fix vulnerabilities.
- 2025-10-28: INCIBE coordinates disclosure: https://www.incibe.es/en/incibe-cert/notices/aviso-sci/multiple-vulnerabilities-circutor-products-0

Vulnerability List

Finding ID	Title	Total Risk	Effort to Fix
GRT_PLC-SGE100_01	Pre-Auth Memory Corruption in TACACSPLUS Library	Critical	Low
GRT_PLC-SGE100_02	Buffer Overflow in ShowDownload Function	High	Low
GRT_PLC-SGE100_03	Buffer Overflow on AddEvent Function	High	Low
GRT_PLC-SGE100_04	Buffer Overflow on ShowMeterDatabase Function	High	Low
GRT_PLC-SGE100_05	Command Injection and Buffer Overflow on GetDNS, CheckPing and TraceRoute Functions	High	Low
GRT_PLC-SGE100_06	Hardcoded authentication key	High	Low
GRT_PLC-SGE100_07	Several Buffers Overflow on ShowMeterPasswords Function	High	Low
GRT_PLC-SGE100_08	Several Buffers Overflow on showMeterReport	High	Low
GRT_PLC-SGE100_09	Several Buffers Overflow on ShowSupervisorParameters Function	High	Low
GRT_PLC-SGE100_10	[PLC] Buffer Overflow On SetUserPassword Function	High	Low
GRT_PLC-SGE100_11	[PLC] Command Injection on SetLan Function	High	Low
GRT_PLC-SGE100_12	Out-Of-Bounds Read on DownloadFile	Medium	Low
GRT_PLC-SGE100_14	Weak Authenticity Algorithm to Upgrade System and Hardcoded Password	Medium	Low

Detailed Findings

GRT_PLC-SGE100_01 - Pre-Auth Memory Corruption in TACACSPLUS Library

Host(s) / File(s)	tacacsplus
Category	CWE-122: Heap-based Buffer Overflow
CVSSv3	10 (Critical) - CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
CVE	CVE-2025-11778

Threat and Impact

IOActive found a remotely exploitable memory corruption in the read_packet() function of the TACACSPLUS implementation. The function reads a header from the network and allocates memory based on the information read from the socket. As can be seen on the highlighted code, later a integer overflow can be used to allocate a small buffer that will later be overflowed on the second call to socketread().

The decompiled code of the affected function:

```
int read packet()
{
[...]
  if ( !a1 )
    return 0;
  v3 = sockread(a1, *( DWORD *)(a1 + 16), v23, 12, 30);
  if ( v3 == 12 )
  {
    if ( (v23[0] & 0xF0) == 192 )
    {
      v2 = (int *)malloc((HIBYTE(v23[2]) | (v23[2] <<
24) | ((v23[2] & 0xFF0000u) >> 8) | ((v23[2] & 0xFF00)
<< 8)) + 12);
      memcpy(v2, v23, 0xCu);
      v4 = sockread(
             a1,
             *( DWORD *)(a1 + 16),
             v2 + 3,
             HIBYTE(v23[2]) | (v23[2] << 24) | ((v23[2]
& 0xFF0000u) >> 8) | ((v23[2] & 0xFF00) << 8),
             30);
[...]
}
```

Recommendations

Make sure there are no integer overflows and that the amount of data allocated will fit the data written.
Additional Information
During the research, it was found that the module seems to be based on the below open source library, so further devices might be affected by this very same issue: https://github.com/ArthurRichard/libtacplus-uClibc/blob/master/tac_packet.c

GRT_PLC-SGE100_02 - Buffer Overflow in ShowDownload Function

Host(s) / File(s)	index.cgi
Category	CWE-121: Stack-based Buffer Overflow
CVSSv3	8.9 (High) - CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:H
CVE	CVE-2025-11782

