Network Research and Monitoring

Author: Yvette K

CONTENTS

Introduction

An automation was created to let cyber units run script from their local devices but executed by the remote server, communicating with the server and executing automatic tasks anonymously.

To better understand network and security, we captured and monitored the traffic during the automated attack on the server.

We then analysed the FTP protocol which was used in the automation and studied how it impacts the CIA Triad. Secure alternatives were thereafter provided with demonstration.

Methodologies

A bash script was first created with geany and language was set (*#!/bin/bash*).

Functions were created to check if the applications required for the script are already installed in the machine using if-else statements.

dpkg command was used to check the status of the application. *grep* was used to determine if the application's status is installed.

Going into the directory where the nipe.pl file is located, the database is updated via *sudo updated* command and the *find* command was used to locate the directory nipe, also creating the directory as a variable which would be used to access the nipe.pl file.

The *sudo perl nipe.pl start* command is used to attempt to connect to the Tor network to achieve anonymity. An if-else statement was used to check status and start *perl* if unsuccessful, until the connection was successful. Otherwise, the script would end, informing user that the connection was unsuccessful.

Information was then collected from the user using *echo* to pose inquiry and *read* to collect details, such as the domain that the user wished to target, and information to access the remote server – IP address, username and password.

sshpass is used to log into the remote server using data collected from user to access the remote server and to collect information we need.

Before proceeding with login, we scan the remote server using *nmap* to ensure that its ssh port is open. Once in the server, we check its IP address, country location and its uptime using commands of the IP address variable collected from user, *whois* and *grep*, and *sshpass* and *uptime* respectively.

Next, we proceeded to use the remote server to check the victim's domain on whois and saved the data on the remote machine. A log was created to document the day, date and time of what domain was scanned using flag *"[$(date "+%a %b %d %Y %H:%M:%S")]"* and using the append command to ensure that data is not overwritten.

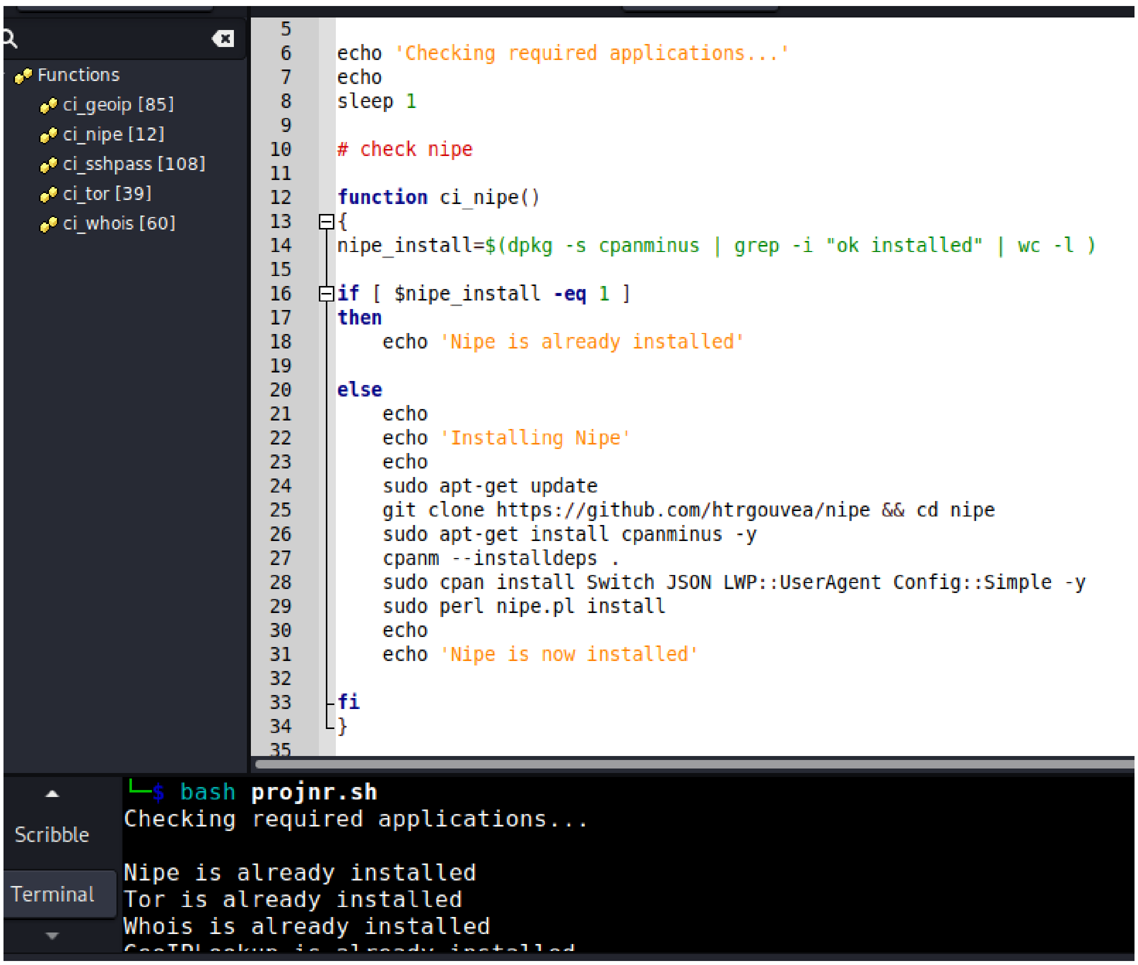
Using *EOF* command to automate FTP, we were able to collect the file from the remote machine and save it into our local machine.

Finally, we also used the remote server to do a *nmap* scan on the victim's address to check its port states, using the *Pn* flag to disable host discovery, *v0* to run the command silently and oX to write its output to an xml file, which can be converted to an html file using the *xsltproc* command.

Data was similarly collected in the same log and saved on local machine and both files' locations were conveyed to the user.

Discussion

We created functions to check if the applications required for the script were already installed in the machine. If-else statements were used to check if they are installed, if the condition was true, a statement would be echoed to notify users that the application was already installed on the machine. Otherwise, the application would be automatically installed after the apt-get update was run.



dpkg command was used to check the status of the application and *-s* flag was used to silence any progress or error messages. *dpkg* command is a tool to install, build, remove and manage Debian packages and is used to interact with packages on our system. *grep* was used to determine if the application's status is installed, if so, that would generate one line that fulfiled the condition of the if-else statement.

geoip [85]
nipe [12]
sshpass [108]
tor [39]
whois [60]

```
sudo updatedb

nipe_dir=$(sudo find / -type d -name nipe 2>/dev/null)

cd "$nipe_dir"

sudo perl nipe.pl start

nipe_connect=$(sudo perl nipe.pl status | grep -i true | wc -l)

if [ $nipe_connect -eq 1 ]
then
    echo "You are anonymous."
    echo "Connecting to the server..."
    echo
    ip=$(curl -s ifconfig.io)
    echo "Your spoofed IP address is: $ip"

    ctry=$(geoiplookup $ip | awk -F: '{print $2}')
    echo "Spoofed country: $ctry"

else
    echo "Can't connect to server"

    exit
```

```
GeoIPLookup is already installed
Sshpass is already installed

You are anonymous.
Connecting to the server...

Your spoofed IP address is: 94.228.169.70
Spoofed country:  AT, Austria
```

After confirming that all applications required were installed on the machine, we proceeded to go into the directory where the nipe.pl file is located. We first updated the database via *sudo updated* command and used the *find* command to locate the directory nipe, also creating the directory as a variable which would be used to access the nipe.pl file.

We then used the *sudo perl nipe.pl start* command to attempt to connect to the Tor network to achieve anonymity. Nipe is a tool developed for users to work anonymously by hiding their details and sending outgoing traffic via the Tor network.

Since it is common to require multiple tries to successfully connect, another if-else statement was used to check status and start *perl* if unsuccessful, until the connection was successful. After which a message would inform users that they were anonymous and connecting to the server. Otherwise the script would end, informing user that the connection was unsuccessful and they would have to run the script again.

Upon connection, the script also generated the spoofed IP address and its country location so that we would be able to identify our machine in any recordings.

```
echo "Enter IP address or URL to whois from remote server: "
read userip
echo

echo "Enter IP address of Remote Server: "
read rserver
echo
echo "Enter username of Remote Server: "
read ruser
echo
echo "Enter password of Remote Server: "
read rpass
```

```
Enter IP address or URL to whois from remote server:
scanme3.nmap.com

Enter IP address of Remote Server:
192.168.80.129

Enter username of Remote Server:
tc

Enter password of Remote Server:
tc
```

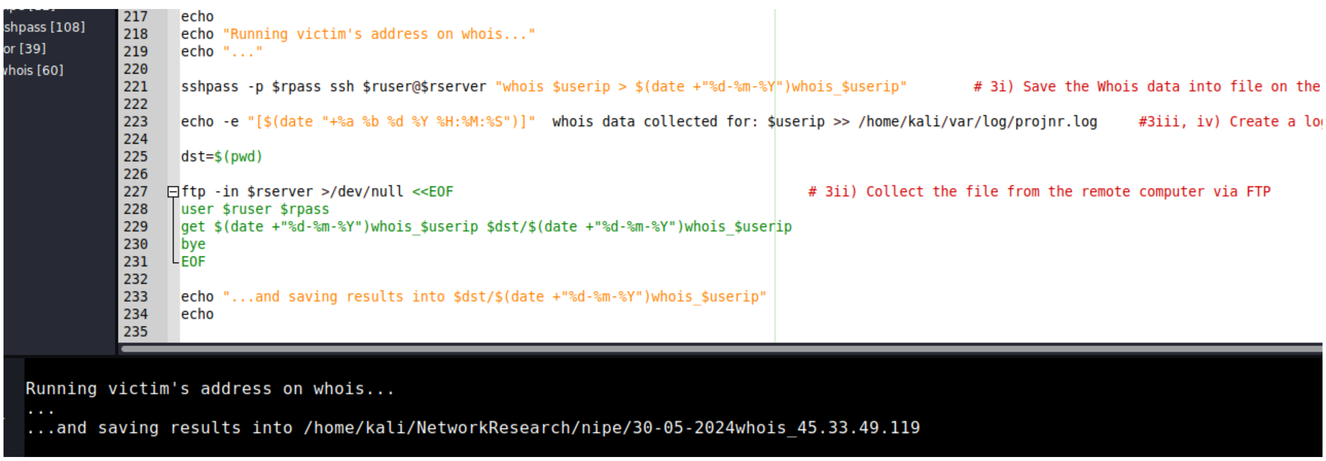
Information was then collected from the user using *echo* to pose inquiry and *read* to collect details, such as the domain that the user wished to target, and information to access the remote server – IP address, username and password.

```

ssh_open=$(nmap $rserver | grep open | grep ssh | wc -l)

echo

if [ $ssh_open -eq 1 ]
then
    echo "Remote Server IP address:"
    echo $rserver
    echo
    echo "Remote Server Country:"
    whois $rserver | grep -i country | awk -F: '{print $2}' | sed 's/ //g'
    echo
    echo "Remote Server Uptime:"
    sshpass -p $rpass ssh $ruser@$rserver 'uptime'

fi
```

```
Remote Server IP address:
192.168.80.129

Remote Server Country:
US

Remote Server Uptime:
 14:07:27 up 10:00,  1 user,  load average: 0.01, 0.02, 0.00
```

