Sep 02 2023

NETMANAGEIT

Intelligence Report

Infamous Chisel Malware Analysis Report

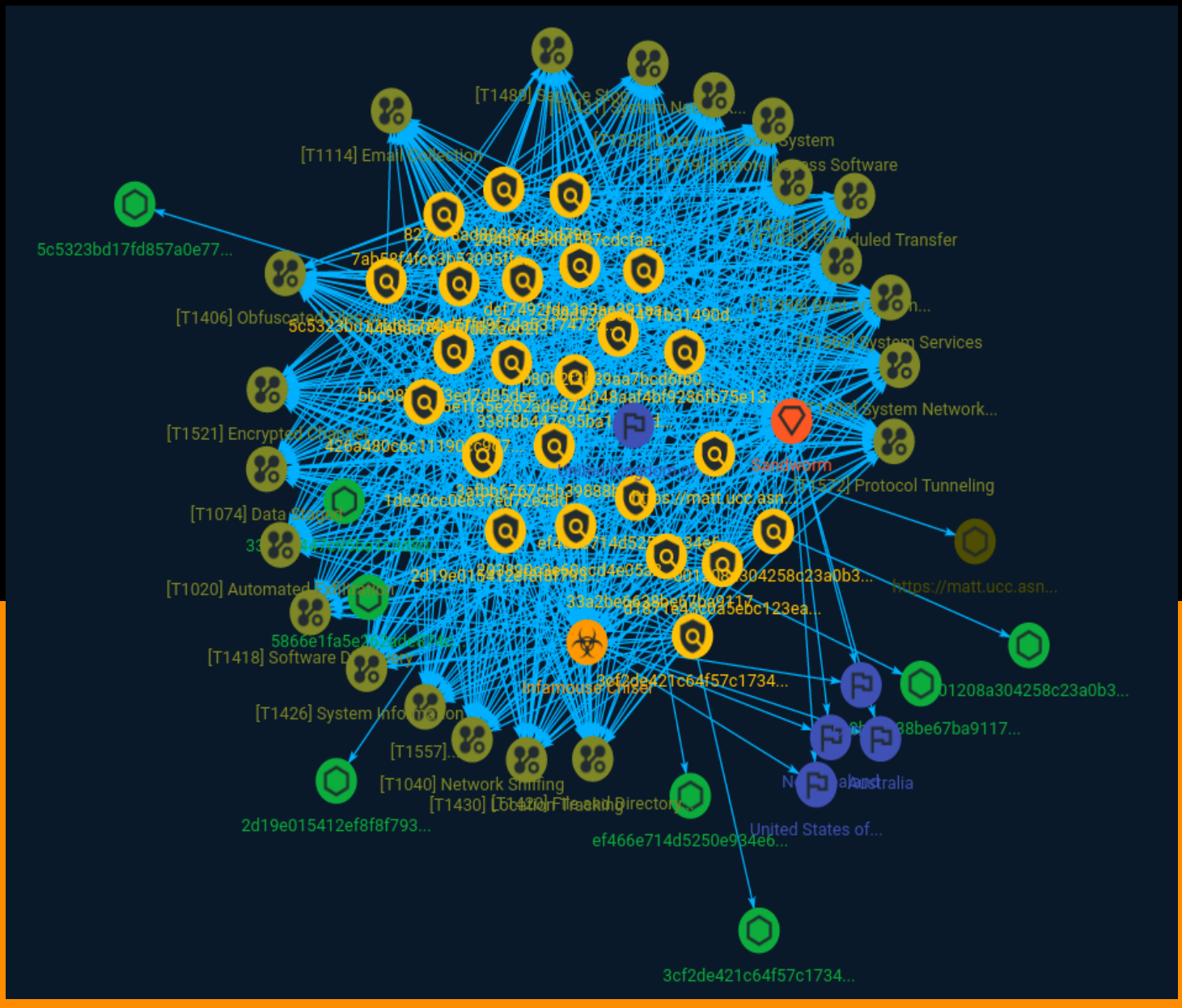

Table of contents

Overview

Entities

Observables

External References

Overview

Description

The Infamous Chisel malware is being investigated by the United States National Security Agency (CISA), which is working with the Australian National Cyber Security Centre (NCSC), New Zealand and Australia.

Confidence

This value represents the confidence in the correctness of the data contained within this report.

