TZWorks® Network Xfer (*nx*) Client/Server Utility Users Guide

Abstract

nx is a standalone, command-line tool that can act either as a client or a server to exchange data from any client computer (Windows, Linux or macOS) to a forensic workstation. Data that is transferred to the client is encrypted. No elevated permissions are required to run the client.

www.tzworks.com

Contact Info: info@tzworks.com

Document applies to v0.43 of ***nx***

Updated: Apr 25, 2025

Table of Contents

TZWorks® Network Xfer Utility (***nx***) Users Guide

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=18
Contact Information: info@tzworks.com

1 Introduction

nx is a command line tool that acts as either a client, or server, for the purposes of transferring data from more than one computer (acting as clients) to a central computer (acting as the server). The server in this case, would be the forensic workstation gathering information during an incident response. The clients would be those computers that are under investigation.

There are other tools available that perform this function such as *netcat* and *cryptcat*. ***nx***, however, focuses on (a) secure collection, (b) archiving the file metadata, and (c) creating a complete log file of all the transactions that occurred. For extracted files, an MD5 hash is logged as well as the original directory path that the file is taken from.

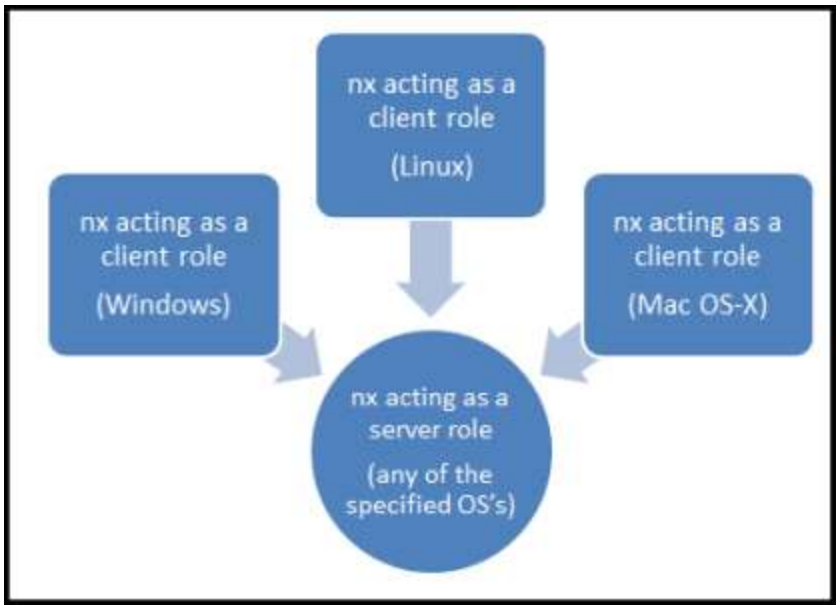
nx operates in one of two modes to transfer data: (a) pulling data from standard input and (b) explicitly copying files. The transport used is TCP/IP with the data content encrypted in a RC4 stream cipher. To ensure data integrity from the client computer to the final archived file on the server, a hash is computed at client side before transmission, and during receipt, by the server. A mismatch in hashes results in the archive file being labeled as having errors. Finally, the results can be controlled by the client end, meaning each specific data transfer is outputted to a separate file. Labels and filenames are allowed to be passed during each data transfer, which are consumed by the server and acted upon accordingly. The number of tunable parameters at this point is still limited, but as suggestions come in, additional flexibility can be added.

Currently, this initial version is restricted to just IPv4. Depending on need, IPv6 can be added. While there are compiled versions for Windows, Linux and macOS, the architecture is such that each one is designed to play well with another instance operating on a different OS (eg. use ***nx*** on a Linux box as the server and use ***nx*** on a Windows client box to send data to the server).

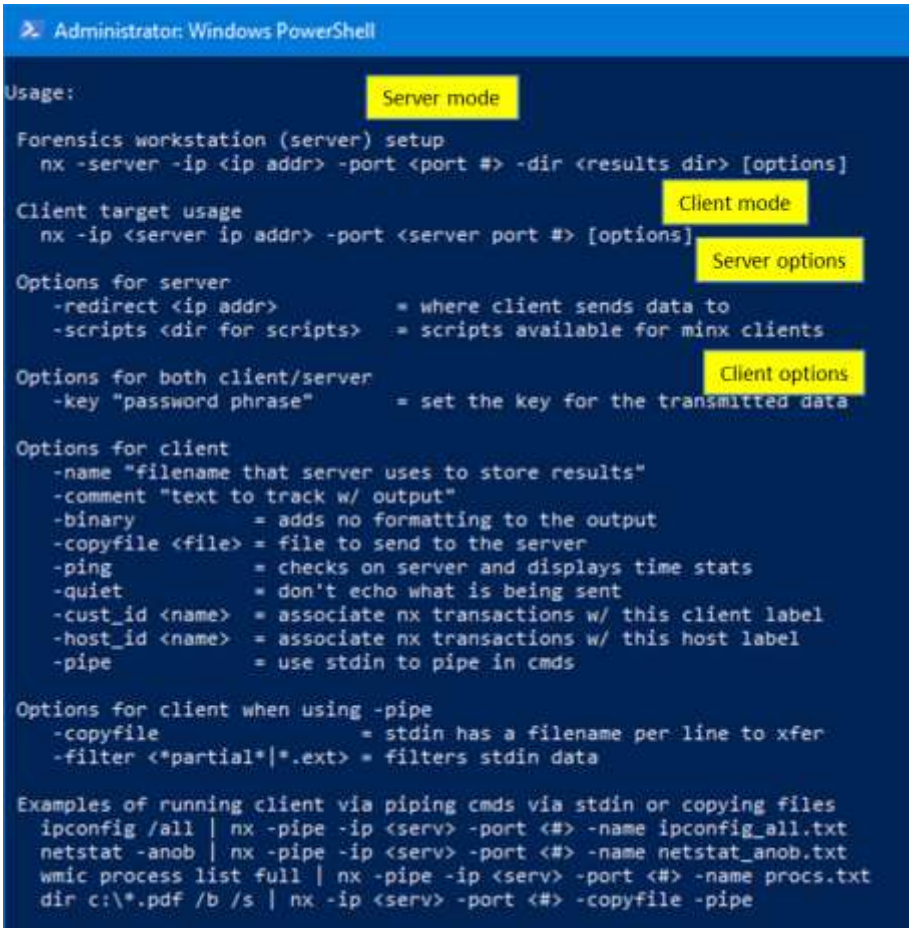
2 How to use *nx* to complement live forensics collection

The terms 'client and server' are used here as 'roles' for the ***nx*** tool. Any machine that ***nx*** runs on does not require the operating system to be configured as a client or server. Any normal (non-server operating system) computer configuration will work. All that is required is that there is some network connectivity between the computers.