Since this is an automation, *sshpass* is used to log into the remote server since it is an excellent tool for non-interactive SSH login. It is a simple and lightweight command line tool that enables us to provide password to the command prompt itself so that automated shell scripts can be executed to take backups via cron scheduler, achieving non-interactive password authentication. However, it is prudent to note that using sshpass is considered to be the least secure as it reveals the password to all system users on the command line with a simple *ps* command. This can be circumvented by using instead SSH Passwordless authentication.

Before proceeding with login, we scanned the remote server using *nmap* to ensure that its ssh port was open. Once in the server, we checked its IP address, country location and its uptime. Uptime monitoring can show whether a server is consistently available or experiences frequent downtimes. High uptime indicates the server is reliable and available for users, while low uptime may signal potential issues that need to be addressed.



Next, we proceeded to use the remote server to check the victim's domain on whois and saved the data on the remote machine. A log was created to document the day, date and time of what domain was scanned.

Using *EOF* command to automate FTP, we were able to collect the file from the remote machine and save it into our local machine.

Finally, we also used the remote server to do a *nmap* scan on the victim's address to check its port states, using the *Pn* flag to disable host discovery, *v0* to run the command silently and oX to write its output to an xml file.

More flags could have been used such as *iL* to scan targets from a given file, *sW* for a TCP Window port scan, specify ports using *p*, *-sV -version-intensity* (level 0-9) where higher number increases possibility of correctness while attempting to determine the version of the service running on port, or *A* which enables OS detection, version detection, script scanning and traceroute, to name a few. A useful script example is nmap -script whois* domain.com which can also run a Whois query to replace the command used in our script.

```
echo
echo "Scanning victim's address..."
echo "..."


sshpass -p $rpass ssh $ruser@$rserver "nmap -Pn -v0 $userip -oX $(date +"%d-%m-%Y")nmap_$userip.xml"        # Save the nmap data ir

echo -e "[$(date "+%a %b %d %Y %H:%M:%S")]"  nmap data collected for: $userip >> /home/kali/var/log/projnr.log      #3 Create a log

ftp -in $rserver >/dev/null <<EOF                                     # Collect the file from the remote computer via FTP
user $ruser $rpass
get $(date +"%d-%m-%Y")nmap_$userip.xml $dst/$(date +"%d-%m-%Y")nmap_$userip.xml
bye
EOF

echo "...and saving results into $dst/$(date +"%d-%m-%Y")nmap_$userip.xml"
```

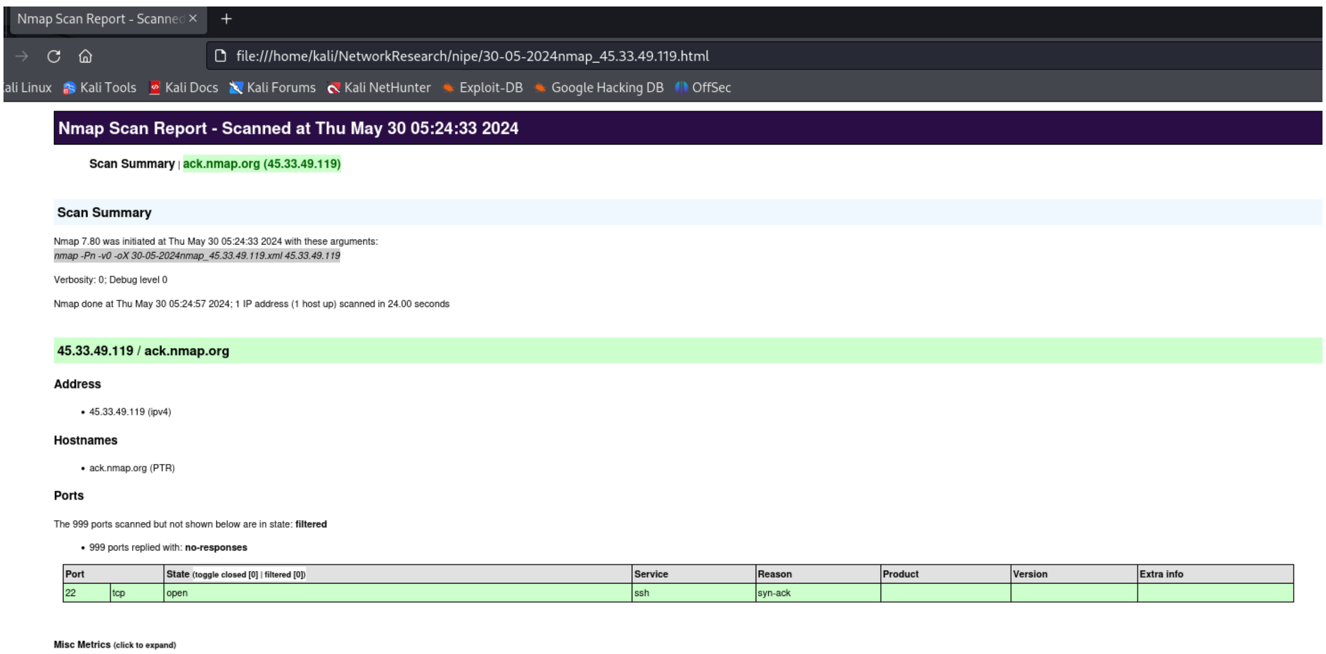
```
Scanning victim's address...
...
...and saving results into /home/kali/NetworkResearch/nipe/30-05-2024nmap_45.33.49.119.xml
```

Data was similarly collected in the same log and saved on local machine and both files' locations were conveyed to the user.

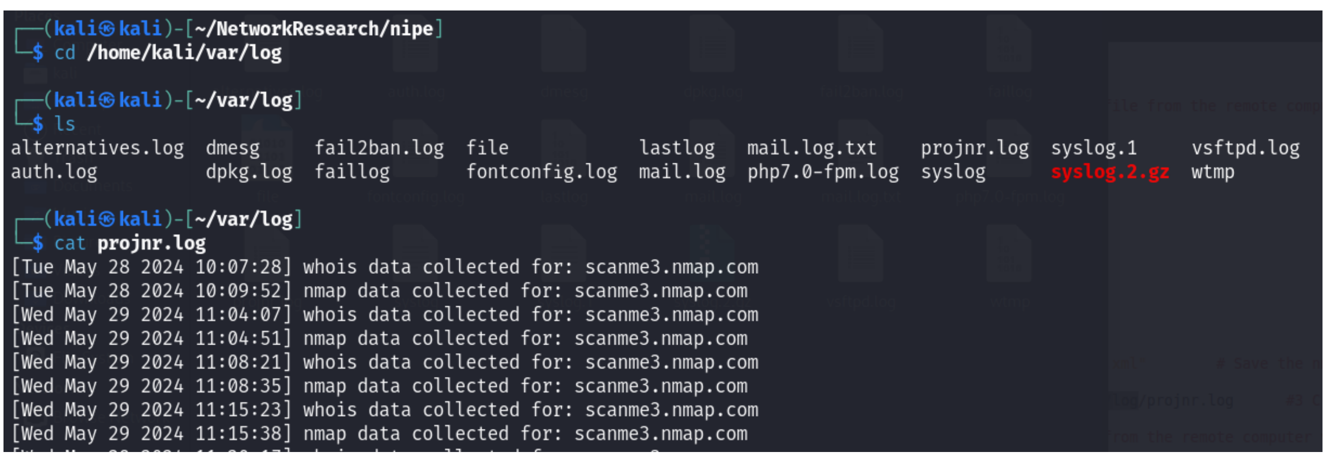
To analyse the data collected, we accessed the directory where the files are stored and *cat* them.

```
┌──(kali㉿kali)-[~/NetworkResearch/nipe]
└─$ ls
30-05-2024nmap_45.33.49.119.html  30-05-2024nmap_45.33.49.119.xml  cpanfile    lib         nipe.pl    SECURITY.md
30-05-2024nmap_45.33.49.119.on    30-05-2024whois_45.33.49.119     Dockerfile  LICENSE.md  README.md

┌──(kali㉿kali)-[~/NetworkResearch/nipe]
└─$ cat 30-05-2024whois_45.33.49.119

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2024, American Registry for Internet Numbers, Ltd.
#


# start

NetRange:       45.33.0.0 - 45.33.127.255
CIDR:           45.33.0.0/17
NetName:        LINODE-US
NetHandle:      NET-45-33-0-0-1
Parent:         NET45 (NET-45-0-0-0-0)
NetType:        Direct Allocation
OriginAS:       AS3595, AS21844, AS6939, AS8001
Organization:   Akamai Technologies, Inc. (AKAMAI)
```

The *oX* flag was used to write the nmap scan output into an xml file as it gives more information than other formats, which can be converted to an html file using the *xsltproc* command and read on a browser.