15 / 100

Attack-Pattern

Name
T1473
ID
T1473
Name
Encrypted Channel
ID
T1521
Description
Adversaries may explicitly employ a known encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Despite the use of a secure algorithm, these implementations may be vulnerable to reverse engineering if necessary secret keys are encoded and/or generated within malware samples/configuration files.
Name
Boot or Logon Initialization Scripts

ID

T1398

Description

Adversaries may use scripts automatically executed at boot or logon initialization to establish persistence. Initialization scripts are part of the underlying operating system and are not accessible to the user unless the device has been rooted or jailbroken.

Name

Adversary-in-the-Middle

ID

T1557

Description

Adversaries may attempt to position themselves between two or more networked devices using an adversary-in-the-middle (AiTM) technique to support follow-on behaviors such as [Network Sniffing](https://attack.mitre.org/techniques/T1040) or [Transmitted Data Manipulation](https://attack.mitre.org/techniques/T1565/002). By abusing features of common networking protocols that can determine the flow of network traffic (e.g. ARP, DNS, LLMNR, etc.), adversaries may force a device to communicate through an adversary controlled system so they can collect information or perform additional actions.(Citation: Rapid7 MiTM Basics) For example, adversaries may manipulate victim DNS settings to enable other malicious activities such as preventing/redirecting users from accessing legitimate sites and/or pushing additional malware.(Citation: ttint_rat)(Citation: dns_changer_trojans)(Citation: ad_blocker_with_miner) Adversaries may also manipulate DNS and leverage their position in order to intercept user credentials and session cookies. (Citation: volexity_0day_sophos_FW) [Downgrade Attack](https://attack.mitre.org/techniques/T1562/010)s can also be used to establish an AiTM position, such as by negotiating a less secure, deprecated, or weaker version of communication protocol (SSL/TLS) or encryption algorithm.(Citation: mitm_tls_downgrade_att)(Citation: taxonomy_downgrade_att_tls)(Citation: tlseminar_downgrade_att) Adversaries may also leverage the AiTM position to attempt to monitor and/or modify traffic, such as in [Transmitted Data Manipulation](https://attack.mitre.org/techniques/T1565/002).

Adversaries can setup a position similar to AiTM to prevent traffic from flowing to the appropriate destination, potentially to [Impair Defenses](https://attack.mitre.org/techniques/T1562) and/or in support of a [Network Denial of Service](https://attack.mitre.org/techniques/T1498).

Name

System Network Connections Discovery

ID

T1421

Description

Adversaries may attempt to get a listing of network connections to or from the compromised device they are currently accessing or from remote systems by querying for information over the network. This is typically accomplished by utilizing device APIs to collect information about nearby networks, such as Wi-Fi, Bluetooth, and cellular tower connections. On Android, this can be done by querying the respective APIs: * `WifiInfo` for information about the current Wi-Fi connection, as well as nearby Wi-Fi networks. Querying the `WiFiInfo` API requires the application to hold the `ACCESS_FINE_LOCATION` permission. * `BluetoothAdapter` for information about Bluetooth devices, which also requires the application to hold several permissions granted by the user at runtime. * For Android versions prior to Q, applications can use the `TelephonyManager.getNeighboringCellInfo()` method. For Q and later, applications can use the `TelephonyManager.getAllCellInfo()` method. Both methods require the application hold the `ACCESS_FINE_LOCATION` permission.

Name

File and Directory Discovery

ID

T1420

Description

Adversaries may enumerate files and directories or search in specific device locations for desired information within a filesystem. Adversaries may use the information from [File and Directory Discovery](https://attack.mitre.org/techniques/T1420) during automated discovery to shape follow-on behaviors, including deciding if the adversary should fully infect the target and/or attempt specific actions. On Android, Linux file permissions and SELinux policies typically stringently restrict what can be accessed by apps without taking advantage of a privilege escalation exploit. The contents of the external storage directory are generally visible, which could present concerns if sensitive data is inappropriately stored there. iOS's security architecture generally restricts the ability to perform any type of [File and Directory Discovery](https://attack.mitre.org/techniques/T1420) without use of escalated privileges.

Name

Data Staged

ID

T1074

Description

Adversaries may stage collected data in a central location or directory prior to Exfiltration. Data may be kept in separate files or combined into one file through techniques such as [Archive Collected Data](https://attack.mitre.org/techniques/T1560). Interactive command shells may be used, and common functionality within [cmd](https://attack.mitre.org/software/S0106) and bash may be used to copy data into a staging location.(Citation: PWC Cloud Hopper April 2017) In cloud environments, adversaries may stage data within a particular instance or virtual machine before exfiltration. An adversary may [Create Cloud Instance](https://attack.mitre.org/techniques/T1578/002) and stage data in that instance. (Citation: Mandiant M-Trends 2020) Adversaries may choose to stage data from a victim network in a centralized location prior to Exfiltration to minimize the number of connections made to their C2 server and better evade detection.

