I'm not a bot

How to use kismet

Kismet is a Network Sniffer Tool and IDS for Wireless Networks ========== Overview of Kismet ---------- Kismet is a powerful tool used to sniff through 802.11 layer 2 traffic, detect wireless networks, and provide real-time intrusion detection capabilities. Developed by Mike Kershaw and written in C++, this tool is widely used among security professionals and enthusiasts alike. Installation ---------- To install Kismet, follow these steps: 1. Clone the kismet repository using `git clone 2. Create a new directory and navigate to it. 3. Insert your wireless card and check ifconfig to ensure Wlan is working. 4. Enable airmon-ng by running `sudo airmon-ng start Wlan0`. 5. Run `iwconfig` or `ifconfig` to verify that airmon-ng is started. Starting Kismet ---------- To begin capturing packets, run the following command: `kismet -c YourCardNameMon` Replace `YourCardNameMon` with your wireless card name and `-c wlan0mon` to specify the capture source. Logging In ---------- Enter your username and password to access the interface. Using Kismet ---------- Kismet provides a simple and intuitive interface for monitoring wireless networks. It detects and deciphers 802.11, Bluetooth, Zigbee, RF, and other protocols, making it an essential tool for wireless security assessment. Recommendation ---------- As a highly recommended tool for monitoring networks, Kismet is a must-have for any security professional or enthusiast. For more information on this powerful tool, visit nullbyte's site. Changes and Updates ---------- * **Kismet 2023-07-R2**: A minor update addressing an html/js injection path in the web UI. * **Kismet 2023-07-R1**: A major release with significant speed boosts, memory improvements, bug fixes, dark-mode UI, improved RF sensor and power meter support, and more. Given text about installing and using Kismet on Kali Linux for wireless network tracking. Kismet is a Wireless Network Detector and Intrusion Detection System for Kali Linux ========== ### What is Kismet? Kismet is a tool used to detect hidden wireless networks and automatically connect to them. It can also identify intrusion in wireless networks and find the location of wireless access points. ### Installing Kismet on Kali Linux 1. Update the system by running `sudo apt-get update` 2. Install required dependencies using `sudo apt install build-essential git libwebsockets-dev pkg-config zlib1g-dev libnl-3-dev libnl-genl-3-dev libcap-dev libpcap-dev libnm-dev libdw-dev libsqlite3-dev libprotobuf-dev libprotobuf-c-dev protobuf-compiler protobuf-c-compiler libsensors4-dev libusb-1.0-0-dev` 3. Install Python packages using `sudo apt-get install python python3-setuptools python3-protobuf python3-requests python3-numpy python3-serial python3-usb python3-dev python3-websockets librtlsdr0 libubertooth-dev` 4. Install libusb using `sudo apt-get install libusb-1.0-0-dev` ### Clone the Kismet Repository Clone the Kismet repository using `git clone --recursive /kismet` and run `./configure`, `make -j$(nproc)`, and `sudo make suidinstall`. ### Launching Kismet on Kali Linux 1. Add yourself to the Kismet group by running `sudo usermod -a -G kismet YourUsername` 2. Connect your wireless network card using `ip aifconfig` and put it in monitor mode with `sudo airmon-ng start YourCardName`. 3. Launch Kismet by running `kismet -c YourCardNameMon`. ### Using Kismet on Kali Linux 1. A list of Wi-Fi devices will appear, sorted by name, signal strength, etc. 2. Select a network to view its clients and access information. Note: Ensure that the wireless card is named as "wlan1" or "wlan0" for proper functionality. To obtain client information, navigate to the "Client List" option under "Windows" in the menu. This is crucial for understanding client behavior and activity. Note that your Wi-Fi card can be utilized to track your location even when not connected to Wi-Fi. When you're not utilizing client devices, such as smartphones, we recommend disabling their Wi-Fi settings. This precaution helps prevent unwanted tracking or monitoring. This article has provided an overview of the Kismet tool in Kali Linux, including its installation and usage. Additionally, it covered methods for dealing with device identification by Kismet. If you have any further questions, feel free to comment below. Using Kismet for Advanced Wireless Analysis ========== ### Use Case 1: Enhanced Data Collection Utilize detected access points and activities to gather richer data for analysis. ### Use Case 2: Customized Capture and Storage ```bash sudo kismet -c wlan0 -d path/to/output ``` This command saves captured packet data to a specified directory, allowing for post-capture analysis and compliance with legal requirements. ### Use Case 3: Predefined Configuration ```bash sudo kismet -c wlan0 -f path/to/config.conf ``` Using a specific configuration file enables the application of predefined settings, such as custom alert triggers or interface options, providing an optimized environment for auditing sessions or devices. ### Use Case 4: Structured Data Logging ```bash sudo kismet -c wlan0 --log-to-db ``` Logging data to an SQLite database allows for structured and persistent storage of captured data, essential for scalable analytics and long-term data retention. ### Use Case 5: Advanced Data Monitoring ```bash sudo kismet -c wlan0 --data-source=rtl433 ``` Using a specific data source, such as RTL-SDR (rtl433), is crucial for specialized monitoring tasks that align with the capabilities of the chosen data source. Wireless ranges allow for broad-spectrum data analysis, including signals from various devices such as weather stations and tire pressure monitors. Use Case 7: Enable Alerts for Specific Events enables real-time monitoring by detecting new access points and alerting administrators to potential intrusions or unwanted behavior. Kismet command `sudo kismet -c wlan0 --enable-alert=new_ap` allows rapid identification of unauthorized devices attempting to connect to or mimic legitimate network hardware. Example output includes logged messages displaying each detected new access point, with notifications available as alerts to administrators. The `--info BSSID` Kismet command displays detailed packet information for a specific access point (by BSSID), enabling targeted analysis of network traffic and performance evaluation. Upon running the command, Kismet provides an array of detailed packet stats, including packet counts, encryption protocols, data rates, and timing analysis, giving insights into activity centered on that network element. Kismet's varied functionalities, from focusing on particular interfaces to extensive database logging and bespoke monitoring setups, enhance the user's capability to secure and scrutinize wireless networks proficiently and effectively. On networks, capturing traffic refers to the process of collecting data packets as they travel through the system. This can be done using various methods, including network taps, switches with port mirroring, and wireless LANs. To capture unicast traffic, adapters must be in promiscuous mode, while multicast and broadcast traffic can be captured without any special provisions. Captured information is typically decoded into a human-readable format for analysis by protocol analyzers. These tools can display and analyze data, generate traffic, and even introduce errors to test devices. They are used to detect network problems, intrusion attempts, and misuse, as well as monitor bandwidth utilization, security status, and gather statistics. Additionally, they can aid in troubleshooting performance issues, monitoring application data, and serve as the primary source for day-to-day network management and monitoring. Packet capture is a method used to fulfill law enforcement warrants and wiretap network traffic. Internet service providers and VoIP providers in the US must comply with regulations from the Communications Assistance for Law Enforcement Act. To provide secure access, telecommunications carriers use packet capture and storage, which allows them to share targeted traffic with internal security systems while maintaining confidentiality through end-to-end encryption. The use of packet sniffing tools can be seen in various software applications such as CommView, dSniff, Ethereal, and Wireshark. Packet sniffing, also known as network or protocol analysis, involves capturing and analyzing packets of data transmitted over a computer network. According to "International Journal of Electrical, Electronics and Computer Engineering," packet sniffing can be achieved through various tools, including protocol analyzers, which are hardware devices or software programs that enable the capture, storage, and analysis of network traffic. A packet analyzer is often referred to as a network analyzer or packet sniffer, and it allows users to monitor and analyze data packets transmitted over the network. This can include capturing and analyzing individual packets, as well as monitoring network activity in real-time. Protocol analyzers are used in various applications, including network forensics, security analysis, and troubleshooting. They work by capturing packets of data from the network and then analyzing them to identify patterns, anomalies, or potential security threats. Kismet is a popular protocol analyzer tool that allows users to detect and analyze wireless networks, devices, and protocols. It can be used for wardriving, wireless intrusion detection, and other applications. kismet_capture_common and kismet_capture_hak5_wifi_coconut contain