To configure ***nx*** as a server role, use the ***-server*** command option. Without the ***-server*** option, ***nx*** runs in client mode. The direction of data flow is always from the client to the server. One can think of the clients as those workstations you would like to extract forensics artifacts from, and the server as the workstation you would like to send the extracted artifacts to. In the server role, ***nx*** can handle multiple clients at once. Since it was designed as a multi-threaded application, ***nx*** simply spawns a separate thread per client connection. Therefore, simultaneous collection from a few clients should not be an issue under normal loading conditions. The maximum number of simultaneous client connections is really a function of: (a) the computer resources of the machine acting in the server role, (b) the amount of data being transferred from each client, and (c) the network bandwidth of the system.



The command line menu shows the options available:



3 Configuring *nx* as a server

Since ***nx*** can be configured as one of two roles (client or server), the menu options are broken out accordingly. The required parameters for the server are: (a) IP address of the server, (b) unique port number to identify how to contact the ***nx*** service, and (c) the directory to store all the collected artifacts to. The one optional parameter is the ***-key*** parameter that can be used to set the crypto key to some user defined passphrase. Without using the ***-key*** option, ***nx*** still encrypts the data transferred over the network (for clients sending data to the server) using its own internal algorithm.

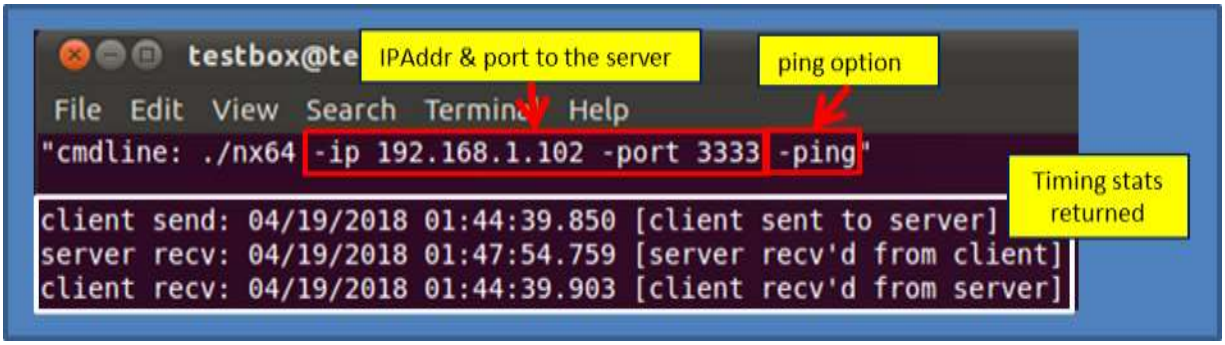
After invoking the above command, the operating system should request permission to open up a network connection at the port specified in the 'listen' mode. For Vista or Windows 7, one should get a popup box similar to the one below.

3.1 Formatting the Results Log

The results log file will display each command that was issued from the client to the ***nx*** service. This is archived as a CSV file. As such, sometimes it is useful to modify how this is rendered, such as excluding whitespace between field delimiters (***-no_whitespace***), to change the field delimiter from a pipe character to a comma character (***-csv_separator ","***), to change the timestamp format (***-dateformat "yyyy/mm/dd"***) or to group the date and time together (***-pair_datetime***). These are the same options available with many of the other *TZWorks* tools, and as such, have the same behavior.

4 Configuring *nx* as a client

Configuring ***nx*** in the client role is a matter of specifying the same IP address and port number used when configuring the server, and then, issuing the desire command to transfer data. As a test, one should try to 'ping' the server from the client computer using the [-ping] option. This will ensure the crypto is synchronized between the client and server. Below is an example of doing this with an Ubuntu 64 bit computer (for the client) talking to an ***nx*** server running on Windows7.



Not only does the [***-ping***] verify the connection, but the timing statistics get archived on the server log. For the above example, if one notices closely, one can see that the client's clock is 'behind' that of the server's clock. How does one know this? Since the ping packet will archive all the time's that were sent and received, one can see that the round trip delay from the client -> server -> client was about 53 milliseconds (difference of 01:44:39.850 and 01:44:39.903). Using half of this difference should approximately equate to the transit time in one direction or 26.5 msec. However, the difference in timestamps from when the client sent the packet (using the client clock) and when the server received the packet (using the server clock) is over 3 minutes. Anything that is significant in time difference should be noted, since any artifacts extracted from a client computer is relative to that specific client's computer clock. While this is a contrived example where we manipulated the clock on one system, it makes the point, that when taking artifacts from one computer and comparing them to another computer, one needs to be able to synchronize the time between artifacts from different machines.

