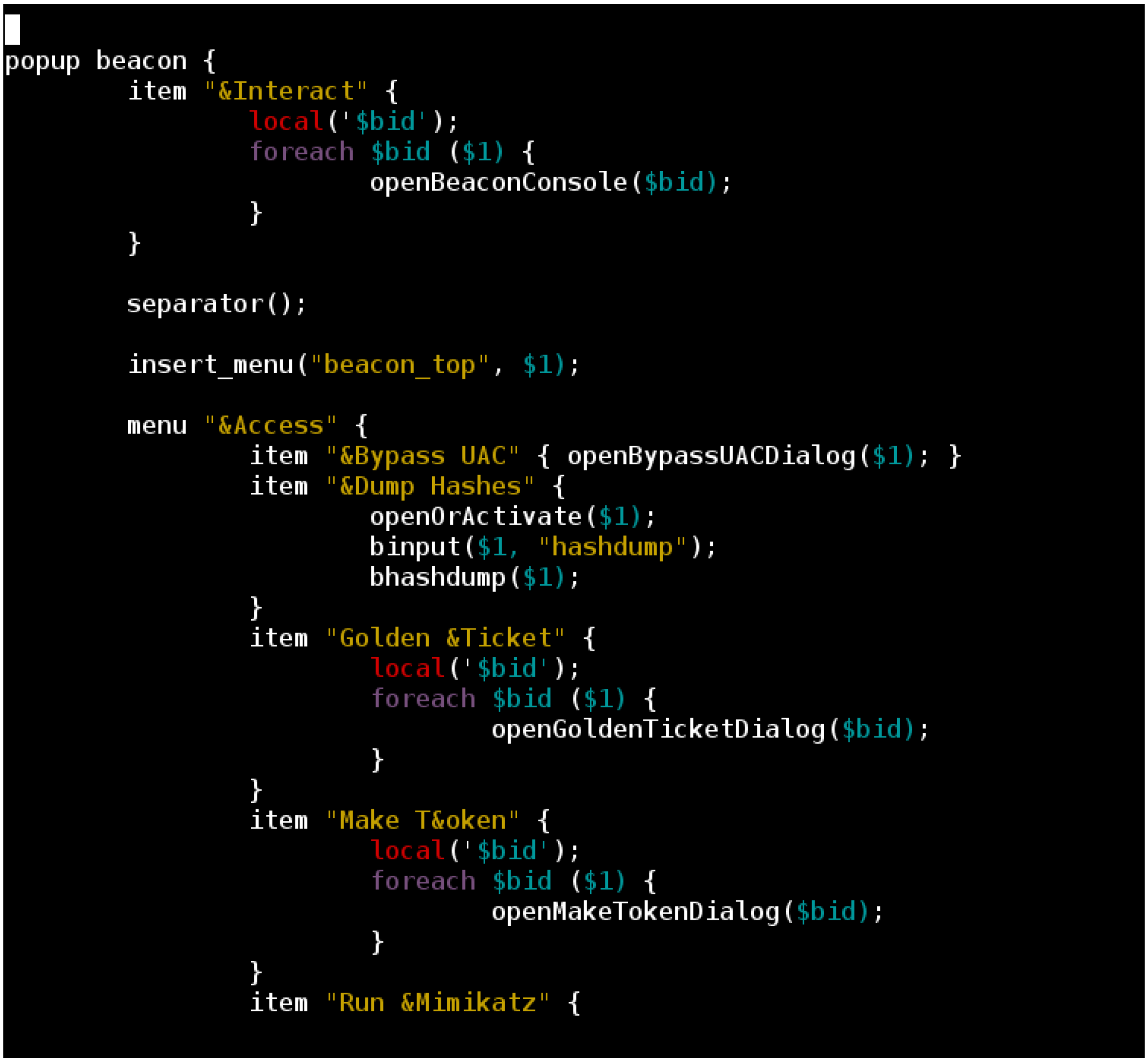
2. Cobalt Strike

The Cobalt Strike Client

The Aggressor Script engine is the glue feature in Cobalt Strike. Most Cobalt Strike dialogs and features are written as stand-alone modules that expose some interface to the Aggressor Script engine.

An internal script, default.cna (default.cna), defines the default Cobalt Strike experience. This script defines Cobalt Strike's toolbar buttons, popup menus, and it also formats the output for most Cobalt Strike events.