Threat and Impact

The code uses sprintf() to format a string that includes user-controlled input from GetParameter("meter") into the fixed-size buffer acStack_4c (64 bytes) without length checking. An attacker can provide an overly long value for the "meter" parameter that exceeds the 64-byte buffer size.

```
void ShowDownload(undefined4 param 1,undefined4 param 2)

{
  undefined4 uVar1;
  int iVar2;
  char acStack 4c [64];

  printf("<table style=\"width:100%%;\"><tr
align=\"center\"><td class=\"RSN\" style=\"height:30px\"
> %s</td></tr></table>\n"
         ,param 1);
  puts("<table style=\"width:100%;text-
align:center;\">");
  uVar1 = GetString(0x20b);
  printf("<tr><td class=\"RIV\">%s</td></tr>",uVar1);
  sprintf(acStack 4c,"Download(\'%d\')",param 2);
  uVar1 = GetString(0x11e);
  printf("<tr><td colspan=\"4\" style=\"text-
align:right;\"><input class=\"RBS\" type=\"%s\" value=\
"%s\" onclick=\"%s\"/></td></tr>"
         ,"button",uVar1,acStack 4c);
  iVar2 = GetParameter("meter");
  if (iVar2 != 0) {
    uVar1 = GetParameter("meter");

sprintf(acStack 4c,"Request(\'MeterMenu\',\'&meter=%s\')
",uVar1); /*uvar1 is user controlled by
getParameter("meter") and acStack 4c is fixed to 64. */
    uVar1 = GetString(0x93);
    printf("<tr><td colspan=\"4\" style=\"text-
align:right;\"><input class=\"RBS\" type=\"%s\" value
=\"%s\" onclick=\"%s\"/></td></tr>"
           ,"button",uVar1,acStack 4c);
  }
  puts("</table>");
  return;
}
```

Recommendations
1. Replace sprintf() with snprintf(). The snprintf() function allows you to specify the maximum number of bytes to write, preventing buffer overflows. 2. Input Validation and Sanitization

GRT_PLC-SGE100_03 - Buffer Overflow on AddEvent Function

Host(s) / File(s)	index.cgi
Category	CWE-121: Stack-based Buffer Overflow
CVSSv3	8.9 (High) - CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:H
CVE	CVE-2025-11783

Threat and Impact

Buffer overflow vulnerability in the AddEvent() function when copying user-controlled username input to a fixed-size buffer (48 bytes) without bounds checking. It can lead to a memory corruption leading to potential remote code execution.

```
LABEL 7:
    if ( SessionStatus )
      goto LABEL 13;
    failedLoginUsername = GetParameter("user");
    AddEvent(0x1Eu, failedLoginUsername, 0);

LABEL 13:
  successLoginUsername = GetParameter("user");
  AddEvent(0x1Cu, successLoginUsername, 0);

int   fastcall AddEvent(unsigned int eventId, const char
*eventMessage, const char *eventDetails)
{
  int loopCounter; // r3
  int eventIndex; // r4
  int tableOffset; // r6
  int eventPriority; // r4
  time t currentTime; // [sp+0h] [bp-24h] BYREF

  if ( eventId > 0x4E )
    return 0;
  LOBYTE(loopCounter) = 0;
  for ( eventIndex = 0; ; ++eventIndex )
  {
    loopCounter = (loopCounter + 1);
    tableOffset = 4 * eventIndex;
    if ( g eventDefinitionTable[4 * eventIndex] ==
eventId )
      break;
    if ( loopCounter == 79 )
      return 0;
  }
  time(&currentTime);
  memset(&g eventRecord, 0, 0x70u);
  eventPriority = g eventDefinitionTable[4 * eventIndex
+ 2];
  g eventRecord = currentTime;
  g eventPriorityLevel = eventPriority + 1;
  if ( eventPriority == 5 )
```

```
    g eventPriorityLevel = 10;
  g eventCategory = g eventDefinitionTable[tableOffset +
1];
  g eventType = eventId;
  if ( eventMessage )
    strcpy(g eventMessageBuffer,
eventMessage);/*eventMessage is the userName failed or
successed, g eventMessageBuffer is a fixed 48 length
string */
  if ( eventDetails )
    strcpy(g eventDetailsBuffer, eventDetails);
  return addEvent(eventPriority);
}
```

Recommendations

Validate all user input including lengths and other user controllable data before being used.

