Vulnerability Report of CGI application ajy.cgi

In firmware version 01.10100.10.47 (latest) for Jennov P31HH35-3-FAS Wireless Security Camera, CGI application ajy.cgi is vulnerable to multiple stack-based buffer overflows. This document serves to provide the manufacturer (Jennov) enough information to fix and patch their firmware.

Vulnerability Name:

proxy parameter on the /api/v1/group-list endpoint is vulnerable to a stack-based buffer overflow.

Vulnerable URL(s):

`http://< camera ip address >:80/api/v1/group-list?proxy=...`

Vulnerability Description:

proxy parameter on the /api/v1/group-list endpoint is vulnerable to a stack-based buffer overflow. By sending a HTTP POST request to the camera with a specially crafted payload in the **proxy** parameter, an attacker can cause the corruption of program memory and possibly crash the CGI application serving the group-list endpoint.

`http://< camera ip address >:80/api/v1/group-list?proxy=< 513 or more characters >:0&`

This buffer overflow is limited to a maximum of 1024 characters by the Boa HTTP server.

Vulnerability Mitigation

QryGroupList function in ajy.cgi that parses the **proxy** parameter:

```
pcVar2 = strstr(pcVar2,"proxy=");
if (pcVar2 == (char *)0x0) goto LAB_004031d8;\
pcVar2 = pcVar2 + 6;
pcVar3 = strchr(pcVar2,':');
if (pcVar3 != (char *)0x0) {
*pcVar3 = '\0';
}
pcVar3 = strchr(pcVar2,'&');
if (pcVar3 == (char *)0x0) goto LAB_004031d8;
*pcVar3 = '\0';
strncpy((char *)&local_270,pcVar2,(int)pcVar3 - (int)pcVar2); <-- buffer
overflow occurs here
```

This function determines the position of the first substring to match "proxy=" and the position of the first character to match ":". The function subtracts the positions to determine the length of the submitted ip address. A buffer overflow can occur if more than 512 characters are placed in-between these two positions, as the length of the submitted ip address is derived from the user input.

A possible solution could calculate the length of the submitted ip address and if the length is greater then the size of the buffer (512 Characters), bound the length to the buffer size.

```
pcVar2 = strstr(pcVar2,"proxy=");
if (pcVar2 == (char *)0x0) goto LAB_004031d8;\
pcVar2 = pcVar2 + 6;
pcVar3 = strchr(pcVar2,':');
if (pcVar3 != (char *)0x0) {
*pcVar3 = '\0';
}
pcVar3 = strchr(pcVar2,'&');
if (pcVar3 == (char *)0x0) goto LAB_004031d8;
*pcVar3 = '\0';

// Possible Solution
int ip_address_length = (int)pcVar3 - (int)pcVar2;
if (ip_address_length > BUFFER_SIZE)
{
    ip_address_length = BUFFER_SIZE;
}
strncpy((char *)&local_270,pcVar2, ip_address_length);
```

Vulnerability Name:

The **deviceId** JSON Object in HTTP request body on the /api/v1/group-list endpoint and /api/v1/group-calendar endpoint is vulnerable to a stack-based buffer overflow.

Vulnerable URL(s):

`http://< camera ip address >:80/api/v1/group-list`
`http://< camera ip address >:80/api/v1/group-calendar`

Vulnerable Description:

The **deviceId** JSON Object in HTTP request body on the /api/v1/group-list endpoint and /api/v1/group-calendar endpoint is vulnerable to a stack-based buffer overflow. By sending a HTTP POST request to the camera with a specially crafted payload in the **deviceId** JSON Object in HTTP request body, an attacker can cause the corruption of program memory and possibly crash the CGI application serving the group-list or group-calendar endpoint.

```
{
    "data": {
        "deviceId": < 129 or more characters >
    }
}
```

Vulnerability Mitigation

Either QryGroupList or QryGroupCalendar in ajy.cgi (Both functions share the same code):

```
iVar6 = cJSON_GetObjectItem(iVar5,"deviceId");
if (((iVar6 != 0) && (pcVar2 = *(char **)(iVar6 + 0x10), pcVar2 != (char
*)0x0)) &&
    (*pcVar2 != '\0')) {
    strcpy(acStack_170,pcVar2); <--- buffer overflow occurs here
}
```

Both functions uses the cJSON library to parse the HTTP POST request body. If the "deviceId" JSON Object exists, a pointer to its location in the heap is found and then copied into a buffer. A buffer overflow can occur as strcpy does not perform a bound check on the source string and destination string. If an attacker submits a value in the deviceId JSON Object larger than then the destination buffer (128 characters) a buffer overflow will occur.

A possible solution could be to to use strncpy with a size of the destination buffer and ensure the last character of the destination buffer is a NULL character.

```
iVar6 = cJSON_GetObjectItem(iVar5,"deviceId");
if (((iVar6 != 0) && (pcVar2 = *(char **)(iVar6 + 0x10), pcVar2 != (char
*)0x0)) &&
    (*pcVar2 != '\0')) {
    // Possible Solution
    strncpy(acStack_170,pcVar2, BUFFER_SIZE);
    acStack_170[BUFFER_SIZE-1] = '\0';
}
```

Vulnerability Name:

The **accessKey** JSON Object in HTTP request body on the /api/v1/group-list endpoint and /api/v1/group-calendar endpoint is vulnerable to a stack-based buffer overflow.

Vulnerable URL(s):

```
http://< camera ip address >:80/api/v1/group-list
http://< camera ip address >:80/api/v1/group-calendar
```

Vulnerable Description:

The **accessKey** JSON Object in HTTP request body on the /api/v1/group-list endpoint and /api/v1/group-calendar endpoint is vulnerable to a stack-based buffer overflow. By sending a HTTP POST request to the camera with a specially crafted payload in the **accessKey** JSON Object in HTTP request body, an attacker can cause the corruption of program memory and possibly crash the CGI application serving the group-list or group-calendar endpoint.

```
{
    "data": {
        "accessKey": < 129 or more characters >
    }
}
```

Vulnerability Mitigation

Either QryGroupList or QryGroupCalendar in ajy.cgi (Both functions share the same code):

```
iVar4 = cJSON_GetObjectItem(iVar4,"accessKey");
if (((iVar4 != 0) && (pcVar2 = *(char **)(iVar4 + 0x10), pcVar2 != (char
*)0x0)) &&
    (*pcVar2 != '\0')) {
    strcpy(acStack_118,pcVar2); <--- buffer overflow occurs here
}
```

Both functions uses the cJSON library to parse the HTTP POST request body. If the "accessKey" JSON Object exists, a pointer to its location in the heap is found and then copied into a buffer. A buffer overflow can occur as strcpy does not perform a bound check on the source string and destination string. If an attacker submits a value in the "accessKey" JSON Object larger than then the destination buffer (128 characters) a buffer overflow will occur.

A possible solution could be to to use strncpy with a size of the destination buffer and ensure the last character of the destination buffer is a NULL character.

```
iVar4 = cJSON_GetObjectItem(iVar4,"accessKey");
if (((iVar4 != 0) && (pcVar2 = *(char **)(iVar4 + 0x10), pcVar2 != (char
*)0x0)) &&
    (*pcVar2 != '\0')) {
    strncpy(acStack_118,pcVar2, BUFFER_SIZE);
    acStack_118[BUFFER_SIZE-1] = '\0';
}
```