This chapter will show you how these features work and empower you to shape the Cobalt Strike client to your needs.



```
popup beacon {
        item "&Interact" {
                local('$bid');
                foreach $bid ($1) {
                        openBeaconConsole($bid);
                }
        }

        separator();

        insert_menu("beacon_top", $1);

        menu "&Access" {
                item "&Bypass UAC" { openBypassUACDialog($1); }
                item "&Dump Hashes" {
                        openOrActivate($1);
                        binput($1, "hashdump");
                        bhashdump($1);
                }
                item "Golden &Ticket" {
                        local('$bid');
                        foreach $bid ($1) {
                                openGoldenTicketDialog($bid);
                        }
                }
                item "Make T&oken" {
                        local('$bid');
                        foreach $bid ($1) {
                                openMakeTokenDialog($bid);
                        }
                }
                item "Run &Mimikatz" {
```

The default.cna script

Keyboard Shortcuts

Scripts may create keyboard shortcuts. Use the **bind** keyword to bind a keyboard shortcut. This example shows **Hello World!** in a dialog box when Ctrl and H are pressed together.

```
bind Ctrl+H {
        show_message("Hello World!");
}
```

Keyboard shortcuts may be any ASCII characters or a special key. Shortcuts may have one or more modifiers applied to them. A modifier is one of: Ctrl, Shift, Alt, or Meta. Scripts may specify the modifier+key.

Popup Menus

Scripts may also add to Cobalt Strike's menu structure or re-define it. The popup keyword builds a menu hierarchy for a popup hook.

Here's the code that defines Cobalt Strike's help menu:

```
popup help {
        item("&Homepage", { url_open("https://www.cobaltstrike.com/"); });
        item("&Support",  { url_open("https://www.cobaltstrike.com/support"); });
        item("&Arsenal",  { url_open("https://www.cobaltstrike.com/scripts?license=" . licenseKey()); }
);
        separator();
        item("&System Information", { openSystemInformationDialog(); });
        separator();
        item("&About", { openAboutDialog(); });
}
```

This script hooks into the help popup hook and defines several menu items. The & in the menu item name is its keyboard accelerator. The code block associated with each item executes when the user clicks on it.

Scripts may define menus with children as well. The menu keyword defines a new menu. When the user hovers over the menu, the block of code associated with it is executed and used to build the child menu.

Here's the Pivot Graph menu as an example of this:

```
popup pgraph {
        menu "&Layout" {
                item "&Circle"    { graph_layout($1, "circle"); }
                item "&Stack"     { graph_layout($1, "stack"); }
                menu "&Tree" {
                        item "&Bottom" { graph_layout($1, "tree-bottom"); }
                        item "&Left"   { graph_layout($1, "tree-left"); }
                        item "&Right"  { graph_layout($1, "tree-right"); }
                        item "&Top"    { graph_layout($1, "tree-top"); }
                }
                separator();
                item "&None" { graph_layout($1, "none"); }
        }
}
```

If your script specifies a menu hierarchy for a Cobalt Strike menu hook, it will add to the menus that are already in place. Use the &popup_clear (functions.html#popup_clear) function to clear the other registered menu items and re-define a popup hierarchy to your taste.

Custom Output

The set keyword in Aggressor Script defines how to format an event and present its output to the user. Here's an example of the set keyword:

```
set EVENT_SBAR_LEFT {
        return "[" . tstamp(ticks()) . "] " . mynick();
}

set EVENT_SBAR_RIGHT {
        return "[lag: $1 $+ ]";
}
```

The above code defines the content of the statusbar in Cobalt Strike's Event Log (**View -> Event Log**). The left side of this statusbar shows the current time and your nickname. The right side shows the round-trip time for a message between your Cobalt Strike client and the team server.

You may override any set option in the Cobalt Strike default script. Create your own file with definitions for events you care about. Load it into Cobalt Strike. Cobalt Strike will use your definitions over the built-in ones.

Events

Use the on keyword to define a handler for an event. The ready event fires when Cobalt Strike is connected to the team server and ready to act on your behalf.

```
on ready {
        show_message("Ready for action!");
}
```

Cobalt Strike generates events for a variety of situations. Use the * meta-event to watch all events Cobalt Strike fires.

```
on * {
        local('$handle $event $args');

        $event = shift(@_);
        $args  = join(" ", @_);

        $handle = openf(">>eventspy.txt");
        writeb($handle, "[ $+ $event $+ ] $args");
        closef($handle);
}
```