GRT_PLC-SGE100_04 - Buffer Overflow on ShowMeterDatabase Function

Host(s) / File(s)	index.cgi
Category	CWE-121: Stack-based Buffer Overflow
CVSSv3	8.9 (High) - CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:H
CVE	CVE-2025-11784

Threat and Impact

Buffer Overflow vulnerability due to unbounded user input being copied to a fixed-size buffer through sprintf(). The function GetParameter("meter") retrieves user input that is directly incorporated into a buffer without size validation. An attacker can supply an oversized input for the "meter" parameter, causing a buffer overflow that could:

- Overwrite adjacent memory
- Corrupt the stack
- Potentially lead to arbitrary code execution
- Cause denial of service through application crashes

```
void ShowMeterDatabase(void){
      char acStack 5c[64];
      int uVar4;
      int uVar3;
      uVar4 = getParatemer("meter");
      sprintf(acStack 5c,%s   -  ,uVar3,uVar4);
      /* acStack fixed to 64, uVar4 controlled by user */
}
```

Recommendations

Validate all user input including lengths and other user controllable data before being used.

GRT_PLC-SGE100_05 - Command Injection and Buffer Overflow on GetDNS, CheckPing and TraceRoute Functions

Host(s) / File(s)	index.cgi
Category	CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CVSSv3	8 (High) - CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H
CVSSv3	CVE-2025-11787

Threat and Impact

The highlighted line sprintf(command,"nslookup %s",uVar1); takes user input from GetParameter("dns") and directly inserts it into a command string that's executed by the system using popen(). An attacker can inject arbitrary commands by including shell metacharacters in the DNS parameter.

```
void GetDNS(void)

{
  undefined4 uVar1;
  FILE *  stream;
  char *pcVar2;
  char acStack 103c [2048];
  char acStack 83c [2048];
  char command [40];

  uVar1 = GetParameter("dns");
                    /* command injection and overflow */
  sprintf(command,"nslookup %s",uVar1);
  memset(acStack 103c,0,0x800);
  memset(acStack 83c,0,2048);
    stream = popen(command,"r");
  if (  stream != (FILE *)0x0) {
    while (pcVar2 = fgets(acStack 103c,0x800,  stream),
pcVar2 != (char *)0x0) {
      strncat(acStack 83c,acStack 103c,0x7ff);
    }
    pclose(  stream);
  }
  pcVar2 = strstr(acStack 83c,"Name:");
  if (pcVar2 == (char *)0x0) {
    ShowError(0x205);
    ShowNetTools();
  }
  else {
    ShowSuccess(0x204);
    ShowNetTools();
    uVar1 = GetParameter("dns");
    AddResultNetTools(0x207,uVar1,acStack 83c);
  }
```

```
  return;
}

int32 t CheckPing() {
void str;
sprintf(&str, "ping -w 4 %s", GetParameter("ping"));
void var 103c;
memset(&var 103c, 0, 0x800);
void s;
memset(&s, 0, 0x800);
int32 t stream = popen(&str, "r");

int32 t TraceRoute(){
void str;
sprintf(&str, "traceroute -I -m 20 -q 1 -w 1 %s",
GetParameter("traceroute"));
void var 103c;
memset(&var 103c, 0, 0x800);
void s;
memset(&s, 0, 0x800);
int32 t stream = popen(&str, "r");
```

Recommendations

- Input validation: Verify that the DNS parameter contains only valid hostname characters
- Command sanitization: Escape or remove dangerous shell characters
- Use safer alternatives: Consider DNS resolution libraries instead of shell commands