Kismet utilities for network device detection, snifering, wardriving and WIDS, that work with Wi-Fi interfaces, Bluetooth interfaces, some SDR hardware like RTLSDR, and specialized capture hardware. ####Dependencies debconf | debconf-2.0 kismet_capture_common libc6 libcap2 libprotobuf-c1 libusb-1.0-0 libwebsockets19t64 ####Usage kismet_cap_hak5_wifi_coconut -h This driver is typically run automatically by the Kismet server. The kismet_cap_hak5_wifi_coconut supports sending data to a remote Kismet server with options: --connect [host]:[port] Connect to remote Kismet server on [host] and [port]. --tcp Use the legacy TCP remote capture protocol, when combined with the --connect option. --ssl Use SSL to connect to a websocket-enabled Kismet server ####Options --user [username] Kismet username for a websockets-based remote capture source. --password [password] Kismet password for a websockets-based remote capture source. --apikey [api key] A Kismet API key for the 'datasource' role; this may be supplied instead of a username and password for websockets based remote capture. --endpoint [endpoint] An alternate endpoint for the websockets connection. ####Example kismet_cap_hak5_wifi_coconut [options] --connect [host]:[port] Connect to remote Kismet server on [host] and [port]; by default this now uses the new websockets interface built into the Kismet webserver on port 2501; kismet-capture-linux-bluetooth Kismet is a wireless network and device detector, sniffer, wardriving tool, and WIDS (wireless intrusion detection) framework. It works with Wi-Fi interfaces, Bluetooth interfaces, some SDR hardware like RTLSDR, and other specialized capture hardware. This package contains the Kismet Linux Bluetooth capture helper. To install: sudo apt install kismet-capture-linux-bluetooth Dependencies: debconf | debconf-2.0 kismet-capture-common libc6 libcap2 libprotobuf-c1 libwebsockets19t64 Usage: kismet_cap_linux_bluetooth [options] --connect [host]:[port] Connect to remote Kismet server on [host] and [port]; by default this now uses the new websockets interface built into the Kismet webserver on port 2501; to connect using the legacy remote capture protocol, specify the '--tcp' option and the appropriate port, by default port 3501. Options: --tcp Use the legacy TCP remote capture protocol. --ssl Use SSL to connect to a websocket-enabled Kismet server --ssl-certificate [certfile] Use SSL to connect to a websocket-enabled Kismet server and use the provided certificate authority certificate to validate the server. --user [username] Kismet username for a websockets-based remote capture source. --password [password] Kismet password for a websockets-based remote capture source. --apikey [api key] A Kismet API key for the 'datasource' role; this may be supplied instead of a username and password for websockets based remote capture. --endpoint [endpoint] An alternate endpoint for the websockets connection. --disable-retry Do not attempt to reconnect to a remote server if there is an error; exit immediately. Fixed-GPS and Daemonization Options The command `kismet_cap_linux_wifi` can be used to set a fixed location, daemonize the tool, list supported devices, or look for a Kismet server in announcement mode. **Fixed Location** Set a fixed GPS location using `[lat,lon,alt]` or `[lat,lon]`. Example: `kismet_cap_linux_wifi [lat,lon,alt]` **Daemonization** Run the capture tool in daemon mode with the `--daemonize` option. Example: `kismet_cap_linux_wifi --daemonize` **List Supported Devices** Detect supported devices using the `--list` option. Example: `kismet_cap_linux_wifi --list` **Remote Capture** Look for a Kismet server in announcement mode and optionally wait for a specific server UUID to be seen using `[uuid:optional]`. Example: `kismet_cap_linux_wifi [uuid:optional]` **Kismet Server Configuration** Requires a Kismet server configured for announcement mode. Fixed GPS and Remote Capture Configuration Options for Kismet Capture NRF51822 By default, remote captures will continuously try to reconnect if the server becomes unavailable. Kismet is a wireless network and device detector, sniffer, wardriving tool, and WIDS framework that works with Wi-Fi interfaces, Bluetooth interfaces, SDR hardware, and other specialized capture hardware. This package contains the NRF52840 BTLE Sniffer capture helper. To use it, you can install Kismet using `sudo apt install kismet-capture-nrf-52840`. The tool has several options for customizing its behavior, including setting a fixed location, choosing an alternate GPS name, and enabling daemon mode or listing supported devices. To connect to a remote Kismet server, you can use the `--connect` option followed by the host and port. You can also specify the legacy TCP protocol using `--tcp`, enable SSL connections with `--ssl`, and provide a certificate authority certificate for validation. Additionally, you can set a username and password or API key for websockets mode. Some other options include setting an endpoint for the websocket connection, specifying a source to send to the remote server, disabling retry attempts in case of errors, and enabling fixed GPS locations. Server errors will trigger an immediate exit. By default, a remote capture will attempt to reconnect indefinitely if the server is unavailable. The `--fixed-gps` option sets a fixed location for this capture (remote only), accepting lat, lon, and alt or just lat and lon. The `--gps-name` option sets an alternate GPS name for this source. Run the capture tool in daemon mode with `--daemonize`. List supported devices detected with `--list`. Use `--autodetect` to look for a Kismet server in announcement mode, optionally waiting for a specific server UUID. Requires a Kismet server configured for announcement mode. Kismet is a wireless network and device detector, sniffer, wardriving tool, and WIDS framework that works with Wi-Fi interfaces, Bluetooth interfaces, some SDR hardware like the RTLSDR, and other specialized capture hardware. This package contains the Kismet nRF MouseJack capture helper, which has an installed size of 176 KB. To install, run `sudo apt install kismet-capture-nrf-mousejack`. The dependencies are debconf, debconf-2.0, kismet-capture-common, libc6, libcap2, libprotobuf-c1, libusb-1.0-0, and libwebsockets19. The usage for `kismet_cap_nrf_mousejack` is as follows: `kismet_cap_nrf_mousejack [options]`. Connect to a remote Kismet server using the `--connect` option followed by `[host]:[port]`. By default, this uses the new websockets interface built into the Kismet webserver on port 2501. To connect using the legacy remote capture protocol, specify the `--tcp` option and the appropriate port (default is port 3501). Other options include `--ssl` for SSL connections to a websocket-enabled Kismet server, `--ssl-certificate [certfile]` for specifying a certificate authority certificate, `--user [username]` and `--password [password]` for username and password authentication in websockets mode, and `--apikey [api key]` for using a Kismet API key. Capture remote servers with no retries and exit immediately. Set a fixed location for this capture (remote only) using latitude, longitude, and altitude coordinates or latitude and longitude. Provide an alternate GPS name for this source. Run the capture tool in daemon mode for background execution. List supported devices detected by Kismet. Automatically detect a Kismet server in announcement mode with optional waiting for a specific UUID to be seen. This package contains the Kismet NXP KW41Z BTLE and Zigbee Sniffer capture helper, which works with Wi-Fi interfaces, Bluetooth interfaces, software-defined radio hardware like RTLSDR, and other specialized capture devices. The kismet-capture-nxp-kw41z command is used for Kismet's wireless network and device detection. It supports sending data to a remote Kismet server using various options, including connection to the host and port, SSL encryption, username, password, API key, and endpoint. Given text content here server; only used in conjunction with remote capture. --disable-retry Do not attempt to reconnect to a remote server if there is an error; exit immediately. By default, a remote capture will keep trying indefinitely if the server is unavailable. --fixed-gps [lat,lon,alt] Set a fixed location for this capture (remote only), accepts lat,lon,alt or lat,lon --gps-name [name] Set an alternate GPS name for this source --daemonize Background the capture tool and enter daemon mode. --list List supported devices detected --autodetect [uuid:optional] Look for a Kismet server in announcement mode, optionally waiting for a specific server UUID to be seen. Requires a Kismet server configured for announcement mode. Specify a source to send to the remote Kismet server; use with remote capture. Disable retry if there's an error; exit immediately instead of reconnecting indefinitely. For remote captures, specify fixed location using lat,lon,alt or lat,lon. Set alternate GPS name for this source using --gps-name. Run capture tool in background mode using --daemonize. List supported devices detected with --list. Look for Kismet server in announcement mode with --autodetect and optionally wait for specific UUID to be seen. This requires a Kismet server configured for announcement mode. Kismet is a wireless network detector, sniffer, wardriver, and WIDS framework that works with Wi-Fi interfaces, Bluetooth interfaces, some SDR hardware like RTLSDR, and other specialized capture hardware. This package contains the Kismet TICC2531 802.15.4 Zigbee Sniffer capture helper, which is