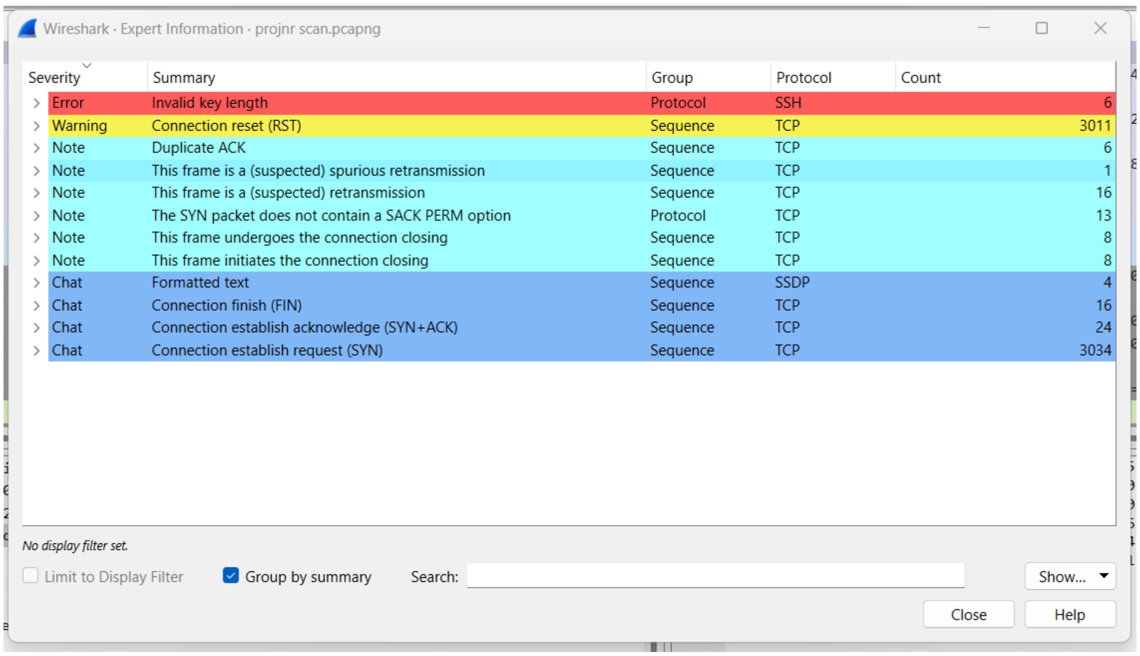


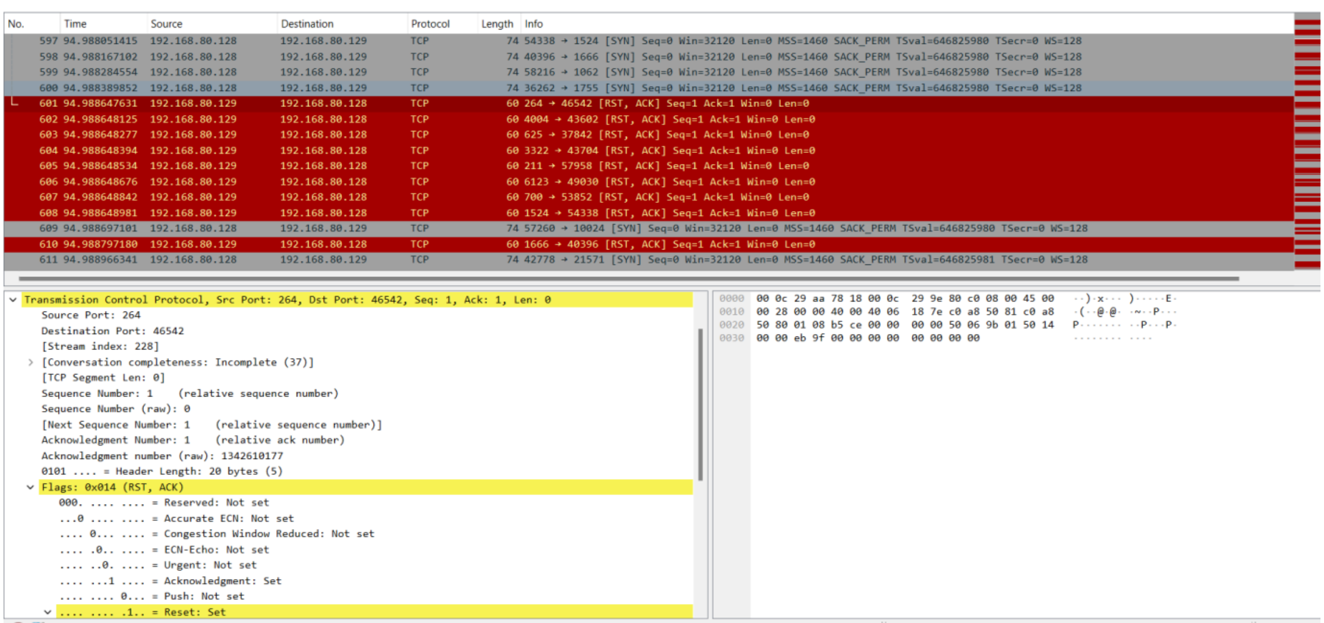
For the log files, we went into the */var/log* directory and looked at the log file saved.



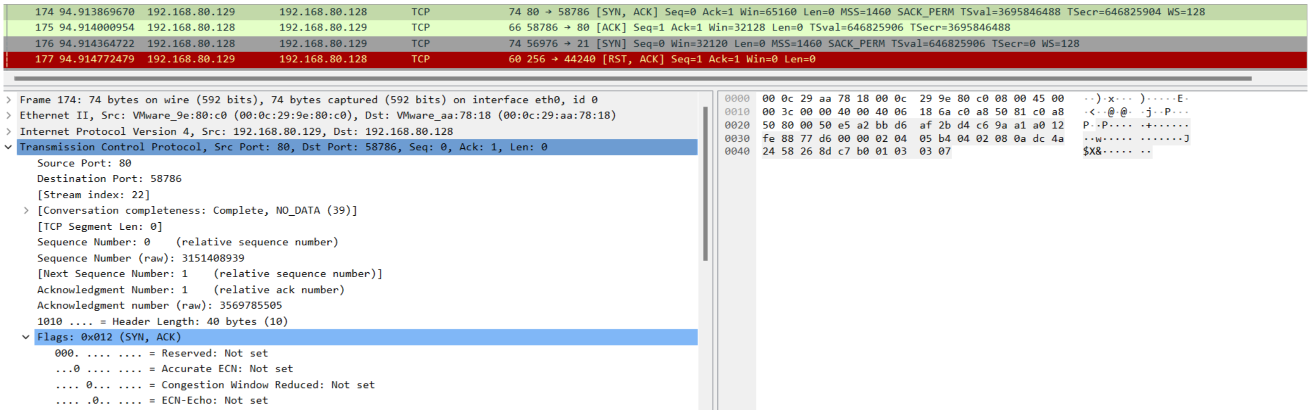
Capture Network Research

Reviewing the Expert Information generated by wireshark we could see that almost half the packages contained warning of Connection reset (RST) which indicated the portions of the automation where nmap was scanning ports of the remote server and found closed ports. The remote server was sending RST, ACK (Reset Acknowledge) to reset the connection and stop. This is in response to the initial SYN package sent.