5 Sending data to the server

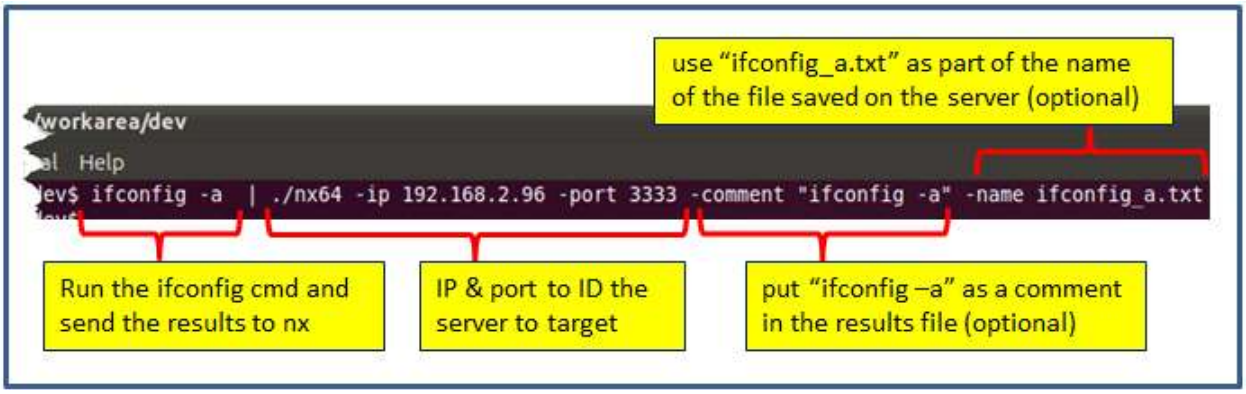
There are a number of ways to send artifacts from a client computer to the server. All of the options are enumerated in the menu displayed from the command prompt, shown below:

```
Options for client
   -name "filename that server uses to store results"
   -comment "text to track w/ output"
   -binary           = adds no formatting to the output
   -copyfile <file>  = file to send to the server
   -ping             = checks on server and displays time stats
   -quiet            = don't echo what is being sent
   -cust_id <name>   = associate nx transactions w/ this client label
   -host_id <name>   = associate nx transactions w/ this host label
   -pipe             = use stdin to pipe in cmds
```

The most basic command, which is not listed in the menu, but implied, is the ability to take any console output from some other tool and relay it to the server. Other options, combined with this basic capability, allow one to add documentation for each dataset that is transmitted. This includes: (a) comments and (b) a name to be used as part of the filename of the data archive.

As an example, one can send the network configuration as described by *ifconfig* (on Linux). As part of the command, one can annotate a comment to help the examiner remember any context

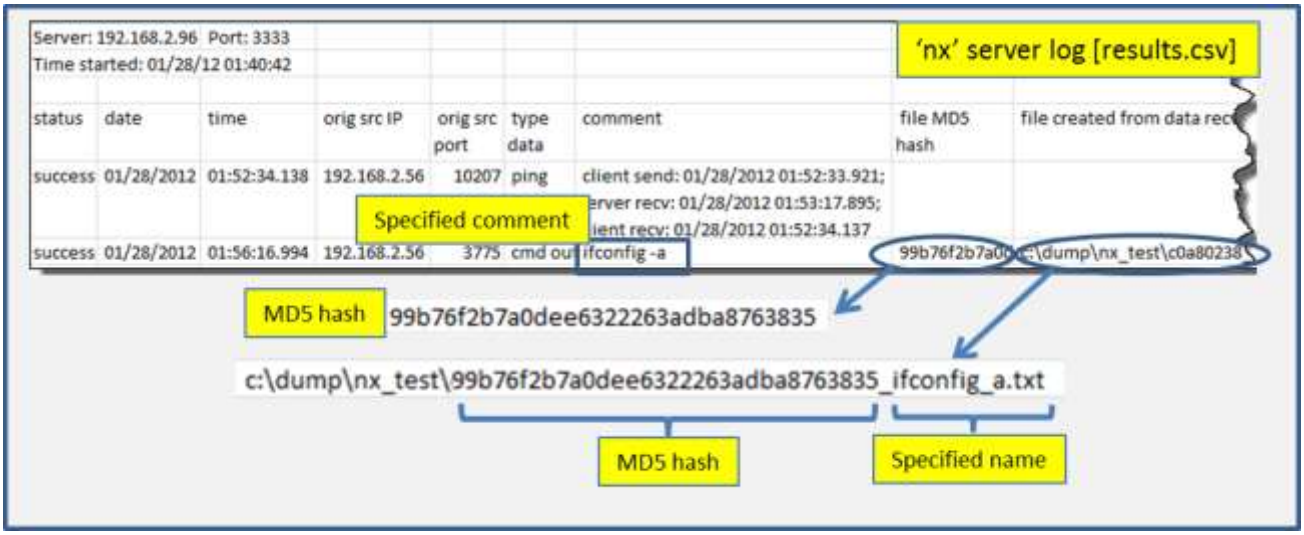
information at the time of the collection. One can also specify a name, to be used as part of the filename that is saved on the server end.



On the server end, all metadata information is archived within the server log. This log is named '<*start_date_time*>*_results.csv*'. From the extension, one can see, it is formatted as a Comma Separated Value (csv) file. This format was chosen since it is ubiquitous across various platform spreadsheet applications. The only disadvantage with the csv format, however, is that some filenames contain commas, since the comma is a valid filename character on Windows. ***nx*** handles this by converting every comma in the filename to another character when putting it into the log file. This is something to be aware of.

Below is a snapshot of the log file. From it, one can see the type of metadata recorded per transaction, such as: status, date/time, source IP/port, data type, etc. For those transactions that created a file in the archive directory, an MD5 hash will be computed and documented here. Furthermore, the name of the archive file itself will incorporate the md5 hash and any name that was requested to be used. While this makes for long filenames, it solves the problem of overwriting another file with the same filename and ensures the files stored are unique.