GRT_PLC-SGE100_06 - Hardcoded authentication key

Host(s) / File(s)	concentrator
Category	CWE-321: Use of Hard-coded Cryptographic Key
CVSSv3	8.8 (High) - CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
CVE	CVE-2025-11781

Threat and Impact

The affected firmware contains a hardcoded static authentication key . An attacker with local access to the device can extract this key (e.g., via firmware image analysis or memory dump) and create valid firmware upgrade packages. This bypasses all intended access controls and grants full administrative privileges.

```
}
    snprintf(
      v12,
      0xFFu,
      "openssl enc -des -d -base64 -in %sencrypt.enc -
out %sclau.sha1 -pass pass:<redacted>",
      "/tmp/updateDC/",
      "/tmp/updateDC/");
    if ( system(v12) )
    {
```

Recommendations

Use a Public Key Cryptography to authenticate upgrade packages.

GRT_PLC-SGE100_07 - Several Buffers Overflow on ShowMeterPasswords Function

Host(s) / File(s)	index.cgi
Category	CWE-121: Stack-based Buffer Overflow
CVSSv3	8.9 (High) - CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:H
CVE	CVE-2025-11785

Threat and Impact

Buffer Overflow vulnerability due to unbounded user input being copied to a fixed-size buffer through sprintf(). The function GetParameter("meter") retrieves user input that is directly incorporated into a buffer without size validation. An attacker can supply an oversized input for the "meter" parameter, causing a buffer overflow that could:

- Overwrite adjacent memory
- Corrupt the stack
- Potentially lead to arbitrary code execution
- Cause denial of service through application crashes

```
void ShowMeterPasswords(void)

{
  int iVar1;
  char *  s1;
  int iVar2;
  undefined4 uVar3;
  undefined4 uVar4;
  uint uVar5;
  int iVar6;
  int iVar7;
  char acStack 5c [64];
  uVar4 = GetParameter("meter");
  sprintf(acStack 5c,"%s   -  
%s",uVar3,uVar4);

  uVar3 = GetParameter("meter");

sprintf(acStack 5c,"Request(\'MeterMenu\',\'&meter=%s\')
",uVar3);
}
```

Recommendations
Validate all user input including lengths and other user controllable data before being used.

GRT_PLC-SGE100_08 - Several Buffers Overflow on showMeterReport

Host(s) / File(s)	index.cgi
Category	CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CVSSv3	8.8 (High) - CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
CVE	CVE-2025-11780

Threat and Impact

Buffer Overflow vulnerability due to unbounded user input being copied to a fixed-size buffer through sprintf(). The function GetParameter("meter") retrieves user input that is directly incorporated into a buffer without size validation. An attacker can supply an oversized input for the "meter" parameter, causing a buffer overflow that could:

- Overwrite adjacent memory
- Corrupt the stack
- Potentially lead to arbitrary code execution
- 1Cause denial of service through application crashes

```
void showMeterReport()
{
      char acStack 60 [64];
        uVar5 = GetParameter("meter"); /* user
controlled */
      sprintf(acStack 60,"Request(\'MeterMenu\',\'&meter=
%s\')",uVar5);
}
void showMeterReport()
{
  char acStack 60 [64];
  uVar5 = GetString(iVar4 + 0x38); /* fixed value */
  uVar6 = GetParameter("meter"); /* user controlled */
  sprintf(acStack 60,"%s   -  
%s",uVar5,uVar6);

}
```

Recommendations

Validate all user input including lengths and other user controllable data before being used.

