Events

Navigation

These are the events fired by Aggressor Script.

*

This event fires whenever any Aggressor Script event fires.

Arguments

`$1` - the original event name
`...` - the arguments to the event

Example

```
# event spy script
on * {
	println("[ $+ $1 $+ ]: " . subarray(@_, 1));
}
```

beacon_checkin

Fired when a Beacon checkin acknowledgement is posted to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the message
`$3` - when this message occurred

beacon_error

Fired when an error is posted to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the message
`$3` - when this message occurred

beacon_indicator

Fired when an indicator of compromise notice is posted to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the user responsible for the input
`$3` - the text of the message
`$4` - when this message occurred

beacon_initial

Fired when a Beacon calls home for the first time.

Arguments

`$1` - the ID of the beacon that called home.

Example

```
on beacon_initial {
        # list network connections
        bshell($1, "netstat -na | findstr \"ESTABLISHED\"");

        # list shares
        bshell($1, "net use");

        # list groups
        bshell($1, "whoami /groups");
}
```

beacon_initial_empty

Fired when a DNS Beacon calls home for the first time. At this point, no metadata has been exchanged.

Arguments

`$1` - the ID of the beacon that called home.

Example

```
on beacon_initial_empty {
        binput($1, "[Acting on new DNS Beacon]");

        # change the data channel to DNS TXT
        bmode($1, "dns-txt");

        # request the Beacon checkin and send its metadata
        bcheckin($1);
}
```

beacon_input

Fired when an input message is posted to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the user responsible for the input
`$3` - the text of the message
`$4` - when this message occurred

beacon_mode

Fired when a mode change acknowledgement is posted to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the message
`$3` - when this message occurred

beacon_output

Fired when output is posted to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the message
`$3` - when this message occurred

beacon_output_alt

Fired when (alternate) output is posted to a Beacon's console. What makes for alternate output? It's just different presentation from normal output.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the message
`$3` - when this message occurred

beacon_output_jobs

Fired when jobs output is sent to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the jobs output
`$3` - when this message occurred

beacon_output_ls

Fired when ls output is sent to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the ls output
`$3` - when this message occurred

beacon_output_ps

Fired when ps output is sent to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the ps output
`$3` - when this message occurred

beacon_tasked

Fired when a task acknowledgement is posted to a Beacon's console.

Arguments

`$1` - the ID of the beacon
`$2` - the text of the message
`$3` - when this message occurred

beacons_update

Fired when the team server sends over fresh information on all of our Beacons. This occurs about once each second.

Arguments

`$1` - an array of dictionary objects with metadata for each Beacon.

disconnect

Fired when this Cobalt Strike becomes disconnected from the team server.

event_action

Fired when a user performs an action in the event log. This is similar to an action on IRC (the /me command)

Arguments

`$1` - who the message is from
`$2` - the contents of the message
`$3` - the time the message was posted

event_beacon_initial

Fired when an initial beacon message is posted to the event log.

Arguments

`$1` - the contents of the message
`$2` - the time the message was posted

event_join

Fired when a user connects to the team server

Arguments

`$1` - who joined the team server
`$2` - the time the message was posted

event_newsite

Fired when a new site message is posted to the event log.

Arguments

`$1` - who setup the new site
`$2` - the contents of the new site message
`$3` - the time the message was posted

event_notify

Fired when a message from the team server is posted to the event log.

Arguments

`$1` - the contents of the message
`$2` - the time the message was posted

event_nouser

Fired when the current Cobalt Strike client tries to interact with a user who is not connected to the team server.

Arguments

`$1` - who is not present
`$2` - the time the message was posted

event_private

Fired when a private message is posted to the event log.

Arguments

`$1` - who the message is from
`$2` - who the message is directed to
`$3` - the contents of the message
`$4` - the time the message was posted

event_public

Fired when a public message is posted to the event log.

Arguments

`$1` - who the message is from
`$2` - the contents of the message
`$3` - the time the message was posted

event_quit

Fired when someone disconnects from the team server.

Arguments

`$1` - who left the team server
`$2` - the time the message was posted

heartbeat_10m

Fired every ten minutes

heartbeat_10s

Fired every ten seconds

heartbeat_15m

Fired every fifteen minutes

heartbeat_15s

Fired every fifteen seconds

heartbeat_1m

Fired every minute

heartbeat_1s

Fired every second

heartbeat_20m

Fired every twenty minutes

heartbeat_30m

Fired every thirty minutes

heartbeat_30s

Fired every thirty seconds

heartbeat_5m

Fired every five minutes

heartbeat_5s

Fired every five seconds

heartbeat_60m

Fired every sixty minutes

keylogger_hit

Fired when there are new results reported to the web server via the cloned site keystroke logger.

Arguments

`$1` - external address of visitor
`$2` - reserved
`$3` - the logged keystrokes
`$4` - the phishing token for these recorded keystrokes.

profiler_hit

Fired when there are new results reported to the System Profiler.

Arguments

`$1` - external address of visitor
`$2` - de-cloaked internal address of visitor (or "unknown")
`$3` - visitor's User-Agent
`$4` - a dictionary containing the applications.
`$5` - the phishing token of the visitor (use &tokenToEmail to resolve to an email address)

ready

Fired when this Cobalt Strike client is connected to the team server and ready to act.

sendmail_done

Fired when a phishing campaign completes

Arguments

`$1` - the campaign ID

sendmail_post

Fired after a phish is sent to an email address.

Arguments

`$1` - the campaign ID
`$2` - the email we're sending a phish to
`$3` - the status of the phish (e.g., SUCCESS)
`$4` - the message from the mail server

sendmail_pre

Fired before a phish is sent to an email address.

Arguments

`$1` - the campaign ID
`$2` - the email we're sending a phish to

sendmail_start

Fired when a new phishing campaign kicks off.

Arguments

`$1` - the campaign ID
`$2` - number of targets
`$3` - local path to attachment
`$4` - the bounce to address
`$5` - the mail server string
`$6` - the subject of the phishing email
`$7` - the local path to the phishing template
`$8` - the URL to embed into the phish

web_hit

Fired when there's a new hit on Cobalt Strike's web server.

Arguments

`$1` - the method (e.g., GET, POST)
`$2` - the requested URI
`$3` - the visitor's address
`$4` - the visitor's User-Agent string
`$5` - the web server's response to the hit (e.g., 200)
`$6` - the size of the web server's response
`$7` - a description of the handler that processed this hit.
`$8` - a dictionary containing the parameters sent to the web server
`$9` - the time when the hit took place.