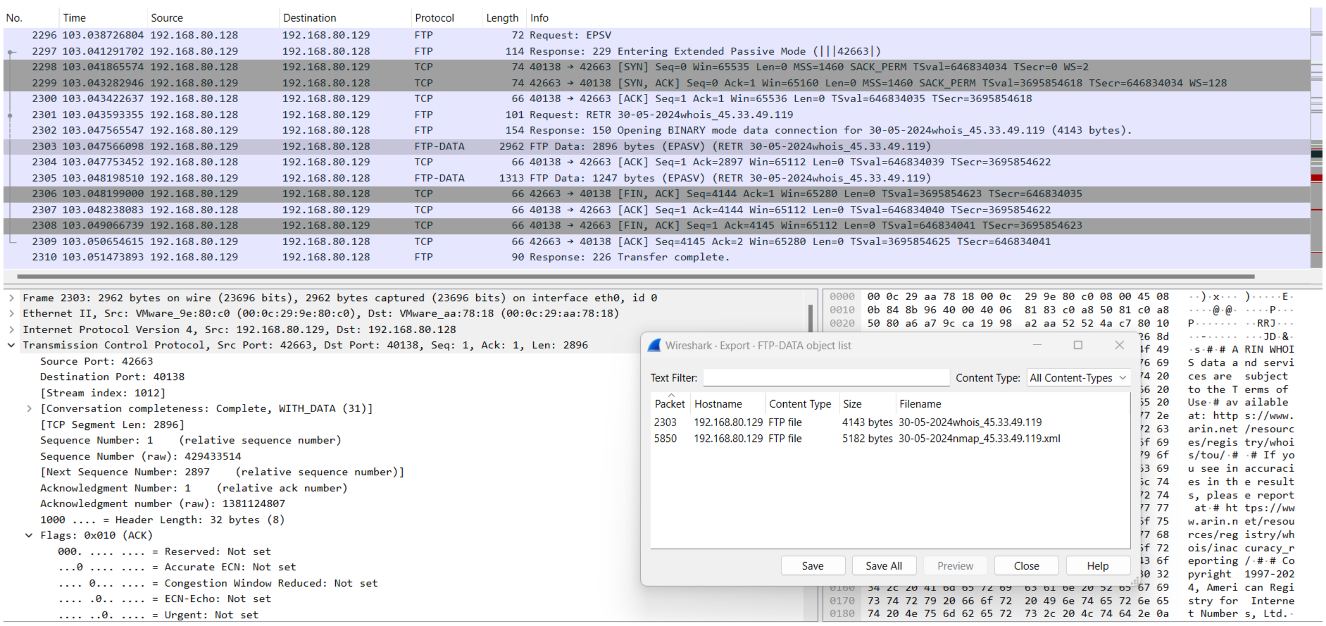




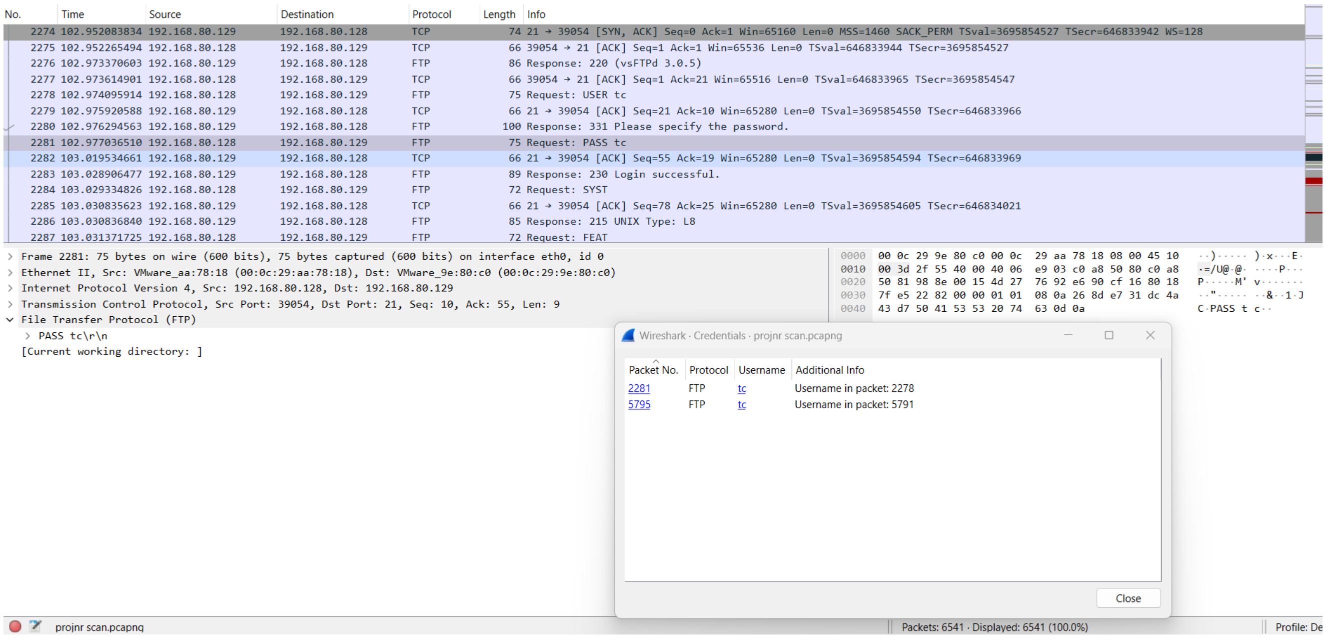
When the scan tried port 80, which was open, it sent SYN, ACK to acknowledge the synchronise.

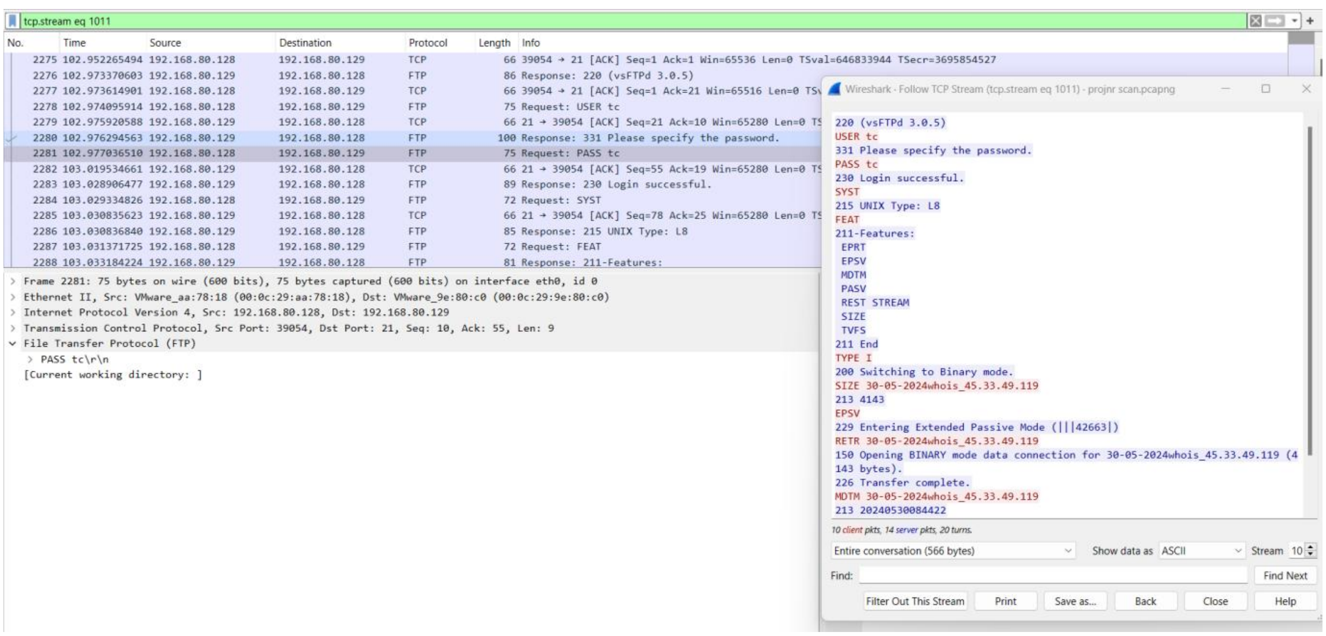


From wireshark we could also trace the whole conversation of how the files were transferred via FTP. The file is downloadable from wireshark and visible in the bottom right window.



Credentials were also easy to retrieve from this wireshark capture.


No. Time Source Destination Protocol Length Info
2274 102.952083834 192.168.80.129 192.168.80.128 TCP 74 21 → 39054 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3695854527 TSecr=646833942 WS=128
2275 102.952265494 192.168.80.128 192.168.80.129 TCP 66 39054 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=646833944 TSecr=3695854527
2276 102.973370603 192.168.80.129 192.168.80.128 FTP 86 Response: 220 (vsFTPd 3.0.5)
2277 102.973614901 192.168.80.128 192.168.80.129 TCP 66 39054 → 21 [ACK] Seq=1 Ack=21 Win=65516 Len=0 TSval=646833965 TSecr=3695854547
2278 102.974095914 192.168.80.128 192.168.80.129 FTP 75 Request: USER tc
2279 102.975920588 192.168.80.129 192.168.80.128 TCP 66 21 → 39054 [ACK] Seq=21 Ack=10 Win=65280 Len=0 TSval=3695854550 TSecr=646833966
2280 102.976294563 192.168.80.129 192.168.80.128 FTP 100 Response: 331 Please specify the password.
2281 102.977036510 192.168.80.128 192.168.80.129 FTP 75 Request: PASS tc
2282 103.019534661 192.168.80.129 192.168.80.128 TCP 66 21 → 39054 [ACK] Seq=55 Ack=19 Win=65280 Len=0 TSval=3695854594 TSecr=646833969
2283 103.028906477 192.168.80.129 192.168.80.128 FTP 89 Response: 230 Login successful.
2284 103.029334826 192.168.80.128 192.168.80.129 FTP 72 Request: SYST
2285 103.030835623 192.168.80.129 192.168.80.128 TCP 66 21 → 39054 [ACK] Seq=78 Ack=25 Win=65280 Len=0 TSval=3695854605 TSecr=646834021
2286 103.030836840 192.168.80.129 192.168.80.128 FTP 85 Response: 215 UNIX Type: L8
2287 103.031371725 192.168.80.128 192.168.80.129 FTP 72 Request: FEAT
Frame 2281: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface eth0, id 0
Ethernet II, Src: VMware_aa:78:18 (00:0c:29:aa:78:18), Dst: VMware_9e:80:c0 (00:0c:29:9e:80:c0)
Internet Protocol Version 4, Src: 192.168.80.128, Dst: 192.168.80.129
Transmission Control Protocol, Src Port: 39054, Dst Port: 21, Seq: 10, Ack: 55, Len: 9
File Transfer Protocol (FTP)
PASS tc\r\n
[Current working directory:]
0000 00 0c 29 9e 80 c0 00 0c 29 aa 78 18 08 00 45 10
0010 00 3d 2f 55 40 00 40 06 e9 03 c0 a8 50 80 c0 a8
0020 50 81 98 8e 00 15 4d 27 76 92 e6 90 cf 16 80 18
0030 7f e5 22 82 00 00 01 01 08 0a 26 8d e7 31 dc 4a
0040 43 d7 50 41 53 53 20 74 63 0d 0a
Wireshark · Credentials · projnr scan.pcapng
Packet No. Protocol Username Additional Info
2281 FTP tc Username in packet: 2278
5795 FTP tc Username in packet: 5791
Close
projnr scan.pcapng
Packets: 6541 · Displayed: 6541 (100.0%)
Profile: De


