Mission Groups

- Parent quest ~ Required completion
- Order ~ Order of apparition (on user & admin page) in relation to other groups.
- There are normal groups and groups with specific purposes (ORG wars, Zone wars..)

Mission

Basic data

- **Order in group** ~ Missions are ordered for user ad admins alike by this.
- **Required parent quest** ~ Required completion (you can use a and for daily missions).
- **Type** ~ Normal - only once | Daily - rewards given every 24h | Repeatable - story value, no rewards
- **Energy** ~ required energy to start quest

*Special Syntax **[Q-Syntax]***

Syntax	**Definition**
USERNAME	Replaced by hacker name
IP_X	X = integer: {0 to numberOfHosts - 1}. Random IP's are generated at the beginning of a missions and assigned to hosts. **e.g. IP_0 = first IP from list = first host (the main server)**
IHP_X_Y	X is same as above. Y refers to which part of the IP you want hidden. **e.g. IP_1 = 11.22.33.44; IHP_1_2 = 11.xx.33.44; IHP_1_4 = 11.22.33.xx**
IRP_X_Y	X same as above. Y refers to which part to reveal. **e.g. IP_1 = 11.22.33.44; IRP_1_2 = xx.22.xx.xx; IRP_1_4 = xx.xx.xx.44**
base64(string)	Encodes given string to base64. **e.g. base64(test) will become dGVzdA==**
RAND_S_X_Y	You can use random alphanumeric strings. X is the identifier of the string. Starting with 0 and ending with X-1 (same principle with IP_X). All strings have 50 characters but you can choose to use only the first Y. You can use as passwords or inside files, running parameters, etc. **e.g. RAND_S_0_10 => will show the first 10 characters of a random string**
RAND_S_X_Y_Z	Same as RAND_S_X_Y just that instead of showing the first Y characters it will show characters from position Y to position Z. Note: The first character has position 0). **e.g.: RAND_S_0_0_9 will have the same output as RAND_S_0_10**
RAND_N_X_Y	Will generate a random number between X and Y. You cannot reuse it like you do for strings where you have an identifier. But you can use it for ports, encryptions, filenames, or let the user guess the number as a parameter ^^.

Objectives

When an objective is completed, the next one is presented until none are left.

- **Order** ~The sequence in which they must to be completed. **IF TWO OBJECTIVES HAVE THE SAME ORDER**: one will be picked randomly for hackers to complete. Does not need to be strictly incremental (one objective can have order 10 and the next 20).

- **Extra time** ~ Gets added to remaining time upon objective completion.
- **Achievement** ~ Delivered if mission is successful.
- **Objective story ~ BBCode Enabled** ~ Story displayed to user.

Objectives are completed when all compulsory side-objectives are finished.

Side-Objectives

They share similar properties with and are childs of objectives.

- **Type** ~ Defines the type of action needed to be completed.
- **Objective data** ~ Defines the type of action needed to be completed.
- **Group** ~ Kind of same as order for Objectives just that order is irrelevant for side-objectives so we call it grouping. Basically if two or more side-objectives are in the same group and one of them gets completed, all side-objectives of same group and same objective are marked as completed.
- **Extra time** ~ Added to remaining time upon objective completion.
- **Compulsory** ~ Whether the side-objective must be done or is optional.
- **Achievement** ~ Delivered if mission is successful

Type	**Data Syntax**	**e.g.**	**Completed when**
ping	IP_X	IP_1	IP_X is pinged
crack	IP_X:port	IP_3:200	IP_X:port is cracked
connect	IP_X:port	IP_2:100	User connected to given IP && port. This is triggered by ssh, smtp, sql commands..
delete_file	filename	test.txt	File is deleted (no matter where it's located)
transfer	filename|IP_X:port	test.txt|IP_1:22	File is transferred to given IP and port. Obviously you will have to be running a SSH service on that host and port.
run_file	filename|IP_X:port	test.txt|IP_1:22	Same as above, but for run.
kill_file	filename|IP_X:port	test.txt|IP_1:22	Same as above.
decrypt_file	filename	log.data	Decrypt command is ran on file.
nmap	IP_X	IP_2	IP is nmapped.
cat	filename	story.text	File is read.
delete_logs	IP:port	IP_2:22	rm logs is executed on IP:port
delete_email	ID|IP_X:port	0|IP_3:345	An email with id ID is deleted from give host on given port. Obviously an smtp service needs to be running on there.
read_email	ID|IP_X:port	1|IP_2:666	The given email is read.
forward_email	ID|name@domain.ext|IP_X:port	2|USERNAME@secretrepublic.net|IP_2:33 3|grandma@yahoo.com|IP_3:29	When the given email is forwarded to the given email from the given service running on the given IP:port.

crash_service	ip:port	IP_3:80	Service is crashed, e.g. ddos is completed on ip:port.
sql_delete	table|row1|ip:port table|row1,row2,...,row(n)|ip:port	users|1,2|1.1.1.1:200	Delete specific rows from table. First row is 1, second 2 etc. You can specify as many rows as you possibly dream of. **WARNING** If you specify multiple rows, they must be deleted from one single instruction, if done one by one => objective fail. **ALTERNATIVE** is to create one compulsory side-objective for each row you want deleted.
sql_drop	table|IP:port	users|IP_1:1433	When a table is dropped.