GRT_PLC-SGE100_09 - Several Buffers Overflow on ShowSupervisorParameters Function

Host(s) / File(s)	index.cgi
Category	CWE-122: Heap-based Buffer Overflow
CVSSv3	8.9 (High) - CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:H
CVE	CVE-2025-11788

Threat and Impact

Buffer Overflow vulnerability due to unbounded user input being copied to a fixed-size registry through sprintf(). The function GetParameter("meter") retrieves user input that is directly incorporated into a buffer without size validation. An attacker can supply an oversized input for the "meter" parameter, causing a buffer overflow that could:

- Overwrite adjacent memory
- Corrupt the stack
- Potentially lead to arbitrary code execution
- Cause denial of service through application crashes

```
void ShowSupervisorParameters(void)

{
  undefined4 uVar1;
  undefined4 uVar2;
  tm *ptVar3;
  int iVar4;
  char *pcVar5;
  uint uVar6;
  undefined4 local 64; /*arm registry is 32-bits long)*/
  undefined1 local 60;
  undefined1 local 5f;
  time t local 24;

  uVar1 = GetString(0x4f);
  uVar2 = GetParameter("meter");
  sprintf((char *)&local 64,"%s   -  
%s",uVar1,uVar2);
  printf("<table style=\"width:100%%;\"><tr
align=\"center\"><td class=\"RSN\" style=\"height:30px\"
> %s</td></tr></table>\n"
          ,&local 64);
      /*local 64 is fixed but uVar2 is user controlled,
so max is 32-bits long*/

  uVar1 = GetParameter("meter");
```

```
  sprintf((char
*)&local_64,"Request(\'MeterMenu\',\'&meter=%s\')",uVar1
);
/*same as above*/
}
```

Recommendations

Validate all user input including lengths and other user controllable data before being used.

GRT_PLC-SGE100_10 - [PLC] Buffer Overflow On SetUserPassword Function

Host(s) / File(s)	PLC Firmware
Category	CWE-121: Stack-based Buffer Overflow
CVSSv3	8.9 (High) - CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:H
CVE	CVE-2025-11786

Threat and Impact

The newPassword parameter is directly embedded into a shell command string via sprintf() without any sanitization or validation, and then executed using system(). This allows an attacker to inject arbitrary shell commands that will be executed with the same privileges as the application.

```
int   fastcall SetUserPassword(const char
*targetUsername, const char *newPassword)
{
  char shellCommandBuffer[272]; // [sp+4h] [bp-110h]
BYREF

  sprintf(shellCommandBuffer, "adduser -h /tmp -H -D
\"%s\" > /dev/null 2>&1", targetUsername);
  system(shellCommandBuffer);
  sprintf(shellCommandBuffer, "(echo \"%s\" ; sleep 1;
echo \"%s\") | passwd \"%s\" > /dev/null 2>&1",
newPassword, newPassword, targetUsername); /*
targetUsername is a fixed valued, but value come from
ManageWebMessage which probably is a message coming from
index.cgi and controlled by a user*/
  if ( !system(shellCommandBuffer) )
    return 1;
  PrintLog("SYSTEM: Error setting user password");
  return 0;
}
```

Recommendations

Validate all user input including lengths and other user controllable data before being used.

GRT_PLC-SGE100_11 - [PLC] Command Injection on SetLan Function

Host(s) / File(s)	PLC Firmware
Category	CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CVSSv3	9 (Critical) - CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
CVE	CVE-2025-11779