tcp.stream eq 1011
No. Time Source Destination Protocol Length Info
2275 102.952265494 192.168.80.128 192.168.80.129 TCP 66 39054 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=646833944 TSecr=3695854527
2276 102.973370603 192.168.80.129 192.168.80.128 FTP 86 Response: 220 (vsFTPd 3.0.5)
2277 102.973614901 192.168.80.128 192.168.80.129 TCP 66 39054 → 21 [ACK] Seq=1 Ack=21 Win=65516 Len=0 TS
2278 102.974095914 192.168.80.128 192.168.80.129 FTP 75 Request: USER tc
2279 102.975920588 192.168.80.129 192.168.80.128 TCP 66 21 → 39054 [ACK] Seq=21 Ack=10 Win=65280 Len=0 TS
2280 102.976294563 192.168.80.129 192.168.80.128 FTP 100 Response: 331 Please specify the password.
2281 102.977036510 192.168.80.128 192.168.80.129 FTP 75 Request: PASS tc
2282 103.019534661 192.168.80.129 192.168.80.128 TCP 66 21 → 39054 [ACK] Seq=55 Ack=19 Win=65280 Len=0 TS
2283 103.028906477 192.168.80.129 192.168.80.128 FTP 89 Response: 230 Login successful.
2284 103.029334826 192.168.80.128 192.168.80.129 FTP 72 Request: SYST
2285 103.030835623 192.168.80.129 192.168.80.128 TCP 66 21 → 39054 [ACK] Seq=78 Ack=25 Win=65280 Len=0 TS
2286 103.030836840 192.168.80.129 192.168.80.128 FTP 85 Response: 215 UNIX Type: L8
2287 103.031371725 192.168.80.128 192.168.80.129 FTP 72 Request: FEAT
2288 103.033184224 192.168.80.129 192.168.80.128 FTP 81 Response: 211-Features:
Frame 2281: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface eth0, id 0
Ethernet II, Src: VMware_aa:78:18 (00:0c:29:aa:78:18), Dst: VMware_9e:80:c0 (00:0c:29:9e:80:c0)
Internet Protocol Version 4, Src: 192.168.80.128, Dst: 192.168.80.129
Transmission Control Protocol, Src Port: 39054, Dst Port: 21, Seq: 10, Ack: 55, Len: 9
File Transfer Protocol (FTP)
PASS tc\r\n
[Current working directory:]
Wireshark · Follow TCP Stream (tcp.stream eq 1011) · projnr scan.pcapng
220 (vsFTPd 3.0.5)
USER tc
331 Please specify the password.
PASS tc
230 Login successful.
SYST
215 UNIX Type: L8
FEAT
211-Features:
EPRT
EPSV
MDTM
PASV
REST STREAM
SIZE
TVFS
211 End
TYPE I
200 Switching to Binary mode.
SIZE 30-05-2024whois_45.33.49.119
213 4143
EPSV
229 Entering Extended Passive Mode (|||42663|)
RETR 30-05-2024whois_45.33.49.119
150 Opening BINARY mode data connection for 30-05-2024whois_45.33.49.119 (4
143 bytes).
226 Transfer complete.
MDTM 30-05-2024whois_45.33.49.119
213 20240530084422
10 client pkts, 14 server pkts, 20 turns.
Entire conversation (566 bytes)
Show data as ASCII
Stream 10
Find:
Find Next
Filter Out This Stream
Print
Save as…
Back
Close
Help

FTP Research

File Transfer Protocol (FTP) is a standard network protocol for transmitting files between computers over Transmission Control Protocol/Internet Protocol (TCP/IP) connections. It is an application layer protocol that moves files between local and remote file systems.

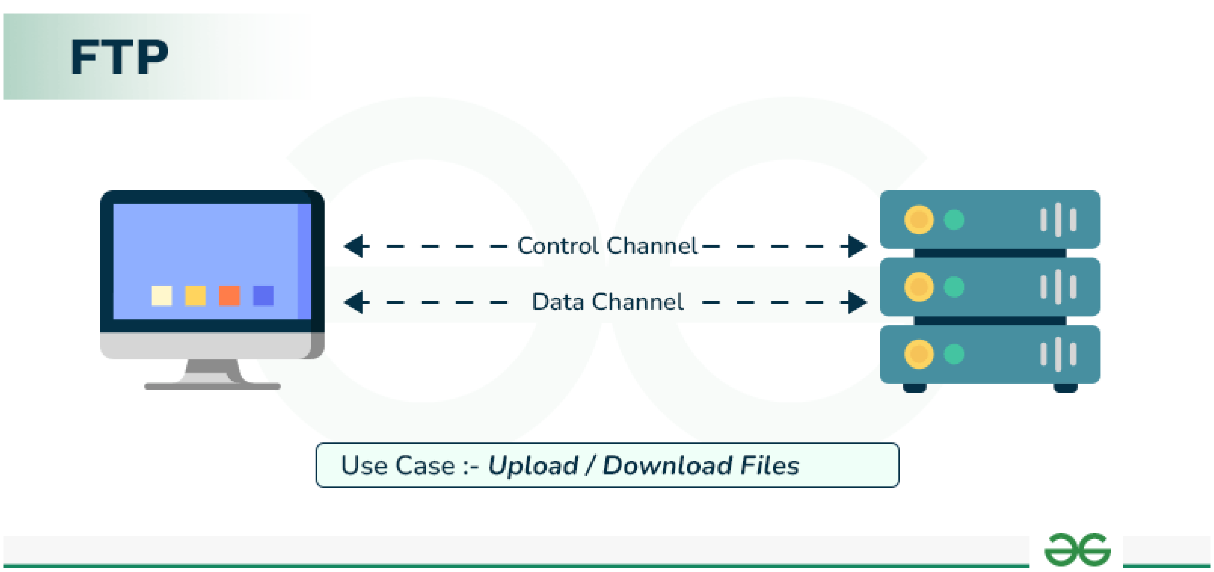
In an FTP transaction, the end user's computer is typically called the local host. The second computer involved in FTP is a remote host, which is usually a server. Both computers need to be connected via a network and configured properly to transfer files via FTP. Servers must be set up to run FTP services, and the client must have FTP software installed to access these services.

Users can work with FTP via a simple command-line interface – from a console or terminal window in Microsoft Windows, Apple macOS or Linux – or with a dedicated graphical user interface. Web browsers can also serve as FTP clients.

There are also built-in FTP programs such as MobaXterm and Putty, which makes it easier to transfer files and it does not require remembering the commands.

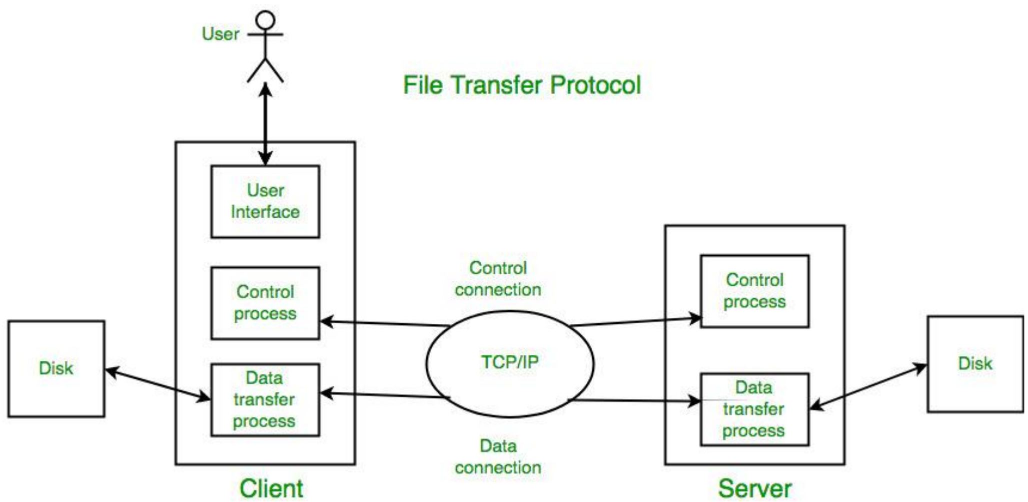
How it works

This client-server protocol relies on two communications channels between the client and server: a command channel for controlling the conversation and a data channel for transmitting file content.



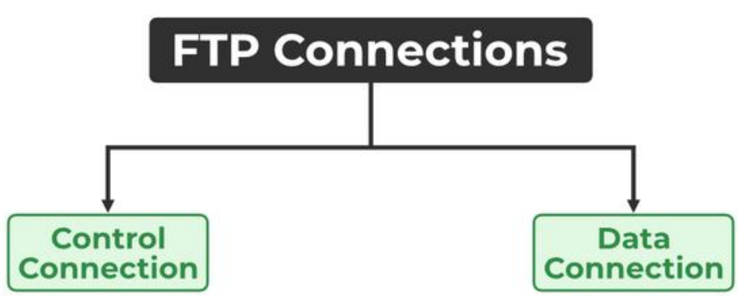
Steps involve:

- User logging in to FTP server (except servers where login is not necessary, e.g. anonymous FTP)
- Client starts conversation with server, upon requesting to download a file
- User can start different functions such as upload, delete, rename, copy files etc on server