a capture driver for Kismet. To start the capture driver, use the command `kismet_cap_ti_cc_2531 -h`. The driver supports sending data to a remote Kismet server using options like --connect, --tcp, and --ssl. It also requires a username, password, or API key for websockets mode. --source [src def] Specify a source for sending data to a remote Kismet server, used with remote capture only --disable-retry Do not reconnect to a remote server on error; exit instead of retrying. Remote captures usually attempt infinite retries by default --fixed-gps [lat,lon,alt] Set a fixed location for this capture (remote only), accepts lat,lon,alt or lat,lon gps coordinates --gps-name [name] Use an alternate GPS name for this source --daemonize Run the capture tool in background mode. --list Show supported devices detected by the tool --autodetect [uuid:optional] Look for a Kismet server announcement, optionally waiting for a specific UUID; requires Kismet configured for announce mode kismet-capture-ti-cc-2540 This is the Kismet TICC2540 BTLE Sniffer capture helper package Installed size: 176 KB To install: sudo apt install kismet-capture-ti-cc-2540 Dependencies: debconf | debconf-2.0, kismet-capture-common, libc6, libcap2, libprotobuf-c1, libusb-1.0-0, libwebsockets19, t64 kismet_cap_ti_cc_2540 This is a capture driver for Kismet, usually started by the server usage: kismet_cap_ti_cc_2540 [options] --connect [host]:[port] Connect to remote Kismet server on [host] and port; default websockets interface uses port 2501 --tcp Use legacy TCP remote capture protocol with --connect option --ssl Secure connection using SSL to websocket-enabled Kismet server --ssl-certificate Use provided certificate authority certificate for SSL validation --user Kismet username for websockets-based remote capture source --password Kismet password for websockets-based remote capture source --apikey Kismet API key for 'datasource' role, used instead of username and password in websockets mode --endpoint Alternate endpoint for websocket connection This guide explains how to use the Kismet capture tool, specifically its Ubertooth One BT Sniffer capture helper. * To specify a remote server for capturing data, you can use the `--source` option followed by the name of the source to send to. * If there is an error connecting to the remote server, you can disable automatic reconnection by using the `--disable-retry` option. * The `--fixed-gps` option allows you to set a fixed location for this capture (remote only), accepting latitude, longitude, and altitude values. * To set an alternate GPS name for this source, use the `--gps-name` option followed by the desired name. * Backgrounding the capture tool and entering daemon mode can be achieved using the `--daemonize` option. * You can list supported devices detected by using the `--list` option. * To look for a Kismet server in announcement mode, use the `--autodetect` option followed by an optional UUID value. The Ubertooth One BT Sniffer capture helper is a part of the Kismet package and helps with wireless network and device detection. It works with Wi-Fi interfaces, Bluetooth interfaces, some SDR hardware like the RTLSDR, and other specialized capture hardware. This tool is included in the Kismet package and helps with wireless network and device detection. The installed size of this package is 172 KB, and it requires dependencies such as `debconf`, `debconf-2.0`, `kismet-capture-common`, `libc6`, `libcap2`, `libprotobuf-c1`, `libubertooth1`, `libwebsockets19`, and `t64`. To install this package, you can use the command `sudo apt install kismet-capture-ubertooth-one`. You can use the options `--connect`, `--tcp`, `--ssl`, `--ssl-certificate`, `--user`, `--password`, `--apikey`, and `--endpoint` to customize your capture process. --endpoint=/kismet/proxy/datasource/remote/remotesource.ws --source [source def] Specify a source to send to the remote Kismet server; only used in conjunction with remote capture. --disable-retry Do not attempt to reconnect to a remote server if there is an error; exit immediately. By default, a remote capture will try to reconnect indefinitely if the server is unavailable. --fixed-gps [lat,lon,alt] Set a fixed location for this capture (remote only), accepts latitude, longitude, and altitude or just latitude and longitude. --gps-name [name] Set an alternate GPS name for this source. --daemonize Background the capture tool and enter daemon mode. --list List supported devices detected. --autodetect [uuid:optional] Look for a Kismet server in announcement mode, optionally waiting for a specific server UUID to be seen, requires a configured Kismet server. server on [host] and [port]; by default, it now utilizes the new websockets interface integrated within the Kismet webserver on port 2501. To