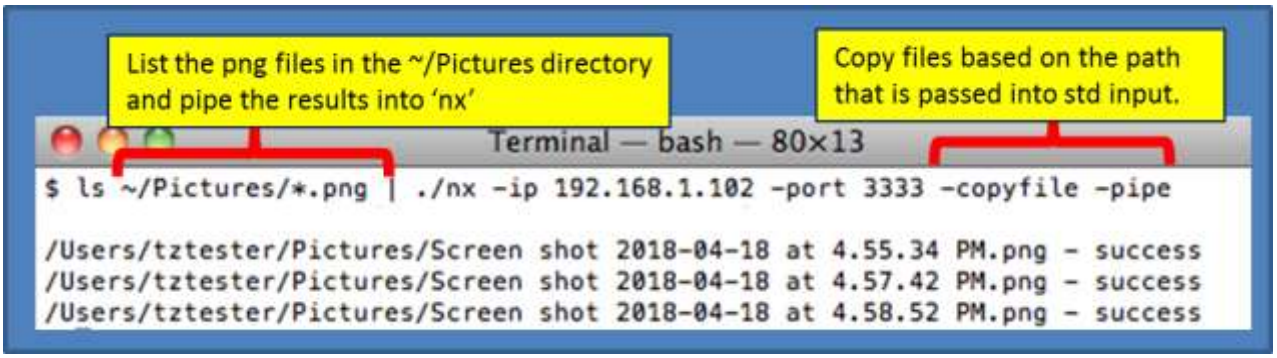
Looking at the last transaction, one can see the results of the *'ifconfig'* command. Note, the comment, specified by the ***-comment*** option, was included into the log entry. Also, the name, specified by the ***-name*** option, was appended to the end of the filename of the archived file.



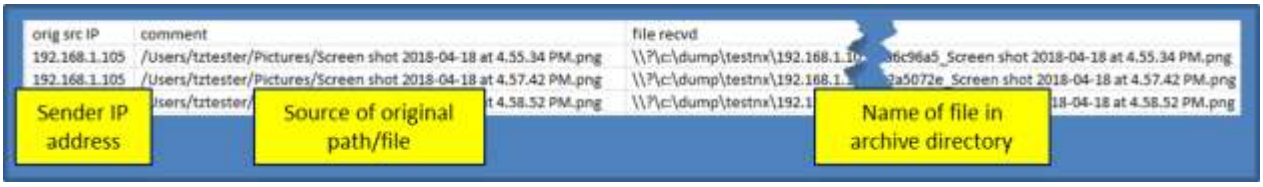
Other commands that send data include: (a) ***-comment <comment to send>***, (b) ***-copyfile <path/file>***, and (c) ***-copyfile -pipe***, which is a specialized version of ***-copyfile*** that receives its path/filenames from standard input.

The first two are self-explanatory; the latter two are better shown with examples. For variety, we chose to use macOS as the client for these examples.

The first example demonstrates how to pipe in a listing of the PNG files from a certain directory into ***nx*** for processing. Since we specified the ***-copyfile*** along with the ***-pipe*** option as part of the command, ***nx*** interprets each path/file passed in as a target file to copy.



Other implicit behavior, as can be seen from the results log file, include: (a) the name of the file is used as the archive name and (b) the path of the file is included in the comments section of the server log. The latter piece of metadata can be rather important when doing analysis later in the investigation cycle.



One point to mention on the client source IP address that gets archived in the log: this IP address is extracted from the client side and packaged as part of the data that gets sent to the ***nx*** server. This means that if a client is behind a nat'd firewall, the actual IP address of the client machine still shows up in the log file as opposed to the firewall's IP address.

5.1 Server's Log File/Format

The fields for the nx server side log file are defined as follows:

Field	Description
Status	Whether the transmission from the client was successful or not
Server Date/Time	Timestamp of the server in UTC (relative to the server's clock) for the client transmission
Target Date/Time	Timestamp of the client in UTC (relative to the client's clock) during the contact with a request to send data
Target	Client's ID. Defaults to the client's IP address, but can be changed using ***-cust_id*** option
Host	Client's host identifier. Defaults to the client's computer name, but can be changed using ***-host_id*** option
Admin	Whether the client is running with admin privs or not
Orig Src IP	Client's IP address used to send data to the server
Orig Src Port	Client's Port used to send data to the server
Type Data	Whether a file was copied or a comment was sent, etc
Comment	Field defined by client to use as necessary

File MD5 Hash	Computed hash of the file sent from the client to the server
File Recvd	Used to affect the formatting of the server log file. Output the date/time as 1 field vice 2 for csv option
Size	Size of the data received from the client
adate	Access timestamp in UTC of the original file sent
cdate	Create timestamp in UTC of the original file sent
mdate	Modify timestamp in UTC of the original file sent

5.2 Volume Shadow Copies

For starters, to access Volume Shadow copies, one needs to be running with administrator privileges. Also, Volume Shadow copies, as is discussed here, only applies to Windows Vista, Win7, Win8 and beyond. It does not apply to Windows XP.

To make it easier with the syntax, we've built in some shortcut syntax to access a specified Volume Shadow copy, via the ***%vss%*** keyword. This internally gets expanded into *\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy*. Thus, to access index 1 of the volume shadow copy, one would prepend the keyword and index, like so, ***%vss%1*** to the normal path of the hive. For example, to access a user hive located in the *testuser* account from the *HarddiskVolumeShadowCopy1*, the following syntax can be used:

```
nx64 -ip 192.168.2.96 -port 3333 -copy %vss%3\Users\testuser\ntuser.dat
```

To determine which indexes are available from the various Volume Shadows, one can use the Windows built-in utility ***vssadmin***, as follows:

```
vssadmin  list  shadows
```

To filter some of the extraneous detail, type

```
vssadmin  list shadows |  find /i  "volume"
```

While the amount of data can be voluminous, the keywords one needs to look for are names that look like this:

```
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
  . . .
```

From the above, notice the number after the word *HarddiskvolumeShadowCopy*. It is this number that is appended to the ***%vss%*** keyword.

5.3 Using other *TZWorks* tools and *nx* to Transport Locked Files

If one has access to the TZWorks bundle license, then you can use the copy facility in ***ntfscopy*** to create a temporary file and then use ***nx*** to transport the data in that temporary file to the receiving forensics workstation.