Name

Scheduled Transfer

ID

T1029

Description

Adversaries may schedule data exfiltration to be performed only at certain times of day or at certain intervals. This could be done to blend traffic patterns with normal activity or availability. When scheduled exfiltration is used, other exfiltration techniques likely apply as well to transfer the information out of the network, such as [Exfiltration Over C2 Channel](https://attack.mitre.org/techniques/T1041) or [Exfiltration Over Alternative Protocol](https://attack.mitre.org/techniques/T1048).

Name

Network Sniffing

ID

T1040

Description

Adversaries may sniff network traffic to capture information about an environment, including authentication material passed over the network. Network sniffing refers to using the network interface on a system to monitor or capture information sent over a wired or wireless connection. An adversary may place a network interface into promiscuous mode to passively access data in transit over the network, or use span ports to capture a larger amount of data. Data captured via this technique may include user credentials, especially those sent over an insecure, unencrypted protocol. Techniques for name service resolution poisoning, such as [LLMNR/NBT-NS Poisoning and SMB Relay](https://attack.mitre.org/techniques/T1557/001), can also be used to capture credentials to websites, proxies, and internal systems by redirecting traffic to an adversary. Network sniffing may also reveal configuration details, such as running services, version numbers, and other network characteristics (e.g. IP addresses, hostnames, VLAN IDs) necessary for subsequent Lateral Movement and/or Defense Evasion activities. In cloud-based environments, adversaries may still be able to use traffic mirroring services to sniff network traffic from virtual machines. For example, AWS Traffic Mirroring, GCP Packet Mirroring, and Azure vTap allow users to define specified instances to collect traffic from

and specified targets to send collected traffic to.(Citation: AWS Traffic Mirroring)(Citation: GCP Packet Mirroring)(Citation: Azure Virtual Network TAP) Often, much of this traffic will be in cleartext due to the use of TLS termination at the load balancer level to reduce the strain of encrypting and decrypting traffic.(Citation: Rhino Security Labs AWS VPC Traffic Mirroring)(Citation: SpecterOps AWS Traffic Mirroring) The adversary can then use exfiltration techniques such as Transfer Data to Cloud Account in order to access the sniffed traffic.(Citation: Rhino Security Labs AWS VPC Traffic Mirroring) On network devices, adversaries may perform network captures using [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) commands such as `monitor capture`.(Citation: US-CERT-TA18-106A)(Citation: capture_embedded_packet_on_software)

Name

Email Collection

ID

T1114

Description

Adversaries may target user email to collect sensitive information. Emails may contain sensitive data, including trade secrets or personal information, that can prove valuable to adversaries. Adversaries can collect or forward email from mail servers or clients.

Name

System Network Configuration Discovery

ID

T1422

Description

Adversaries may look for details about the network configuration and settings, such as IP and/or MAC addresses, of operating systems they access or through information discovery of remote systems. On Android, details of onboard network interfaces are accessible to

apps through the `java.net.NetworkInterface` class.(Citation: NetworkInterface) Previously, the Android `TelephonyManager` class could be used to gather telephony-related device identifiers, information such as the IMSI, IMEI, and phone number. However, starting with Android 10, only preloaded, carrier, the default SMS, or device and profile owner applications can access the telephony-related device identifiers.(Citation: TelephonyManager) On iOS, gathering network configuration information is not possible without root access. Adversaries may use the information from [System Network Configuration Discovery](https://attack.mitre.org/techniques/T1422) during automated discovery to shape follow-on behaviors, including determining certain access within the target network and what actions to do next.

Name

Software Discovery

ID

T1418

Description

Adversaries may attempt to get a listing of applications that are installed on a device. Adversaries may use the information from [Software Discovery](https://attack.mitre.org/techniques/T1418) during automated discovery to shape follow-on behaviors, including whether or not to fully infect the target and/or attempts specific actions. Adversaries may attempt to enumerate applications for a variety of reasons, such as figuring out what security measures are present or to identify the presence of target applications.

Name

Data from Local System

ID

T1533

Description

Adversaries may search local system sources, such as file systems or local databases, to find files of interest and sensitive data prior to exfiltration. Access to local system data, which includes information stored by the operating system, often requires escalated privileges. Examples of local system data include authentication tokens, the device keyboard cache, Wi-Fi passwords, and photos. On Android, adversaries may also attempt to access files from external storage which may require additional storage-related permissions.