establish a connection using the legacy remote capture protocol, specify the '--tcp' option along with the corresponding port number, which defaults to port 3501. --tcp Use the legacy TCP remote capture protocol in conjunction with the '--connect' option. The modern protocol relies on websockets built into the Kismet server and doesn't require this option. --ssl Establish an SSL connection to a websocket-enabled Kismet server. --ssl-certificate Specify the SSL certificate for connecting to a websocket-enabled Kismet server, thereby validating the server's authority. --user Enter your Kismet username for a websockets-based remote capture source; a combination of username and password or API key is necessary. Note that usernames are only used in websockets mode, while passwords are also unique to this setting. --password Enter your Kismet password for a websockets-based remote capture source; similar to the username, a combination of credentials is required for websocket mode. A username and password will be exclusively used with websockets connections. --apikey Provide a Kismet API key for the 'datasource' role, which may replace the need for a username and password in websocket-based remote captures. The provided API key is unique to this setting as well. --endpoint Set an alternate endpoint for your websocket connection; by default, remote datasources are terminated at /datasource/remote/remotesource.ws, with the exception of when using an HTTP proxy. Endpoints should be formatted as a full path to the websocket endpoint. For instance: --endpoint=/kismet/proxy/datasource/remote/remotesource.ws --source Specify a source for remote transmission to the Kismet server; this parameter is only applicable in conjunction with remote capture. --disable-retry Prevent reconnect attempts to a remote server following an error; immediately exit instead of retrying indefinitely by default, which will reconnect without issue when available. By providing '--fixed-gps', you can set a fixed location for your capture (remote capture only), accepting either geographical coordinates or a combination of the same. Alternatively, you can use '--gps-name' to designate an alternate GPS name for your source. --daemonize Operate the capture tool in background mode and enter daemon mode. --list List supported devices detected during connection attempts. The option '--autodetect' is optional, requiring Kismet server configuration and awaiting a specific server UUID if desired. To connect to a websocket-enabled Kismet server, use SSL and provide the certificate authority certificate for validation purposes. The username "username" is required for websockets mode, along with either a password or API key. Only usernames and passwords are used in websockets mode. --api_key [API Key] For remote capture, an alternate endpoint can be specified if needed. By default, data sources terminate at "/datasource/remote/remotesource.ws" but may need to be adjusted when using a HTTP proxy that houses the Kismet service under a directory. Endpoints should include the full path to the websocket connection. --source [Source Def] In remote capture mode, specify the source to send data to the Kismet server. The "disable-retry" option prevents reconnecting to a remote server if there's an error. By default, it attempts indefinite reconnections until the server becomes available. --fixed-gps [Lat,Lon,Alt] Remote captures can have a fixed location set using lat, lon, and alt coordinates or just lat and lon. This setting is only applicable for remote sources. --gps-name [Name] Provide an alternate GPS name for this source. If the tool gets stuck in infinite reconnections, check if the server is down. Kismet is a wireless network and device detector, sniffer, wardriving tool, and WIDS (wireless intrusion detection) framework. It supports various interfaces including Wi-Fi, Bluetooth, SDR hardware like RTLSDR, and specialized capture hardware. The kismet-logtools package contains tools for managing Kismet logs, such as kismetdb_clean, kismetdb_dump_devices, kismetdb_statistics, and kismetdb_strip_packets. These tools can be used to perform various tasks, including cleaning up incomplete log files, converting device data to JSON or GPX formats, stripping packet content, and generating statistics. Some of the options available for these tools include: - Overriding logged types - Setting default log title - Specifying a directory for storing log files - Disabling logging entirely Additionally, kismetdb_to_gpx and kismetdb_to_kml tools can be used to convert packet data from KismetDB logs to GPX or KML formats, respectively. These tools offer options such as excluding records within a certain distance of a specified location and using basic average location information for faster