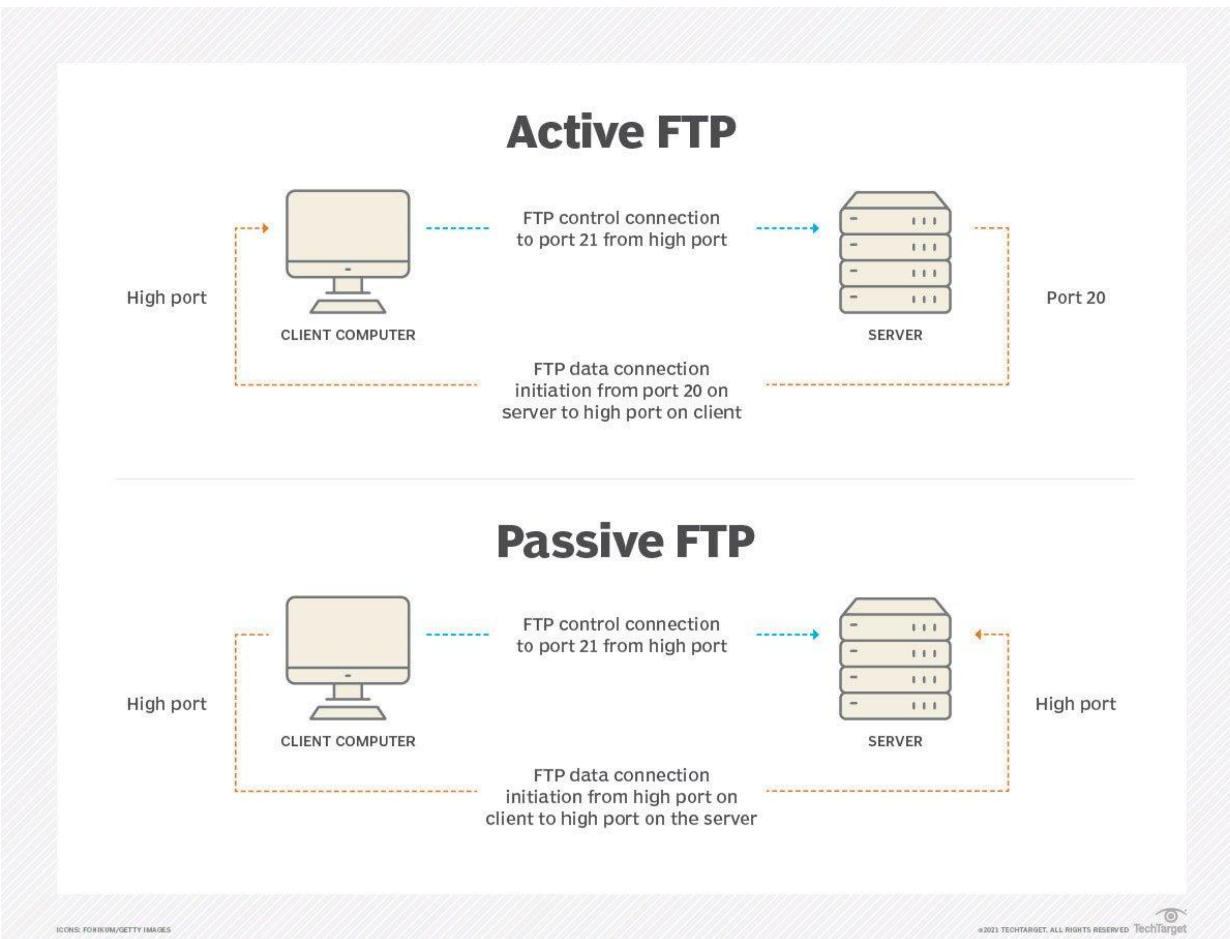
There are two different types of connection in FTP, namely Control Connection and Data Connection. For sending control information like user identification, password, commands to change the remote directory, commands to retrieve and store files etc, FTP makes use of a control connection. This is initiated on port number 21.

For sending the actual file, FTP makes use of a data connection. It is initiated on port number 20. FTP sends the control information out-of-band as it uses a separate control connection. Some protocols such as HTTP and SMTP send their request and response header lines and the data in the same TCP connection, which is to say they send their control information in-band.



When an FTP session is started between a client and a server, the client initiates a control TCP connection with the server side. The client sends control information over this. When the server receives this, it initiates a data connection to the client side. But the control connection remains active throughout the user session. While HTTP is stateless, FTP needs to maintain a state about its user throughout the session.

FTP sessions work in either active or passive modes. In the **active** mode, the client initiates a session via a command channel request and the server creates a data connection back to the client and begins transferring data. In the **passive** mode, the server uses the command channel to send the client the information it needs to open a data channel. The passive mode works well across firewalls and network address translation gateways because it has the client initiating all connections.



The **data type** of a file, which determines how the file is represented overall, is the first piece of information that can be provided about it. The FTP standard specifies the following four categories of data: ASCII, EBCDIC, image and local.

FTP command-line options for Linux and UNIX

Command-Line Option	Description of Command
-4	Use only IPv4 to contact any host.
-6	Use IPv6 only.
-e	Disables command editing and history support, if it was compiled into the **ftp** executable. Otherwise, it does nothing.
-p	Use passive mode for data transfers. Allows the use of **ftp** in environments where a firewall prevents connections from the outside world back to the client machine. Requires the ftp server to support the PASV command .
-i	Turns off interactive prompting during multiple file transfers.
-n	Restrains **ftp** from attempting auto-login upon initial connection. If auto-login is enabled, **ftp** checks the **.netrc** (see **netrc**) file in the user's home directory for an entry describing an account on the remote machine. If no entry exists, **ftp** prompts for the remote machine login name (the default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login.
-g	Disables file name globbing.
-v	The verbose option forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics.
-d	Enables debugging.

Request for Comments (RFC) is a numbered document, which includes appraisals, descriptions and definitions of online protocols, concepts, methods and programmes. RFCs are administered by the IETF (Internet Engineering Task Force). A large part of the standards used online are published in RFCs. Some fundamental RFCs were officially adopted as standards. While a large proportion are not granted "Standard" status, they are still used as such all over the world. The reason behind this is that the individuals or groups working on an RFC primarily use their time to improve protocols and not for the standardisation process.

The first specification for FTP was published as RFC 114 on April 16, 1971, and was written by Abhay Bhushan, then a student at the Massachusetts Institute of Technology. The original idea behind FTP was to enable the transfer of files over ARPANET, the precursor to the internet.

As the modern internet began to take shape, the FTP specification underwent several revisions to align with networking standards, including TCP/IP. In 1980, a new version of FTP was defined in RFC 765 by Jon Postel, a research scientist at the Information Sciences Institute at the University of Southern California at the time. Five years later, FTP was redefined yet again with RFC 959, which introduced new management capabilities for the protocol, including the ability to make and remove a file directory. Prior iterations of FTP were largely limited to transferring files to and from existing file directory structures.

In 1997, RFC 959 was updated with new capabilities defined in RFC 2228 to provide security capabilities. Two years later, FTP was updated with RFC 2428 to support the IPv6 protocol.

Importance of FTP

FTP can enable expansive file transfer capabilities across IP networks. It uses TCP as a transport layer protocol. Compared to other protocols like HTTP which is also used to transfer files between computers, FTP offers more clarity, precision and control. It is also good for simple file transfer, such as during boot time. FTP shields users from differences in operating systems, directories, structures, character sets, etc typical of other file transfer protocols and transfers data efficiently and reliably.

Some of its common use cases include:

Backup – FTP can be used by backup services or individual uses to backup data from one location to a secured backup server running FTP services.

Replication – FTP can facilitate duplication of data from one system to another but takes a more comprehensive approach to provide higher availability and resilience.

Access and data loading – FTP is also commonly used to access shared web hosting and cloud services as a mechanism to load date onto a remote system.

There are several ways through which a server and a client can do a file transfer using FTP.

Types of FTP

Anonymous FTP – This is the most basic form of FTP which provides support for data transfers without encrypting data or using a username and password. It is most commonly used for download of material that is allowed for unrestricted distribution and is enabled on some sites whose files are available for public access. A user can access these files without having any username or password. Instead, the username is set to anonymous, and the password is to the guest by default. Here, user access is very limited. For example, the user can be allowed to copy the files but not to navigate through directories. This works on port 21.

Password Protected FTP – This is also a basic FTP service similar to Anonymous FTP, but it requires the use of a username and password, though the service might not be encrypted or secure. It also works on port 21.

FTP Secure (FTPS) – Sometimes referred to as **FTP Secure Sockets Layer (FTP-SSL)**, this approach enables implicit Transport Layer Security (TLS) as soon as an FTP connection is established. FTPS was initially used to help enable a more secure form of FTP data transfer. It typically defaults to using port 990.

FTP over explicit SSL/TLS (FTPES) – This approach enables explicit TLS support by upgrading an FTP connection over port 21 to an encrypted connection. This is a commonly used approach by web and file sharing services to enable secure file transfers.

Secure FTP (SFTP) - Not technically an FTP protocol but functioning similarly, SFTP is a subset of the Secure Shell (SSH) protocol that runs over port 22. SSH is commonly used by systems administrators to remotely and securely access systems and applications, and SFTP provides a mechanism within SSH for secure file transfer.

FTP Security

FTP was initially defined in 1971, predating TCP and IP, and it has been redefined several times since then to accommodate new technologies, including the use of TCP/IP, or Request for Comments 765 and RFC 959, and IPv6, or RFC 2428.

Information could not go across a secure tunnel since FTP was not intended to do so. Thus, a hacker would not need to struggle with encryption to access or alter data that is usable if they could intercept an FTP transaction. Even with FTP cloud storage, data can still be intercepted and misused if the service provider’s system is attacked.

As a result, data sent via FTP is a target for spoofing, sniffing, brute force, FTP bounce attack, distributed denial-of-service attack and other types of attacks that move somewhat slowly. A hacker might examine an FTP transmission and try to take advantage of any flaws by simply port scanning.

FTP has undergone several updates to enhance FTP security. These include versions that encrypt via an implicit TLS connection (FTPS) or explicit TLS connection (FTPES) or that work with SFTP. By default, FTP does not encrypt traffic, and individuals can capture packets to read usernames, passwords and other data. By encrypting FTP with FTPS or FTPES, data is protected, limiting the ability of an attacker to eavesdrop on a connection and steal data.

Strengths and Weaknesses

Speed and efficiency are the main benefits of FTP. However, this comes at the expense of security as the speed is due to the lack of encryption. This makes it easy for individuals to capture packets to obtain sensitive information such as usernames and passwords.

Multitasking is possible using FTP as transferring and downloading can go in parallel and multiple files and directories can be transferred at the same time. Many FTP clients also have scripting features.

However, a few FTP providers do not offer encryption. Therefore it may be vulnerable to brute-force attacks against user/password authentication spoofing, an FTP bounce attack or a distributed denial-of-service attack. When sending files via FTP, compliance can be a problem.

As one of the basic building blocks of information security, the CIA Triad is likewise a vital piece in establishing secure enterprise file transfers, with its basic principles being Confidentiality, Integrity and Availability. To establish a secure system, these three objectives need to be achieved.

Given its accessibility and lack of encryption, when information is exchanged over FTP, the confidentiality of the said information will be at risk. Information can be relatively easy stolen. It is imperative to establish countermeasures that can mitigate unauthorised access and disclosures. Likewise for data integrity, the ease of access also means that the information can easily be altered using similar methods. For availability, the data can be readily accessible.