As an example we wish to copy various registry hives. These include the user hive: *ntuser.dat* and *usrclass.dat* and the system hives. The steps to do this are shown in the following batch file.

```
rem - make copies of any locked files
vssenum64 -dir c:\users -filter "ntuser.dat|usrclass.dat" -level 10 | ntfscopy64 -pipe -dst my_temp -md5
vssenum64 -dir %systemroot%\system32\config -filter "software|system|security|sam|components" | ntfscopy64 -pipe -dst my_temp
vssenum64 -dir %systemroot%\AppCompat\Programs -filter "amcache.hve" | ntfscopy64 -pipe -dst my_temp

rem - send the contents of the my_temp directory to the nx server
dir my_temp\* /b /s | nx -ip 127.0.0.1 -port 2222 -copyfile -pipe -comment "used ntfscopy to copy hives"

rem - clear out the temp directory
rmdir /s /q my_temp
```

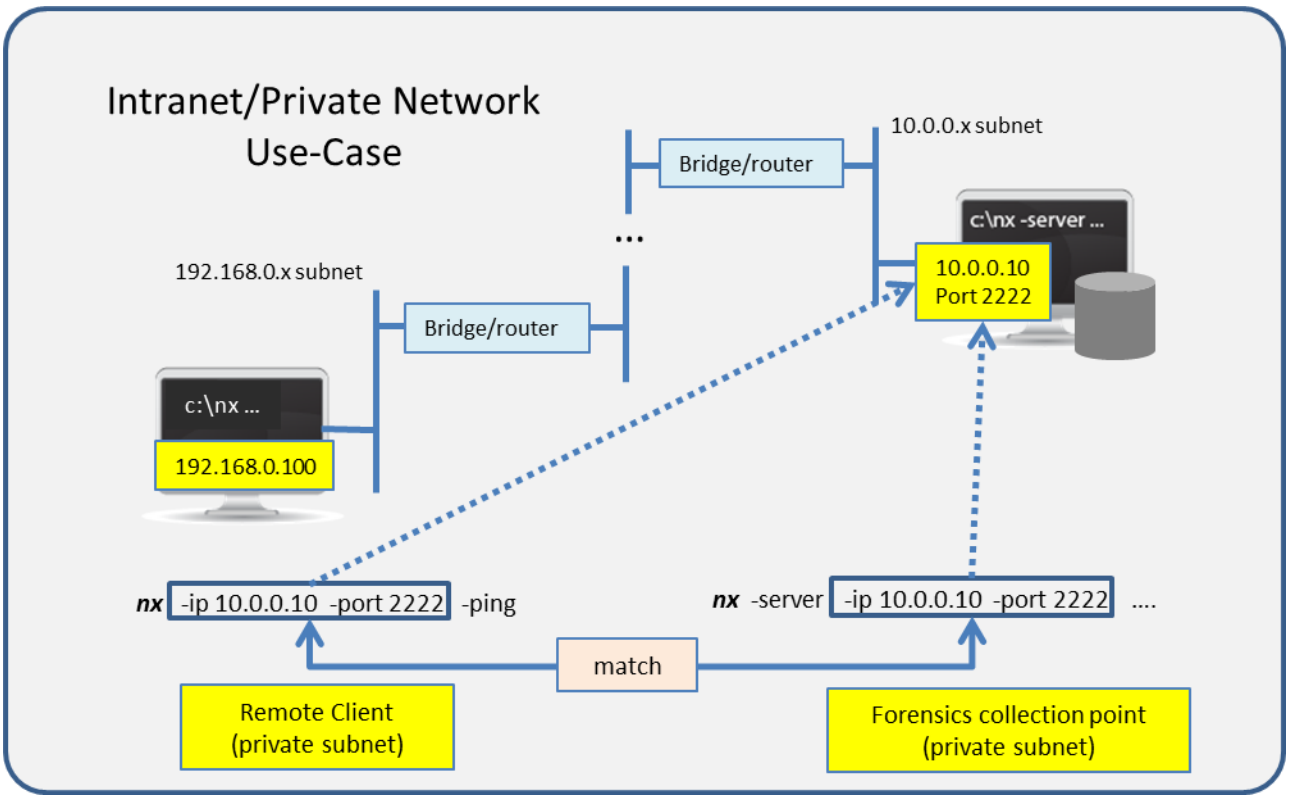
6 Various Use-Case for Transferring Data

The communications between the ***nx*** client and server is what is called peer to peer communications. This is defined to be communications between two nodes in contrast to multi-node or broadcast communication. In this case, ***nx*** (the client) must specify the IP address of the ***nx*** server and communications is directly between them. One must assume routing will occur, but it should not impact the communication, since the ***nx*** client explicitly specifies the ***nx*** IP address of the server when running. For peer to peer communication, no domain credentials are required to be set up for enterprise networks. As long as the client can communicate to the ***nx*** server IP address, without being impacted by firewalls or other network devices that can block IP traffic, the communications should be seamless.

There are a number of use-cases in setting up the client and server, depending on the network architecture. This section goes over some of the common ones.