but less accurate results. kismetdb_to_kml -i, --in [filename] Convert kismetdb file to KML for Google Earth -o, --out [filename] Output KML file -f, --force Force writing to target file, even if it exists -v, --verbose Verbose output -s, --skip-clean Don't clean input database -e, --exclude lat,lon,dist Exclude records within 'dist' meters of provided lat,lon -g, --group Group by type into KML folders kismetdb_to_pcap -i, --in [filename] Convert kismetdb file to pcap -o, --out [filename] Output file name -f, --force Overwrite any existing output files -v, --verbose Verbose output -s, --skip-clean Don't clean input database kismetdb_to_wiglecsv -i, --in [filename] Convert kismetdb file to WigleCSV -o, --out [filename] Output Wigle CSV file -f, --force Force writing to target file, even if it exists -v, --verbose Verbose output Note: The options for each command are separated by a newline character for better readability. Given article text here output -s, --skip-clean Don't clean (sql vacuum) input database -e, --exclude lat,lon,dist Exclude records within 'dist' *meters* of the lat,lon provided. This can be use to exclude packet close to your home, or other sensative location. Kismet is wireless network and device detector, sniffer, wardriving tool, and WIDS (wireless intrusion detection) framework. It work with Wi-Fi interfaces, Bluetooth interfaces, some SDR (software defined radio) hardware like the RTLSDR, and other specialized capture hardware. This package contain kismet BLTE geiger datasource. installed size: 377 KB dependencies python3, kismet eventbus, externalkismet_discovery root@kali:~# kismet_eventbus -h usage: kismet_eventbus [-h] [--in-fd INFD] [--out-fd OUTFD] root@kali:~# kismet_proxytest -h usage: kismet_proxytest [-h] [--in-fd INFD] [--out-fd OUTFD] installed size: 109 KB dependencies python3, python3-bluepypython3-protobufpython3-websocketskismet_cap_bt_geiger Kismet is a comprehensive wireless network and device detector, sniffer, wardriving tool, and WIDS framework. It supports various interfaces, including Wi-Fi, Bluetooth, SDR hardware like RTLSDR, and specialized capture hardware. #### Installation To install Kismet, run the command `sudo apt install python3-kismetcapturefreaklabszigbee`. #### Dependencies Kismet requires the following dependencies: python3, python3-protobuf, python3-serial, python3-websockets, and kismet_cap_freaklabs_zigbee. ### Usage The basic syntax for Kismet is: `kismet_cap_freaklabs_zigbee -h` This displays the help message and exits. #### Options Some common options include: * `--in-fd INFD`: specifies the incoming fd pair (IPC mode only) * `--out-fd OUTFD`: specifies the outgoing fd pair (IPC mode only) * `--connect CONNECT`: specifies the remote Kismet server on host:port * `--source SOURCE`: captures the source definition, required for remote capture #### Example Use Case To use Kismet with a specific server UUID and in announce mode, run: `kismet_cap_freaklabs_zigbee --autodetect ` Note that Kismet requires Freaklabs hardware or compatible SenSniff-based devices. Looking to set up Kismet for remote capture? Start by enabling TCP mode, legacy protocol --tcp enable, and SSL encryption --ssl enable. Provide a valid SSL CA certificate --ssl-certificate SSLCERTIFICATE to validate the server. Additionally, specify your Kismet username and password for websockets-based remote capture --user USER and --password PASSWORD respectively. An API key is also required --apikey APIKEY for secure access. You can use an alternate endpoint --endpoint ENDPOINT if needed. To disable automatic reconnection, set --disable-retry ENDPOINT. The package includes the rtladsb datasource for RTL-SDR support. Kismet Datasource Options The following options are available for configuring the RTL-SDR AMR Kismet datasource: * Command-line arguments: + --help: Displays help message and exits. + --in-fd, --out-fd: Specifies incoming and outgoing file descriptors (IPC mode only). + --connect: Establishes a connection to a remote Kismet server on host:port. Defaults to websocket mode; use --tcp for legacy TCP mode. * Capture source options: + --source: Defines the capture source definition, required for remote capture. * Security options: + --ssl: Enables SSL encryption. + --ssl-certificate: Provides a SSL CA certificate for server validation (SSLCERTIFICATE). + --user, --password: Specifies Kismet username and password for websockets-based remote capture. + --apikey: Supplies the Kismet API key for websockets-based remote capture. * Endpoint options: + --endpoint: Sets an alternate endpoint for websockets remote capture. + --disable-retry: Disables automatic reconnection on endpoint failure. * Autodetect option: + --autodetect: Enables server autodetection, optionally waiting for a specific server UUID.