Secure File Transfer Protocol (SFTP)

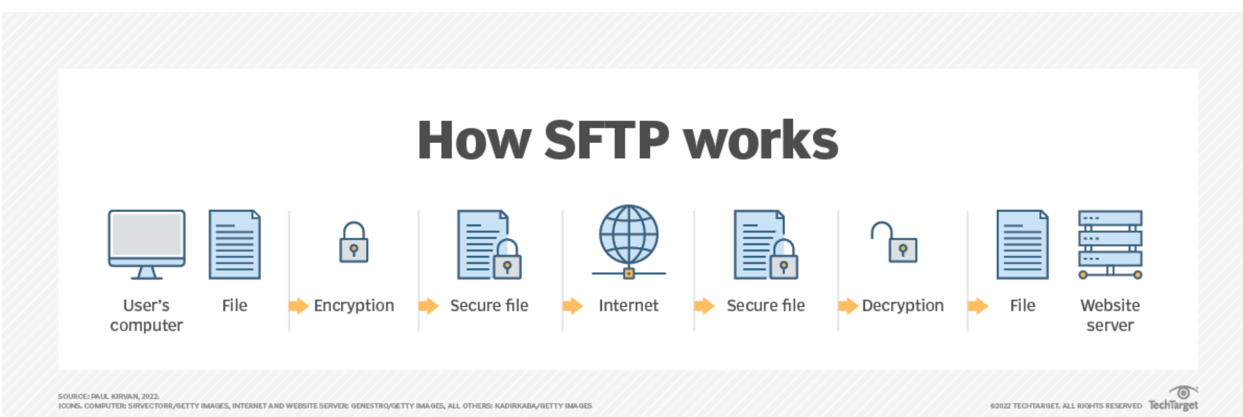
SFTP is a network protocol for securely accessing, transferring and managing large files and sensitive data. Designed by the Internet Engineering Task Force as an extension of Secure Shell (SSH), SFTP enables access, transfer and management of files over a network. It's used for secure file transfers over Transport Layer Security and the transfer of data for virtual private network (VPN) applications

Secure File Transfer Protocol was developed to securely transfer and manage files over a TCP/IP network. It uses the same commands as the standard File Transfer Protocol (FTP) and most SFTP commands are similar or identical to the Linux shell commands. SFTP performs numerous tasks, including transferring sensitive files, removing files and resuming paused transfers. To establish server connection, SFTP only needs to be connected to the normal SSH port 22.

SFTP also needs an SFTP client and server. An STFP client is software that lets users connect to a server and store files on the server. Files are stored and retrieved from the STFP server. When a user clicks on a file, the request travels through the network and ultimately reaches a server. This data is then sent to the requesting device. SFTP ensures all files are encrypted before transferring them.

SSH keys are typically used to automate access to servers and are often used in scripts, backup systems and configuration management tools. SSH keys in SFTP have half of the key stored on the client device, while the other half is on the server associated with a public key. Users are properly authenticated when SSH key pairs match.

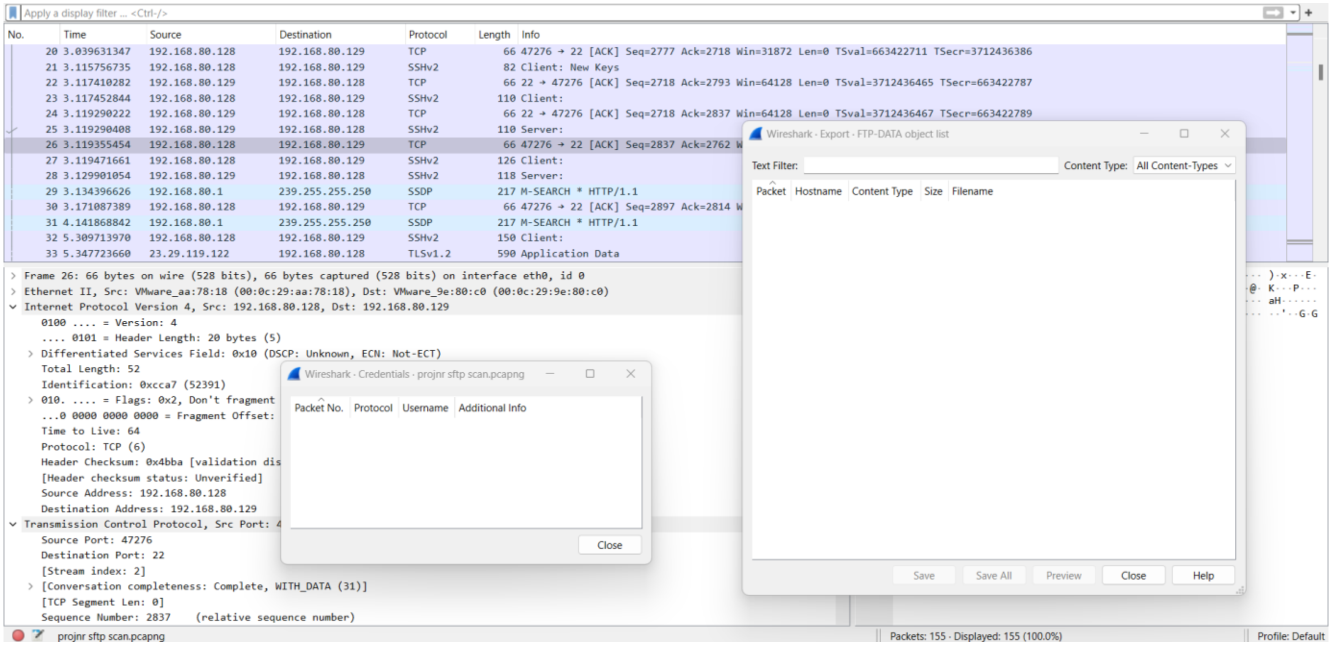
SFTP works over an SSH data stream to establish a secure connection. Encryption algorithms securely move data to a server, keeping files unreadable during the process. To further prevent unauthorised file access, authentication is also enabled. Users can choose to be identified by a user ID and password, SSH keys or both.

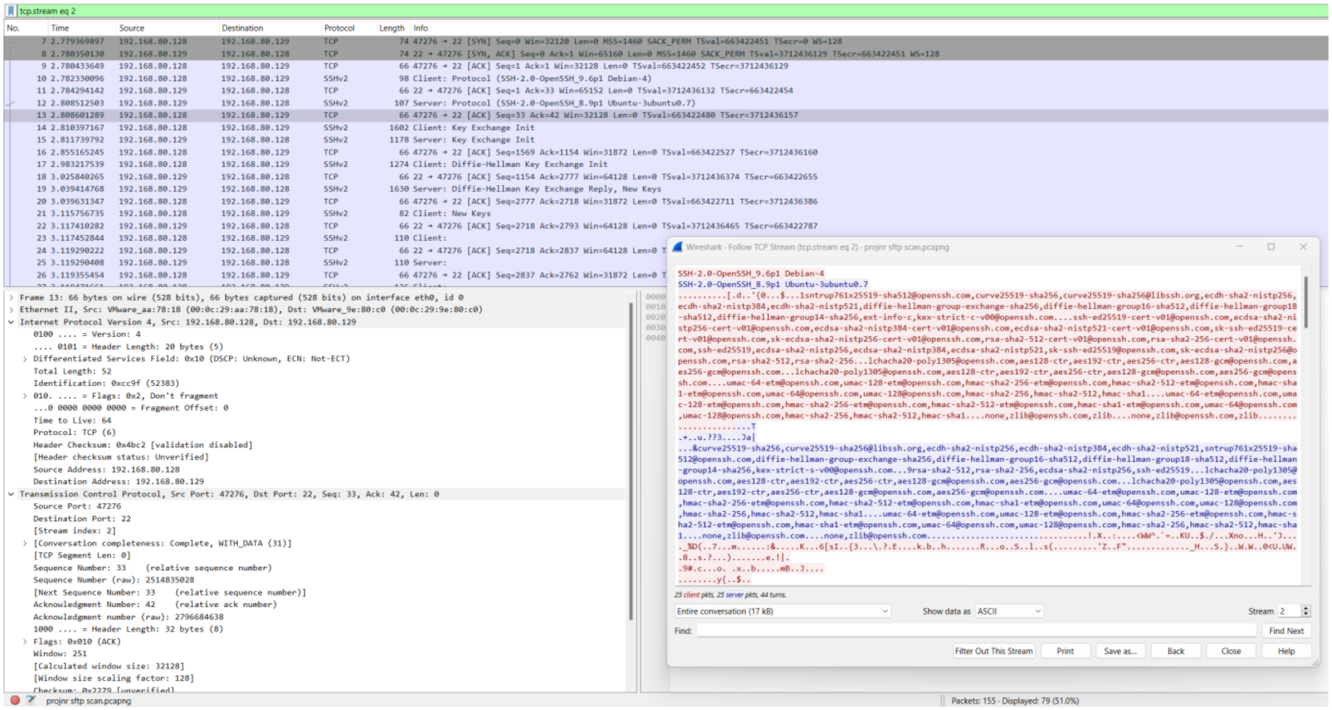


Difference between FTP and SFTP

We used wireshark to capture packages while running SFTP to download a file from the remote server.

We can see that compared to FTP, credentials and the downloaded file are not visible and when attempting to follow the TCP stream, the information is encrypted.





Strengths and Weaknesses

Strengths of SFTP include the ability to protect data in transit by enabling data security, encryption and public key authentication. This security makes SFTP a reliable file transfer option. This protects the confidentiality of information and integrity of the data and thus SFTP helps enterprises meet regulations for file transfer compliance in accordance with HIPAA, GDPR and other regulatory rulings. SFTP also lets businesses securely transfer billing data, funds and data recovery files. Individuals may also use SFTP to encrypt their communications. Commands and data are encrypted to prevent passwords and other sensitive information from being exposed to the network in plain text.

SFTP supports large file transfers and transferring of multiple files from one server to another simultaneously, similar to FTP. It also integrates well with VPNs and firewalls, and can be managed through a web interface or an SFTP client.

Compared to FTP, only one connection is used as data connection is not required in SFTP.