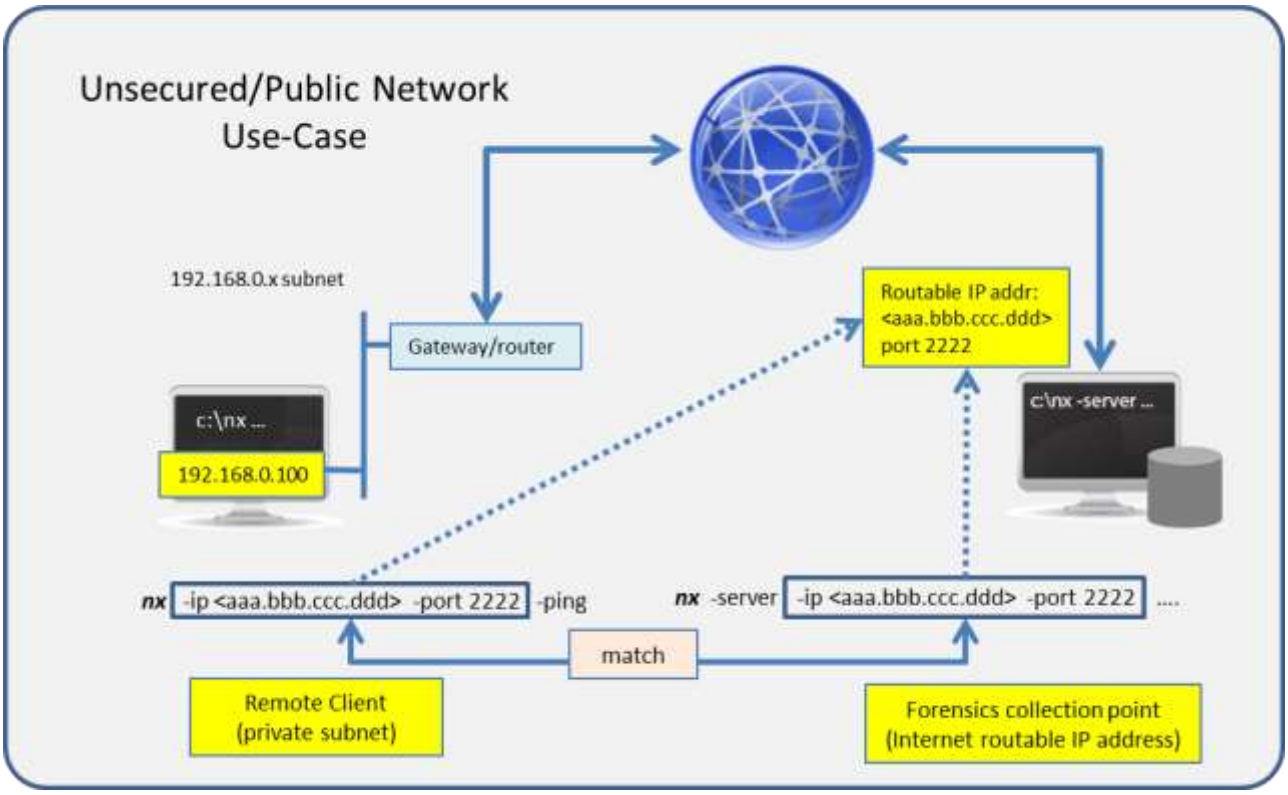
6.1 Normal Private Intranet for both *minx* and *nx*

For the case where the target endpoint and the forensics workstation are on the same private Intranet, the IP and port specified on the command line match as shown below. In the example, the target endpoint (where the data is being collected from, eg. ***nx***) has an IP address of 192.168.0.100 and the forensics workstation (running ***nx*** in server mode) is at IP address 10.0.0.10. The ***nx*** service is using port 2222, in this example.



6.2 Sending Data from a Private Intranet to a Public Internet Address

If you have a situation where your forensics collection point is accessible from the Internet, then the setup is not any different than the case discussed above. Both the IP address and port specified in the ***nx*** client setup is the same and will match whatever the ***nx*** server has specified.