Name

Location Tracking

ID

T1430

Description

Adversaries may track a device's physical location through use of standard operating system APIs via malicious or exploited applications on the compromised device. On Android, applications holding the `ACCESS_COAURSE_LOCATION` or `ACCESS_FINE_LOCATION` permissions provide access to the device's physical location. On Android 10 and up, declaration of the `ACCESS_BACKGROUND_LOCATION` permission in an application's manifest will allow applications to request location access even when the application is running in the background.(Citation: Android Request Location Permissions) Some adversaries have utilized integration of Baidu map services to retrieve geographical location once the location access permissions had been obtained.(Citation: PaloAlto-SpyDealer)(Citation: Palo Alto HenBox) On iOS, applications must include the `NSLocationWhenInUseUsageDescription`, `NSLocationAlwaysAndWhenInUseUsageDescription`, and/or `NSLocationAlwaysUsageDescription` keys in their `Info.plist` file depending on the extent of requested access to location information.(Citation: Apple Requesting Authorization for Location Services) On iOS 8.0 and up, applications call `requestWhenInUseAuthorization()` to request access to location information when the application is in use or `requestAlwaysAuthorization()` to request access to location information regardless of whether the application is in use. With elevated privileges, an adversary may be able to access location data without explicit user consent with the `com.apple.locationd.preauthorized` entitlement key.(Citation: Google Project Zero Insomnia)

Name

Obfuscated Files or Information

ID

T1406

Description

Adversaries may attempt to make a payload or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the device or in transit. This is common behavior that can be used across different platforms and the network to evade defenses. Payloads may be compressed, archived, or encrypted in order to avoid detection. These payloads may be used during Initial Access or later to mitigate detection. Portions of files can also be encoded to hide the plaintext strings that would otherwise help defenders with discovery. Payloads may also be split into separate, seemingly benign files that only reveal malicious functionality when reassembled.(Citation: Microsoft MalLockerB)

Name

System Services

ID

T1569

Description

Adversaries may abuse system services or daemons to execute commands or programs. Adversaries can execute malicious content by interacting with or creating services either locally or remotely. Many services are set to run at boot, which can aid in achieving persistence ([Create or Modify System Process](https://attack.mitre.org/techniques/T1543)), but adversaries can also abuse services for one-time or temporary execution.

Name

System Information Discovery

ID

T1426

Description

Adversaries may attempt to get detailed information about a device’s operating system and hardware, including versions, patches, and architecture. Adversaries may use the information from [System Information Discovery](https://attack.mitre.org/techniques/T1426) during automated discovery to shape follow-on behaviors, including whether or not to fully infects the target and/or attempts specific actions. On Android, much of this information is programmatically accessible to applications through the `android.os.Build` class. (Citation: Android-Build) iOS is much more restrictive with what information is visible to applications. Typically, applications will only be able to query the device model and which version of iOS it is running.

Name

Service Stop

ID

T1489

Description

Adversaries may stop or disable services on a system to render those services unavailable to legitimate users. Stopping critical services or processes can inhibit or stop response to an incident or aid in the adversary's overall objectives to cause damage to the environment.(Citation: Talos Olympic Destroyer 2018)(Citation: Novetta Blockbuster) Adversaries may accomplish this by disabling individual services of high importance to an organization, such as `MSExchangeIS`, which will make Exchange content inaccessible (Citation: Novetta Blockbuster). In some cases, adversaries may stop or disable many or all services to render systems unusable.(Citation: Talos Olympic Destroyer 2018) Services or processes may not allow for modification of their data stores while running. Adversaries may stop services or processes in order to conduct [Data Destruction](https://

attack.mitre.org/techniques/T1485) or [Data Encrypted for Impact](https://attack.mitre.org/techniques/T1486) on the data stores of services like Exchange and SQL Server.(Citation: SecureWorks WannaCry Analysis)

