**Development platform**

* Server: golang
* Client: jquery / websockets

**Assumptions / Limitations / Discussion points / Research Needed**

* Kismet will be pre-installed and running
* Third-party tools will be pre-installed (checked during startup)
* Some of the tools will be re-written/updated to include in native code (Please update the sections below accordingly. I added what I could)
* Original plan was to allow someone to run the web interface locally and connect to a remote kismet server; however, with the added additions, my guess is that we will need to run this on the same device as the kismet server.
* Need to re-evaluate if sqlite will be necessary

**Overall application structure**

* Single binary – No dependencies other than kismet and associated tools.
* Static files will be compiled into a package prior to building the final version of the tool (see statik package)
* Modular – Built in a way that a new tool can be introduced to the framework by writing a golang package and installing the associated tool
* Compatible with any version of kismet

**Application Arguments**

* lhost – The HTTP server host IP address to bind to (Default: 127.0.0.1)
* lport – The HTTP server port (Default: 8080)
* khost – The kismet server IP address (Default: 127.0.0.1)
* kport – The kismet server port (Default: 2501)
* dbfile – The sqlite database file name (Default: gokismet.db)  
  \*\*\* I would like to do some testing on how long it would take to import one or more large kismet packet/xml files. If the time is negligible, we could probably remove this and work strictly off of the kismet files. \*\*\*
* outdir – The kismet output file directory (Default: .)
* debug – Debug flag, which will display additional data to the console if enabled (Default: false)
* aircrack – Full path to aircrack (Default: search default PATH)
* asleap – Full path to asleap (Default: search default PATH)
* john – Full path to john the ripper (Default: search default PATH)
* hashcat – Full path to oclHashcat (Default: search default PATH)
* kismograph – Full path to kismograph executable (Default: search default PATH)
* evilap – Location of EvilAP scripts (Default: /opt/evilap)
* <Chris’ visualization tool> - Location of script(s) (Default: /opt/<script name>)

**Attack Sequences / Tools**

* Deauth one or all clients
* Configure evil twin
* Send captured pre-shared key handshake to Aircrack, john, or hashcat
* Send captured radius hashes to asleap, john, or hashcat

**Functional Requirements**

Server

* Create sqlite database file or read in the file if it exists
* Verify existence of third-party tools
* Kismet
  + Establish connection to kismet
  + Send commands to kismet
  + Process kismet responses
  + Dump all data to database file upon receiving a “Save Files” response from kismet
* HTTP Server
  + Startup HTTP service
  + Listen for websocket calls
  + Update web interface for specified kismet responses
* Misc
  + Parse output from third-party tools
  + Send notifications when keys/passwords are cracked and update the interface accordingly

Navigation bar

* Connect/Disconnect from kismet
* Programmatically generate links to included functions
* Links should be based on the packages associated with each embedded tool and only appear if the tool exists

Home page

* Display kismet server information  
  \*\*\* I’ve tested against a couple versions of Kismet and the information doesn’t appear to change. We could potentially pull this functionality \*\*\*
* Display real-time server statistics
* List in-scope SSIDs
* List wireless interfaces
* View in-scope SSID information
  + Network Name (SSID)
  + Cloaked (Y/N)
  + Channels
  + First seen timestamp
  + Last seen timestamp
  + Network speed
  + Max signal strength
  + Min signal strength
  + Client count
  + BSSID count
  + Encryption type
  + Captured handshake (Y/N)
* View in-scope NIC information
  + Interface name
  + Active/Inactive
  + Alias
  + Current channel
  + Hopping (Y/N)
  + Velocity (Channels/sec)
  + Channel list
* Manage wireless interfaces
  + Add as kismet source
  + Delete from kismet service  
    \*\*\* The API call for this works, however, it seems to crash the kismet service. From what I can tell, it’s an issue with kismet, not the application. I’ve tested this by manually connecting to kismet through netcat and issuing the command and still have a crash. I’m guessing this is why the option doesn’t appear in the standard kismet client. \*\*\*

Discovery page

* Manage wireless interface
  + Switch hopping/locked
  + Select locked channel / List of hopping channels
  + Set the velocity when hopping
* Pause output to allow for copy/paste
* Filter – Programmatically generate filter options from kismet data. AND each option.
  + Network scope
    - in-scope
    - Rogues
  + Hidden (Y/N)
  + Network name (SSID)
  + Channel
  + BSSID  
    \*\*\* Will search for BSSIDs that match all or a portion of the BSSID. For example aa:bb:cc: would find all BSSIDs that begin with aa:bb:cc:
  + Clients
    - Associated
    - Not-associated
* Display real-time network information
  + SSID
  + BSSID
  + Channel
  + Last seen
  + Current signal strength
  + Max signal strength
  + Number of clients
  + Number of data packets
  + Captured handshake (Y/N)
* Sort on each network column, ascending or descending
* Display full details regarding each network/bssid
  + Include all characteristics of the network/bssid, as per information from kismet
  + Show all associated notes
  + Show if the network is in-scope or a rogue
  + Add/remove network to/from the scope
  + Send network/bssid to available tools
  + Real-time heat maps
* Display real-time client information
  + Associated Network Name (SSID)
  + MAC address
  + Last seen
  + Current signal strength
  + Max signal strength
  + Min signal strength
  + Packet count
* Sort on each client column, ascending or descending
* Display full client details
  + Send to available tools
  + Show all associated notes
* Notes
  + Note type:
    - General
    - Network (SSID)
    - Access Point (BSSID)
    - Client

Notes

* Display all notes
* Add/Modify/Delete notes
* Order by type and/or time

Kismograph

* Select/Filter networks/access points for inclusion in graph
* Select/Filter clients for inclusion in graph
* Generate temp files for kismograph
* Execute kismograph
* Update web page with graphic

Chris’ Visualization Tool (Couldn’t remember what it was called or find the scripts)

* Add content

EvilAP

* Configure EvilAP
* Execute the EvilAP scripts to generate the fake access point
* Display the output in real-time within the interface