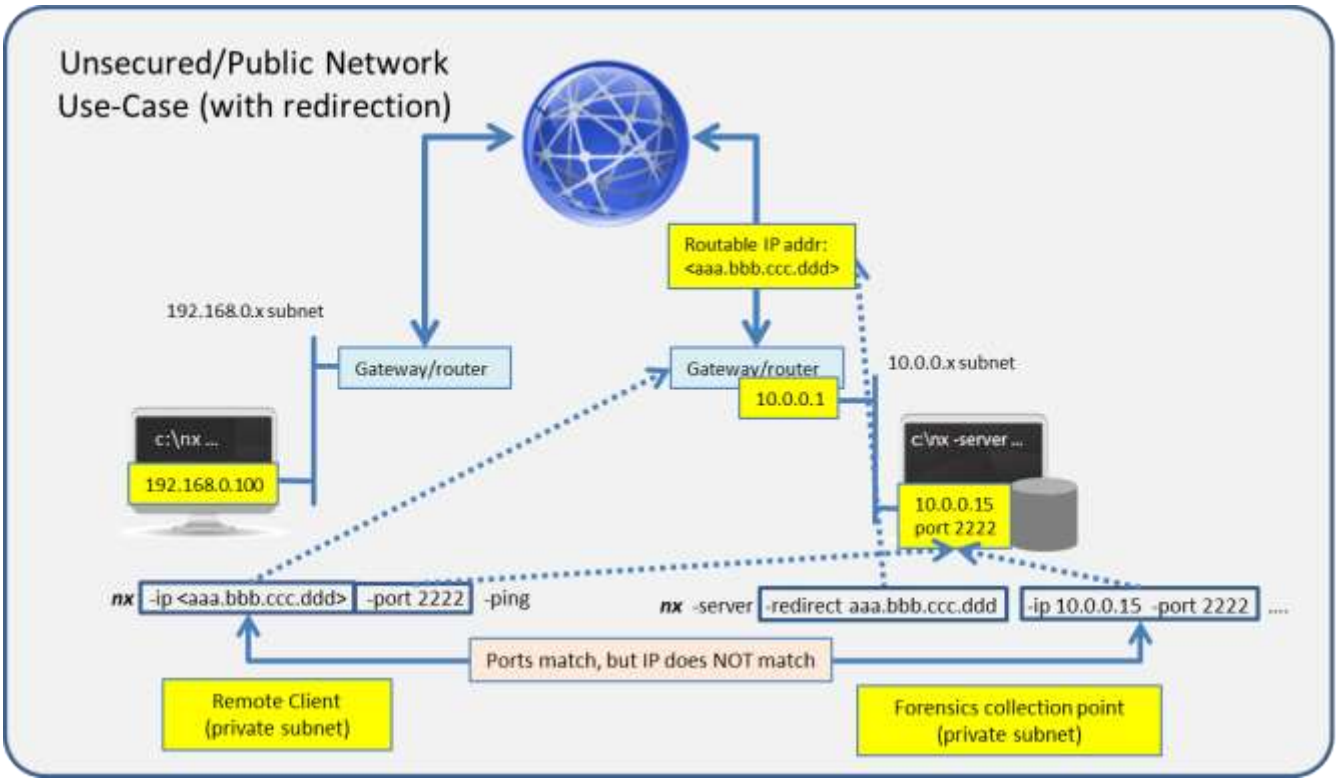


6.3 Using TCP/IP Redirection

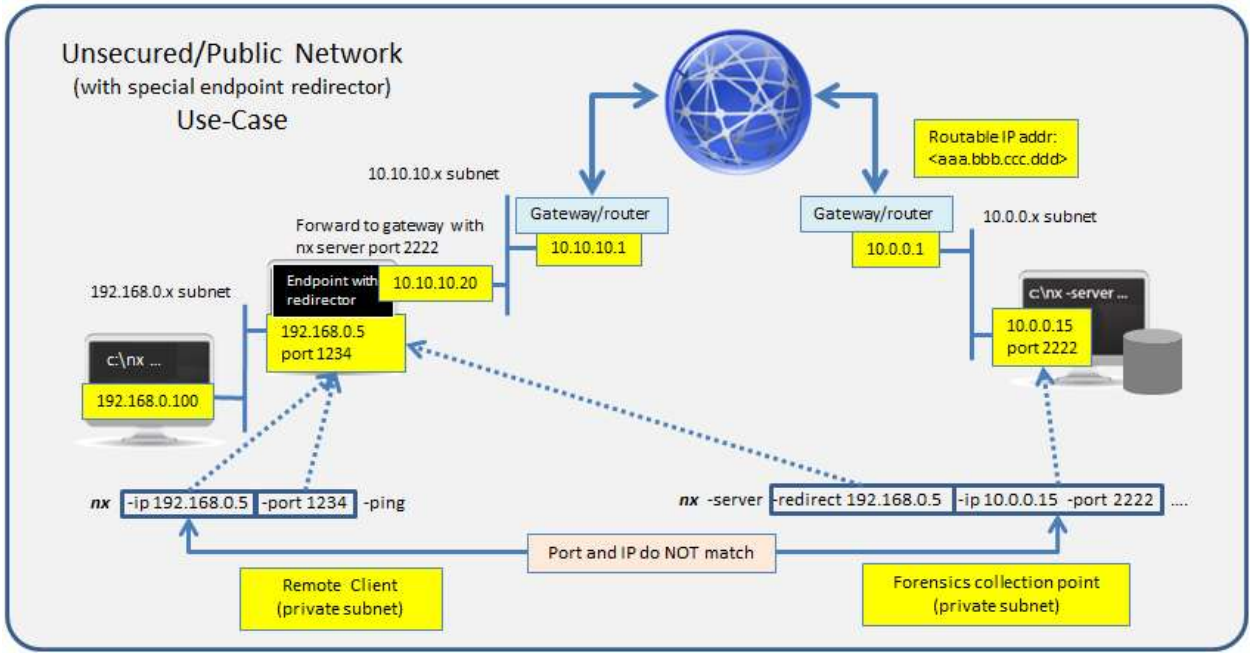
These examples require more advanced knowledge and familiarity with setting up redirectors and VPNs.

For more complex cases, where one needs to use a redirector to send the packets to the forensics collection point, two cases are shown below. The first one is similar to the previous case with a variation in that the forensics workstation is protected by a firewall/router. Since the IP address of the firewall/router (on the ***nx*** service side) is the only thing addressable from an *Internet* standpoint then the ***nx*** client must use this address. The firewall/router, needs to be setup to redirect this traffic on the ***nx*** port (eg. 2222 in this example) to the forensics workstation.

Normally, for peer to peer communications this should work without any other changes; however, for the ***nx*** architecture, one must specify this redirection on the ***nx*** server side. To be clear, the redirection only needs to be specified from the ***nx*** perspective (as opposed to any other case that may happen unbeknownst to user). To do this, when one sets up the ***nx*** server, one specifies the additional option ***-redirect <IP address that minx will be specifying to send data to the server>***. Below is a diagram with command line arguments for both ***minx*** and ***nx*** that shows an example of how to set this up.



The second variation of the redirection use-case is when the target endpoint that is sending data is restricted from doing so because of a firewall or network blocking device. There are a number of solutions that one can implement. The one shown here is to set up an intermediary computer on that is accessible to the target endpoint running the ***nx*** client. The intermediary computer will employ a redirection service for the internal private network to access the outside network. In this example, the internal private network is on subnet 192.168.0.x. The intermediary computer is connected to both this private network as well as to the 10.10.10.x subnet, so it can effectively route/redirect the any packets received on port 1234 on IP address 192.168.0.5 to port 2222 on IP address aaa.bbb.ccc.ddd, as well as handle the reverse redirection. From the ***nx*** client's perspective, it is sending data to IP address 192.168.0.5, port 1234. Doing so will get to the forensics workstation running the ***nx*** service. For the ***nx*** service to accept the ***nx***'s client packets, however, it must be told that that a ***-redirect*** was occurring on 192.168.0.5. These command arguments are shown below.