Name

Protocol Tunneling

ID

T1572

Description

Adversaries may tunnel network communications to and from a victim system within a separate protocol to avoid detection/network filtering and/or enable access to otherwise unreachable systems. Tunneling involves explicitly encapsulating a protocol within another. This behavior may conceal malicious traffic by blending in with existing traffic and/or provide an outer layer of encryption (similar to a VPN). Tunneling could also enable routing of network packets that would otherwise not reach their intended destination, such as SMB, RDP, or other traffic that would be filtered by network appliances or not routed over the Internet. There are various means to encapsulate a protocol within another protocol. For example, adversaries may perform SSH tunneling (also known as SSH port forwarding), which involves forwarding arbitrary data over an encrypted SSH tunnel. (Citation: SSH Tunneling) [Protocol Tunneling](https://attack.mitre.org/techniques/T1572) may also be abused by adversaries during [Dynamic Resolution](https://attack.mitre.org/techniques/T1568). Known as DNS over HTTPS (DoH), queries to resolve C2 infrastructure may be encapsulated within encrypted HTTPS packets.(Citation: BleepingComp Godlua JUL19) Adversaries may also leverage [Protocol Tunneling](https://attack.mitre.org/techniques/T1572) in conjunction with [Proxy](https://attack.mitre.org/techniques/T1090) and/or [Protocol Impersonation](https://attack.mitre.org/techniques/T1001/003) to further conceal C2 communications and infrastructure.

Name

Automated Exfiltration

ID

T1020

Description

Adversaries may exfiltrate data, such as sensitive documents, through the use of automated processing after being gathered during Collection. When automated exfiltration is used, other exfiltration techniques likely apply as well to transfer the information out of the network, such as [Exfiltration Over C2 Channel](https://attack.mitre.org/techniques/T1041) and [Exfiltration Over Alternative Protocol](https://attack.mitre.org/techniques/T1048).

Name

Remote Access Software

ID

T1219

Description

An adversary may use legitimate desktop support and remote access software, such as Team Viewer, AnyDesk, Go2Assist, LogMein, AmmyyAdmin, etc, to establish an interactive command and control channel to target systems within networks. These services are commonly used as legitimate technical support software, and may be allowed by application control within a target environment. Remote access tools like VNC, Ammyy, and Teamviewer are used frequently when compared with other legitimate software commonly used by adversaries.(Citation: Symantec Living off the Land) Remote access tools may be installed and used post-compromise as alternate communications channel for redundant access or as a way to establish an interactive remote desktop session with the target system. They may also be used as a component of malware to establish a reverse connection or back-connect to a service or adversary controlled system. Installation of many remote access tools may also include persistence (ex: the tool's installation routine creates a [Windows Service](https://attack.mitre.org/techniques/T1543/003)). Admin tools such as TeamViewer have been used by several groups targeting institutions in countries of interest to the Russian state and criminal campaigns.(Citation: CrowdStrike 2015 Global Threat Report)(Citation: CrySyS Blog TeamSpy)

Indicator

Name

3afbb6767c5b39888b4ddf7bc459cc1fea223e2e

Description

Unique file paths created by netd

Pattern Type

yara

Pattern

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 }

Name

338f8b447c95ba1c3d8d730016f0847585a7840c0a71d5054eb51cc612f13853

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '338f8b447c95ba1c3d8d730016f0847585a7840c0a71d5054eb51cc612f13853']

Name

ef466e714d5250e934e681bda6ebdecd314670bb141f12a1b02c9afddbd93428

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'ef466e714d5250e934e681bda6ebdecd314670bb141f12a1b02c9afddbd93428']

Name

5866e1fa5e262ade874c4b869d57870a88e6a8f9d5b9c61bd5d6a323e763e021

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '5866e1fa5e262ade874c4b869d57870a88e6a8f9d5b9c61bd5d6a323e763e021']

Name

b80b2f3b39aa7bcd6fb0190883054da0042b12d4

Description

netd pid for loop

Pattern Type

yara

Pattern

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 }

Name

2d19e015412ef8f8f7932b1ad18a5992d802b5ac62e59344f3aea2e00e0804ad

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '2d19e015412ef8f8f7932b1ad18a5992d802b5ac62e59344f3aea2e00e0804ad']

Name

8bd6ffd9f7da5317473dad7c1d000147b8664c9a

Description

Shell script commands found in netd

Pattern Type

yara

Pattern

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 }

Name

https://matt.ucc.asn.au/dropbear/dropbear.html

Pattern Type

stix

Pattern

[url:value = 'https://matt.ucc.asn.au/dropbear/dropbear.html']

Name

bbc987c4ff3ed7d85dee29b897c4b51190e9d43d

Description

killer binary strings

Pattern Type

yara

Pattern

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

Name

def7492fda3a3ac391aace5b25e238faffb739fd

Description

File extension strings

Pattern Type

yara

Pattern

rule netd_FileExtensionString { meta: author = "NCSC" description = "File extension strings" date = "2023-08-31" strings: $ = ".dat,.bak,.xml,.txt,.ovpn,.xml,wa.db,msgstore.db,.pdf,.xlsx,.csv,.zip,telephony.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 }