Weaknesses include complexity. Even though SFTP is manageable, the process of creating and setting up an SFTP client is much more complicated than the process of creating an FTP client. Since communication is binary, it is more difficult to login. Its speed is also slow.

SFTP private keys must be stored on the device that users want to transfer files from and the device should also be protected. The keys are difficult to maintain and verify. Standards around SFTP are described as optional and recommended, which may lead to compatibility issues in software developed by different vendors. These may affect the availability of information and thus should be carefully weighed against importance of preserving confidentiality and integrity of said data.

Conclusion

The ease of which we were able to create an automated script to access a remote server to potentially conduct illegal scans anonymously highlights how essential network security is.

Network security protocols are a strong line of defence in the battle against hackers because they can prevent people from infiltrating, corrupting, stealing or taking advantage of our communication system. For example, attackers would have difficulty wiretapping or listening in on communications where encryption would result in them only getting a jumbled set of nonsensical characters.

Protocols intersect with many facets of cybersecurity frameworks because the frameworks use them as tools to keep communications and data safe. Those who design cybersecurity frameworks often piece together various security protocols like components of a machine to keep the networks that use the frameworks running smoothly. Without network security protocols, framework designers may have to code their own solutions instead of using proven, effective technologies.

A well-designed network security solution reduces overhead expenses and safeguards organisations from costly losses that occur from a data breach or other security incident. Since many organisations handle user data, they must ensure the confidentiality, integrity and availability of data on their network. Network security prevents the security breaches that can expose sensitive information, damage a business's reputation and result in financial losses. Ensuring legitimate access to systems, applications and data also enables business operations and delivery of services and products to customers.

Recommendations

By setting up our security protocols correctly, we can prevent denial of service attacks. For example, IPsec can be used to require that each entity that interacts with your network goes through an authentication procedure. This would prevent hackers from randomly sending thousands or millions of requests to a server in a denial-of-service attack because they wouldn't be able to access the server in the first place.

Network security encompasses all the steps taken to protect the integrity of a computer network and the data within it. Successful network security strategies employ multiple security solutions to protect users and organisations from malware and cyber attacks.

Strong security often involves using multiple approaches, known as *layered security* or *defence in depth* to give organisations as many security controls as possible. Other than access control covered by SFTP, some other commonly used types of network security tools and software include:

- **Antivirus and animalware.** These are software designed to detect, remove or prevent viruses and malware from infecting a computer and consequently a network.

- **Application security.** It is crucial to monitor and protect applications that organisations use to run their businesses. This is true whether an organisation creates that application or buys it, as modern malware threats often target Open Source code and containers that organisations use to build software and applications.

- **Behavioural analytics.** This method analyses network behaviour and automatically detects and alerts organisations to abnormal activities.

- **Data loss prevention (DLP).** These tools monitor data in use, in motion and at rest to detect and prevent data breaches. DLP often classifies the most important and at-risk data and trains employees in best practices to protect that data. For instance, not sending important files as attachments in emails is one such best practice.

- **Email security.** Email is one of the most vulnerable points in a network. Employees become victims of phishing and malware attacks when they click on email links that secretly download malicious software. Email is also an insecure method of sending files and sensitive data that employees unwittingly engage in.

- **Firewall.** Software or firmware inspects incoming and outgoing traffic to prevent unauthorised network access. Firewalls are some of the most widely used security tools and are positioned in multiple areas on the network. Next-generation firewalls offer increased protection against application-layer attacks and advanced malware defence with inline deep packet inspection.

- **Intrusion detection system (IDS).** An IDS detects unauthorised access attempts and flags them as potentially dangerous but does not remove them. An IDS and an intrusion prevention system (IPS) are often used in combination with a firewall.

- **Intrusion prevention system.** IPSes are designed to prevent intrusions by detecting and blocking unauthorised attempts to access a network.

- **Sandboxing.** This approach lets organisations scan for malware by opening a file in an isolated environment before granting it access to the network. Once opened in a sandbox, the organisation can observe whether the file acts in a malicious way or shows any indications of malware.

- **Security information and event management (SIEM).** This security management technique logs data from applications and network hardware and monitors for suspicious behavior. When an anomaly is detected, the SIEM system alerts the organisation and takes other appropriate action.

- **Software-defined perimeter (SDP).** An SDP is a security method that sits on top of the network it protects, concealing it from attackers and unauthorised users. It uses identity criteria to limit access to resources and forms a virtual boundary around networked resources.

- **Virtual private network (VPN).** A VPN secures the connection from an endpoint to an organization's network. It uses tunnelling protocols to encrypt information sent over a less secure network. Remote access VPNs let employees access their company network remotely.

- **Zero-trust network access.** Similar to network access control, zero-trust network access only grants a user the access they must have do their job. It blocks all other permissions.

References

e09330e1749e4191a9dfe441e5129f. "The dpkg Command in Linux - A Beginners Reference" *DigitalOcean*, 4 August 2022, https://www.digitalocean.com/community/tutorials/dpkg-command-in-linux

exploitone. "How to Anonymously Use Kali OS For Hacking" *Medium*, 1 April 2020, https://medium.com/@exploitone1/how-to-anonymously-use-kali-os-for-hacking-bf95b8e08677

Kili, Aaron. "sshpass: An Excellent Tool for Non-Interactive SSH Login – Never Use on Production Server" *TECMINT*, 16 December 2016, https://www.tecmint.com/sshpass-non-interactive-ssh-login-shell-script-ssh-password/

Ashwathnarayana, Satyadeep. "Server Uptime Monitoring: Why do we need it? Ensuring High Availability with Continuous Monitoring" *Netdata*, 2 May 2023, https://www.netdata.cloud/blog/server-uptime-monitoring-why-do-we-need-it/#:~:text=Uptime%20monitoring%20can%20show%20whether,that%20need%20to%20be%20addressed

Velimirović, Andreja. "Date Command in Linux: How to Set, Change, Format and Display Date" *phoenixNAP*, 1 October 2020, https://phoenixnap.com/kb/linux-date-command

"How to Automate FTP transfers in Linux Shell Scripting" *eduonix*, 12 December 2015, https://blog.eduonix.com/2015/12/how-to-automate-ftp-transfers-in-linux-shell-scripting/

House, Nathan. "Nmap Cheat Sheet 2024: All the Commands & Flags" *StationX*, 10 May 2024, https://www.stationx.net/nmap-cheat-sheet/

"Output" *nmap.org*, https://nmap.org/book/man-output.html

"FAQ: Why are there duplicate TCP ACKs in my network?" *Informatica*, 19 May 2022, https://knowledge.informatica.com/s/article/148599?language=en_US#:~:text=In%20TCP%20communication%20the%20%E2%80%9CTCP,of%20order%20at%20the%20receiver

"What would cause [RST,ACK]" *Wireshark*, 6 January 2011, https://osqa-ask.wireshark.org/questions/1652/what-would-cause-rstack/#:~:text=This%20is%20very%20simply%20that,possibly%20it%20has%20been%20firewalled

Wallerstorfer, Dirk. “Detecting network errors and their impact on services” *Dynatrace*, 14 December 2022, https://www.dynatrace.com/news/blog/detecting-network-errors-impact-on-services/

“TCP flags” *GeeksforGeeks*, 13 April 2023, https://www.geeksforgeeks.org/tcp-flags/

“Request for Comments (RFC)” *NFON*, https://www.nfon.com/en/get-started/cloud-telephony/lexicon/knowledge-base-detail/request-for-comments-rfc#:~:text=A%20Request%20for%20Comments%20(RFC,online%20are%20published%20in%20RFCs.

“FTP Commands for Linux and UNIX” *Serv-U*, https://www.serv-u.com/linux-ftp-server/commands

“File Transfer Protocol (FTP) in Application Layer” *GeeksforGeeks*, 23 January 2024, https://www.geeksforgeeks.org/file-transfer-protocol-ftp-in-application-layer/

“Difference between File Transfer Protocol (FTP) and Secure File Transfer Protocol (SFTP)” *GeeksforGeeks*, 21 February 2023, https://www.geeksforgeeks.org/difference-between-file-transfer-protocol-ftp-and-secure-file-transfer-protocol-sftp/

Kerner, Sean Michael. “FTP (File Transfer Protocol)” *TechTarget*, May 2021, https://www.techtarget.com/searchnetworking/definition/File-Transfer-Protocol-FTP

Villanueva, John Carl. “Applying The CIA Triad To Your Enterprise File Transfer” *jscape*, 25 October 2022, https://www.jscape.com/blog/implementing-the-cia-triad-when-transferring-files-through-the-internet

Gillis, Alexander S.. “Secure File Transfer Protocol (SSH File Transfer Protocol)” *TechTarget*, October 2022, https://www.techtarget.com/searchcontentmanagement/definition/Secure-File-Transfer-Protocol-SSH-File-Transfer-Protocol?_gl=1*138a2nd*_ga*MjA4NDc3Mzk5MS4xNzExODkxMjc0*_ga_TQKE4GS5P9*MTcxNzA1MjY1Ny42LjEuMTcxNzA1Mjk1Ni4wLjAuMA

S, Edward. “How to Use SFTP (SSH File Transfer Protocol)” *Hostinger*, 20 March 2024, https://www.hostinger.com/tutorials/how-to-use-sftp-to-safely-transfer-files/

“ A deep dive into network security protocols: Safeguarding digital infrastructure 2024” *Infosec*, 19 December 2023, https://www.infosecinstitute.com/resources/network-security-101/a-deep-dive-into-network-security-protocols-safeguarding-digital-infrastructure-2024/#:~:text=The%20role%20of%20protocols%20in,advantage%20of%20your%20communication%20system

"Benefits of Network Security" *Check Point*, https://www.checkpoint.com/cyber-hub/network-security/what-is-network-security/#:~:text=Network%20Security%20is%20vital%20in,as%20protection%20from%20cyber%20threats

Barney, Nick. "DEFINITION network security" *TechTarget*, October 2022, https://www.techtarget.com/searchnetworking/definition/network-security#:~:text=Network%20security%20is%20important%20because,network%20is%20usable%20and%20trustworthy