7 Available Options

7.1 Server Options

Option	Description
-server	Server option that specifies this session as the server. Without this option, the session is run as a client.
-ip	IP address of the server (used for both client and server sessions). The syntax is: ***-ip <server IP address>***.
-port	Port address of the server (used for both client and server sessions). The syntax is: ***-port <server port address>***.
-dir	Server required option to specify what directory to dump the data coming from the client. (not used for the client). The syntax is: ***-dir <results directory>***.
-key	Client and Server option. If network encryption is required, this is the symmetric key to use. If used, the same key is needed for both the client and server. The syntax is: ***-key <password phrase>***.
-redirect	Server option. Tells the server that the client will initially route packets to

	the TCP/IP address specified. Otherwise, without this option, the server will assume client is sending packets directly to server IPv4 address. Without using this option, if a redirector is used between client and server, server will ignore packets that client targets to a redirector IP address. This option is only available with site licenses. The syntax is: ***-redirect <IPv4 address of where the client sends data to>***
-scripts	Server option that tells the server which folder has the configuration files (scripts) that can be queried by the client (only useful if using a ***minx*** client).
-no_whitespace	Used to affect the formatting of the server log file. Used in conjunction with *-csv* option to remove any whitespace between the field value and the CSV separator.
-csv_separator	Used to affect the formatting of the server log file. Used in conjunction with the *-csv* option to change the CSV separator from the default comma to something else. Syntax is *-csv_separator "|"* to change the CSV separator to the pipe character. To use the tab as a separator, one can use the *-csv_separator "tab"* OR *-csv_separator "\t"* options.
-dateformat	Output the date using the specified format. Default behavior is *-dateformat "yyyy-mm-dd"*. Using this option allows one to adjust the format to mm/dd/yy, dd/mm/yy, etc. The restriction with this option is the forward slash (/) or dash (-) symbol needs to separate month, day and year and the month is in digit (1-12) form versus abbreviated name form.
-timeformat	Used to affect the formatting of the server log file. Output the time using the specified format. Default behavior is *-timeformat "hh:mm:ss.xxx"* One can adjust the format to microseconds, via *"hh:mm:ss.xxxxxx"* or nanoseconds, via *"hh:mm:ss.xxxxxxxxx"*, or no fractional seconds, via *"hh:mm:ss"*. The restrictions with this option is that a colon (:) symbol needs to separate hours, minutes and seconds, a period (.) symbol needs to separate the seconds and fractional seconds, and the repeating symbol 'x' is used to represent number of fractional seconds. (Note: the fractional seconds applies only to those time formats that have the appropriate precision available. The Windows internal filetime has, for example, 100 nsec unit precision available. The DOS time format and the UNIX 'time_t' format, however, have no fractional seconds). Some of the times represented by this tool may use a time format without fractional seconds and therefore will not show a greater precision beyond seconds

	when using this option.
-pair_datetime	Used to affect the formatting of the server log file. Output the date/time as 1 field vice 2 for csv option

7.2 Client Options

Option	Description
-ip	IP address of the server (used for both client and server sessions). The syntax is: ***-ip <server IP address>***.
-port	Port address of the server (used for both client and server sessions). The syntax is: ***-port <server port address>***.
-key	Client and Server option. If network encryption is required, this is the symmetric key to use. If used, the same key is needed for both the client and server. The syntax is: ***-key <password phrase>***.
-name	Option for the client to request the server to use the specified filename to archive the results. The syntax is: ***-name <filename>***.
-comment	Option for the client to send to the server to make a comment in the logs stored on the server session side. The syntax is: ***-comment <"phrase">***.
-binary	Option for the client that says not to add any formatting to the data that is read from the target computer and sent to the server.
-copyfile	Option for the client to copy a specified file and send it to the server. The syntax is: ***-copyfile <file to copy>***. Alternatively, this option can be used with the ***-pipe*** option to copy files that are listed via standard input.
-ping	Option for the client to check whether the server is up and synchonized time statistics. These statistics are recorded in the server side logs.
-quiet	Option for the client to not echo to the user (client side) what is being sent to the server.
-cust_id	Option for the client to identify itself to the server with a unique client label. This label is associated with all the file sent by this client. The syntax

	is: ***-cust_id <label>***.
-host_id	Option for the client to identify itself to the server with a unique host label. This label is associated with all the file sent by this client. The syntax is: ***-host_id <label>***.
-pipe	Used to pipe files or commands into the tool via STDIN (standard input). Each file passed in is parsed in sequence. Below are some examples: ipconfig /all | nx -pipe -ip <serv ip> -port <port #> -name ipconfig_all.txt netstat -anob | nx -pipe -ip <serv ip> -port <port #> -name netstat_anob.txt wmic process list full | nx -pipe -ip <serv ip> -port <port #> -name wmic_proc.txt dir c:*.pdf /b /s | nx -pipe -ip <serv ip> -port <port #> -copyfile
-filter	Filters data passed in via STDIN via the ***-pipe*** option. The syntax is ***-filter <"*.ext | *partialname* | ...">***. The wildcard character '*' is restricted to either before the name or after the name.

8 Zlib Dependency

nx makes the of the *zlib* library. If one is unfamiliar with zlib, the official *zlib* website is https://zlib.net. It has documentation and details on everything one would ever want to know. The *zlib* library is statically linked into ***nx*** binary. What this means is the tool is standalone and does not require any external *zlib* shared libraries.

9 Authentication and the License File

This tool has authentication built into the binary. The primary authentication mechanism is the digital X509 code signing certificate embedded into the binary (Windows and macOS).

The other mechanism is the runtime authentication, which applies to all the versions of the tools (Windows, Linux and macOS). The runtime authentication ensures that the tool has a valid license. The license needs to be in the same directory of the tool for it to authenticate. Furthermore, any modification to the license, either to its name or contents, will invalidate the license.

9.1 *Limited* versus *Demo* versus *Full* in the tool's Output Banner

The tools from *TZWorks* will output header information about the tool's version and whether it is running in *limited, demo* or *full* mode. This is directly related to what version of a license the tool authenticates with. The *limited* and *demo* keywords indicates some functionality of the tool is not available, and the *full* keyword indicates all the functionality is available. The lacking functionality in the *limited* or *demo* versions may mean one or all of the following: (a) certain options may not be available, (b) certain data may not be outputted in the parsed results, and (c) the license has a finite lifetime before expiring.

10 References

1. ***zlib*** library version 1.2.8, April 28th, 2013, by Jean-loup Gailly and Mark Adler
2. Gary Wright and Richard Stevens, TCP/IP Illustrated, Vol 2, The Implementation, 1995
3. Douglas Comer & David Stevens, Networking with TCP/IP Client-Server Programming and applications, 2001
4. Jones, Bejtlick, Rose, Real Digital Forensics, Computer Security and Incident Response, 2006