Conditional Statements

Overview

Previously, original DuckyScript payloads executed sequentially — line by line from start to finish.

With DuckyScript 3.0, it isn't necessary for payload execution to be linear. Conditional statements, loops and functions allow for dynamic execution.

`IF`

The flow control statement `IF` will determine whether or not to execute its block of code based on the evaluation of an expression. One way to interpret an `IF` statement is to read it as "`IF` this condition is true, `THEN` do this".

Syntax

- The IF statement consists of these parts
- The `IF` keyword
- The condition, or expression that evaluates to `TRUE` or `FALSE`In nearly all cases, the expression should be surrounded by parenthesis `( )`
- The `THEN` keyword
- One or more newlines containing the block of code to execute
- The `END_IF` keyword

Example

```
REM Example IF THEN

$FOO = 42
$BAR = 1337

IF ( $FOO < $BAR ) THEN
```

```
    STRING 42 is less than 1337
END_IF
```

Result

- The expression "Is 42 less than 1337" is evaluated and determined to be `TRUE`.
- Because the `IF` condition is `TRUE`, the code between the keywords `THEN` and `END_IF` are executed.
- The string "`42 is less than 1337`" is typed.

`ELSE`

The ELSE statement is an optional component of the IF statement which will only execute when the IF statement condition is FALSE. One way to interpret an `ELSE` statement is to read it as "`IF` this condition is true, `THEN` do this thing, or `ELSE` do another thing".

Example

```
REM Example IF THEN ELSE

IF ( $_CAPSLOCK_ON == TRUE ) THEN
    STRING Capslock is on!
ELSE IF ( $_CAPSLOCK_ON == FALSE ) THEN
    STRING Capslock is off!
END_IF
```

Result

- The condition of the capslock key, as determined by the target operating system, is checked.
- If the capslock key state has been reported by the target as ON, the string "`Capslock is on`" will be typed.
- Otherwise, if the capslock key state has not been reported by the target (or it has been reported as not being on), the string "`Capslock is off`" will be typed.

Nested `IF` Statements

A nested IF statement is quite simply an IF statement placed inside another IF statement. Nested IF statements may be used when evaluating a combination of conditions.

Example

```
REM Example nested IF statements

IF ( $_CAPSLOCK_ON == TRUE ) THEN
    IF ( $_NUMLOCK_ON == TRUE ) THEN
        STRING Both Capslock and Numlock are on!
    END_IF
END_IF
```

Result

- The condition of the first `IF` statement is evaluated — whether or not the target has reported that the Capslock key is on. If it is `TRUE`, then the nested `IF` statement will run.
- The second `IF` statement is evaluated much like the first, only this time checking the status of the Numlock key.
- If both the capslock and numlock keys have been reported by the target as being on, then the string "`Both Capslock and Numlock are on!`" will be typed.

`IF` Statements with logical operators

In some cases it may be more efficient to use logical operators within a single IF statement, rather than using a nested IF structure.

Example

```
REM Example IF statement with logical operators

IF (( $_CAPSLOCK_ON == TRUE ) && ( $_NUMLOCK_ON == TRUE )) THEN
```

```
    STRING Both Capslock and Numlock are on!
END_IF
```

Result

- Because the AND logical operator is in use, the whole condition will only evaluate as TRUE if both sub conditions are TRUE.
- Similar to the Nested IF example, the string "`Both Capslock and Numlock are on!`" will only be typed if both capslock and numlock are reported by the target as being on.

`IF` Statement Optimization

The syntax of `IF` states that in nearly all cases the expression should be surrounded by parenthesis `( )` — however there is an exception to this rule.

If the condition of only one variable is *true* or *false*, the parenthesis may be omitted. This results in a slightly smaller encoded `inject.bin` file as well as slightly faster payload execution. This is because it removes the step of first reducing the order precedence.

Example

```
REM Example of optimized and unoptimized IF statements

REM Consider

VAR $FLAG = TRUE

IF $FLAG THEN
    STRING FLAG is TRUE
END_IF

REM versus

IF ( $FLAG == TRUE ) THEN
    STRING FLAG is TRUE
END_IF
```

Result

- In the first example, the `IF` statement without the parenthesis results in a 6 bytes added to the compiled `inject.bin` file.
- In the second example, the `IF` statement surrounded by parenthesis results in 16 bytes added to the compiled `inject.bin` file.

Example

```
REM Example of optimized IF statement with internal variable

IF $_CAPSLOCK_ON THEN
    STRINGLN The caps lock key is on
END_IF
```

Result

- The internal variable `$_CAPSLOCK_ON` is checked.
- If it evaluates as `TRUE`, the message "`The caps lock key is on`" is typed.