Threat and Impact

The function SetLan is invoked when new setup is applied. This new setup function is triggered by manage web request, which can be invoked by a user when doing changes in the "index.cgi" web application. The parameters are not being sanitized, therefore it can be lead to command injection.

```
int   fastcall SetLAN(
        const char *ipAddress,
        const char *netmask,
        const char *gateway,
        int enableDhcp,
        const char *dhcpClientId,
        const char *primaryDns,
        const char *secondaryDns)
{
  int ipSetResult; // r0
  int netmaskSetResult; // r0
  int gatewaySetResult; // r0
  int dhcpIdSetResult; // r0
  int primaryDnsSetResult; // r0
  int secondaryDnsSetResult; // r0
  char commandBuffer[292]; // [sp+0h] [bp-124h] BYREF

  if ( system("/etc/init.d/S40networking.sh stop") )
    PrintError((int)"SYSTEM: Error stopping network");
  snprintf(commandBuffer, 0x100u, "ubootenv -s
ipaddr=%s", ipAddress);
  ipSetResult = system(commandBuffer);
  if ( ipSetResult )
    PrintLog("SYSTEM: Error setting ipaddr %d (%s)",
ipSetResult, ipAddress);
  snprintf(commandBuffer, 0x100u, "ubootenv -s
netmask=%s", netmask);
  netmaskSetResult = system(commandBuffer);
  if ( netmaskSetResult )
    PrintLog("SYSTEM: Error setting netmask %d (%s)",
netmaskSetResult, netmask);
  snprintf(commandBuffer, 0x100u, "ubootenv -s
gatewayip=%s", gateway);
  gatewaySetResult = system(commandBuffer);
  if ( gatewaySetResult )
```

```
    PrintLog("SYSTEM: Error setting gatewayip %d (%s)",
gatewaySetResult, gateway);
  if ( enableDhcp == 1 )
    snprintf(commandBuffer, 0x100u, "ubootenv -s
dhcp=on");
  else
    snprintf(commandBuffer, 0x100u, "ubootenv -s
dhcp=off");
  if ( system(commandBuffer) )
    PrintLog("SYSTEM: Error setting DHCP");
  snprintf(commandBuffer, 0x100u, "ubootenv -s
dhcpiden=%s", dhcpClientId);
  dhcpIdSetResult = system(commandBuffer);
  if ( dhcpIdSetResult )
    PrintLog("SYSTEM: Error setting DHCP client
identifier %d (%s)", dhcpIdSetResult, dhcpClientId);
  snprintf(commandBuffer, 0x100u, "ubootenv -s
dnsip=%s", primaryDns);
  primaryDnsSetResult = system(commandBuffer);
  if ( primaryDnsSetResult )
    PrintLog("SYSTEM: Error setting primary DNS %d (%s",
primaryDnsSetResult, primaryDns);
  snprintf(commandBuffer, 0x100u, "ubootenv -s
dnsip2=%s", secondaryDns);
  secondaryDnsSetResult = system(commandBuffer);
  if ( secondaryDnsSetResult )
    PrintLog("SYSTEM: Error setting secondary DNS %d
(%s", secondaryDnsSetResult, secondaryDns);
  if ( system("/etc/init.d/S40networking.sh start") )
    PrintError((int)"SYSTEM: Error starting network");
  return 1;
}
```

Recommendations

Validate all user input including lengths and other user controllable data before being used.

GRT_PLC-SGE100_12 - Out-Of-Bounds Read on DownloadFile

Host(s) / File(s)	index-cgi
Category	CWE-125: Out-of-bounds Read
CVSSv3	6.5 (Medium) - CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
CVE	CVE-2025-11789

Threat and Impact

The code converts the parameter to an integer using atoi() and then uses it as an index into the FilesDownload array with (&FilesDownload)[iVar2]. If the parameter is too large, it will access memory beyond the bounds of the FilesDownload array, causing:

- Reading from unintended memory locations
- Potential access to sensitive information
- Possible segmentation fault/crash

```
void DownloadFile(void)

{
    off t  Var1;
  int iVar2;
  char *pcVar3;
  char *  path;
  char *pcVar4;
  int iVar5;
  stat sStack 70;

  iVar2 = GetParameter("file"); /
  if (iVar2 != 0) {
    pcVar3 = (char *)GetParameter("file");
    iVar2 = atoi(pcVar3);
    if (iVar2 < 44) {
      pcVar3 = (char *)GetParameter("file");
      iVar2 = atoi(pcVar3);
      pcVar3 = (&FilesDownload)[iVar2];
      iVar2 = open(pcVar3,0,0);
```

Recommendations

Validate all user input including lengths and other user controllable data before being used.