Name

33a2be6638be67ba9117e0ac7bad26b12adbcdf6f8556c4dc2ff3033a8cdf14f

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '33a2be6638be67ba9117e0ac7bad26b12adbcdf6f8556c4dc2ff3033a8cdf14f']

Name

44e0aa08ef5fdb2aec2f393078204fb70e271b63

Description

blob wait on event loop

Pattern Type

yara

Pattern

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

Name

048aaf4bf9286fb75e13dbc88fa62ab3e289c7ce

Description

netd wait loop

Pattern Type

yara

Pattern

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 }

Name

1de20cc0e637ed72e4ade338639808c8909d5a14

Description

db Android path strings

Pattern Type

yara

Pattern

rule db_androidpaths { meta: author = "NCSC" description = "db Android path strings" date = "2023-08-31" 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 }

Name

827278ad89486debd79c6bfc38d8b00fa5fb90ad

Description

db and td path strings found in netd

Pattern Type

yara

Pattern

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 }

Name

d1871e43c0a5ebc123eafa91b44ce00636ef02a3

Description

Tor configuration file strings in blob

Pattern Type

yara

Pattern

rule blob_TorCommandLine { meta: author = "NCSC" description = "Tor configuration file strings in blob" date = "2023-08-31" 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 }

Name

5c5323bd17fd857a0e77be4e637841dad5c4367a72ac0a64cc054f78f530ba37

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '5c5323bd17fd857a0e77be4e637841dad5c4367a72ac0a64cc054f78f530ba37']

Name

f38d170804471b31490d23fb483e0dd81c1a8be3

Description

blob path string found in netd

Pattern Type

yara

Pattern

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 }

Name

203890c3e60ccd4e05a2f9b1a784b010bb2d42d6

Description

Tor hostname path string found in netd

Pattern Type

yara

Pattern

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 }

Name

7ab58f4fcc3b53095ffa84c3de49c04d0072045f

Description

Application directories strings searched by netd

Pattern Type

yara

Pattern

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 }

Name

001208a304258c23a0b3794abd8a5a21210dfeaf106195f995a6f55d75ef89cd

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '001208a304258c23a0b3794abd8a5a21210dfeaf106195f995a6f55d75ef89cd']

Name

426a480c6c11190cc9d7c069ef409ae0b6bffa13

Description

POST request strings present in netd

Pattern Type

yara

Pattern

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 }

Name

294a16e3d8f507cdcfaa4582b0f93427fe924ad8

Description

ndbr scan strings

Pattern Type

yara

Pattern

rule ndbr_ScanStrings { meta: author = "NCSC" description = "ndbr scan strings" date = "2023-08-31" 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 }

Name

3cf2de421c64f57c173400b2c50bbd9e59c58b778eba2eb56482f0c54636dd29

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '3cf2de421c64f57c173400b2c50bbd9e59c58b778eba2eb56482f0c54636dd29']

Malware

Name
Infamouse Chisel

Country

Name
New Zealand
Name
Canada
Name
Australia
Name
United Kingdom of Great Britain and Northern Ireland
Name
United States of America

Intrusion-Set

Name
Sandworm

StixFile

Value
5c5323bd17fd857a0e77be4e637841dad5c4367a72ac0a64cc054f78f530ba37
5866e1fa5e262ade874c4b869d57870a88e6a8f9d5b9c61bd5d6a323e763e021
001208a304258c23a0b3794abd8a5a21210dfeaf106195f995a6f55d75ef89cd
33a2be6638be67ba9117e0ac7bad26b12adbcdf6f8556c4dc2ff3033a8cdf14f
338f8b447c95ba1c3d8d730016f0847585a7840c0a71d5054eb51cc612f13853
3cf2de421c64f57c173400b2c50bbd9e59c58b778eba2eb56482f0c54636dd29
ef466e714d5250e934e681bda6ebdecd314670bb141f12a1b02c9afddbd93428
2d19e015412ef8f8f7932b1ad18a5992d802b5ac62e59344f3aea2e00e0804ad

Url

Value
https://matt.ucc.asn.au/dropbear/dropbear.html

External References

- https://otx.alienvault.com/pulse/64f1e0b1856cc88a0eeff4d4
- https://www.cisa.gov/news-events/analysis-reports/ar23-243a