Hosts

- **Hostname** ~ Usually displayed when pinging, connecting to it. E.g. backup.tsrbank.com
- **Discovered** ~ In order for a host to be displayed on the users list inside the mission he needs to ping it. If you mark it as discovered by default, it will become visible on the list by default.
- **MAC Address** ~ Generated random when quest starts. e.g. : 00:EF:6F:D1:76:25. Appears when running ifconfig. **No real use for now.**

Hosts receive (per quest) ID's starting from 0. The first host is always the hacker himself. The first host is locked and can have only an SSH file server. Whatever files you put there are available to the hacker immediately after the start of the quest. He is by default connected to the first host and when he disconnects from another host he gets connected back to the this host, to IP_0.

Every host is assigned a random IP. You can refer to the IP of host X using IP_X as described above.

Services

Are childs of hosts. Services run on given ports on given hosts. User has to interact differently with each service. Cannot add services to the main host (0). **You cannot run two service on the same port.** The later service will replace the first one in this case.

- **Port** ~ An integer specifying the port on which the server is running. You might want to google the usual port for certain services so that it's more realistic.
- **Encryption** ~ Service protection. Needs to be cracked in order to connect to host.
- **Welcome message** ~ Displayed after the protection is cracked and user connects. **BBCode enabled.** e.g. FBI database server 132. Welcome, USERNAME.

SSH Service

SSH services contain files and allows usual linux commands to be executed such as ls, rm, etc. On SSH files can be ran or killed, etc.

Files

- **Title** ~ Filename + extension if any. **If you put a . in front of the name, e.g. ".file" it will become a hidden file. Hidden files can be viewed using ls -a. Pick sensible filenames. Don't use anything else than alpha-numeric characters and dots (NO SPACES!!).**

- **Encryption** ~ Defines if it requires decryption or not and how long that will last.
- **Can Run** ~ Defines if the file can be executed.
- **Can't kill** ~ Can the file be killed or Cardinal OS should decline the action.
- **Running** ~ Defines if it is running by default.
- **Required running on current host** ~ Files must be separated by | (e.g. file1.t|file2.d)
- **Required running on main server** ~ Like above just that for main.
- **Parameters** ~ Separated by |. (e.g. for **run file.t 123 345** use **123|345**)
- **Required (Side)Objective** - file remains fully non-existent up till completion.
- **Size ~** Currently irrelevant
- **Burst ~** The file will execute then kill itself. This kind of is useful only if you tie it to files which are tied to side objectives which are tied to running of files with burst enabled so that they give access to other files once ran. For example an account cracker does not need to stay running but run, give access to some extra data then die quietly.
- **Execution Time ~** Simple, right? The amount of seconds the file will need before starting up and triggering the run_file event.

SMTP Service

Contains just emails. Allows specific commands.

Emails

- **ID** ~ Assigned uniquely automatically. E.g. first is 0, last is numberOfEmails-1
- **Title/Subject** ~ Self-explanatory.
- **Sender** ~ Name or an email address.
- **Content** ~ **BBCode enabled**
- **Required (Side)Objective** - email remains fully non-existent up till completion.

HTTP Service

Means a webserver is running on that port. **No specific functionality implemented** other than its existence. Can be DDOSed.

SQL Service

This is a tricky-ish one, maybe? Usually SQL has databases which have tables which have columns, rows and data.

Tables

- **Title** ~ Should be made only of letters, maybe a number at the end if you're keen on it
- **Content** ~ This is the not-so-tricky-but-tricky-ish-part.

e.g. of content:

```
log_id|ip|date
1|IP_3|13/2/2014
2|IP_4|14/2/2014
```

The first row defines the column headers, separated by |.
The next rows define the table rows, with a value for each defined column, separated again by |.

Column names should not contain spaces or weird stuff. You should use alphanumeric and _'s.

UNKNOWN Service

You can use this when you simply want the user to crack a port before you show him the next objective. For example I used it as a port on a car which the user had to crack then execute a file with the IP and port as parameters to break into the car.

TIPS AND TRICKS

Outputs for runnable files

If you want to generate an output file when another file is ran (for example let's say you've created scanner.sh and you want to show the user foundIPs.txt after scanner.sh is ran) you can add a side-optional-objective which gets completed when scanner.sh is ran and attach the side objective to the file foundIPs.txt so that it becomes visible only when that side is completed.

Wildcard * IP's and PORTs in objective data syntax

Whenever you have IP_X:PORT in Data Syntax you can use IP_X:*, or *:PORT or *:*. * is a wildcard: IP_X:'any port', 'any IP':PORT, 'any IP':'any port'.

NOW THAT WAS A WALL OF TEXT!

GET 2 WORK, NOW!